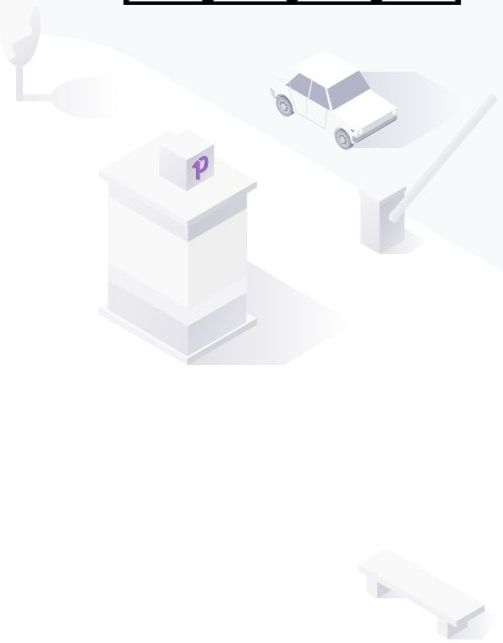
prima

ROMHACK CAMP



You shall not PassRole!

AWS Privilege Escalation and Defense

24th September 2022

whoami.

aws sts get-caller-identity

Edoardo Rosa - dodo

Security Engineer @ Prima Assicurazioni

~~old~~ senior member of Cesena Security and Network Application (*CeSeNA*)

Experience on Blue and Red Teaming, penetration testing on on-premise and cloud infrastructures with a passion on defences (and bypasses) and automation.

notdodo

*_notdodo_*

Agenda

01 **AWS** - Basic knowledge and how it works.

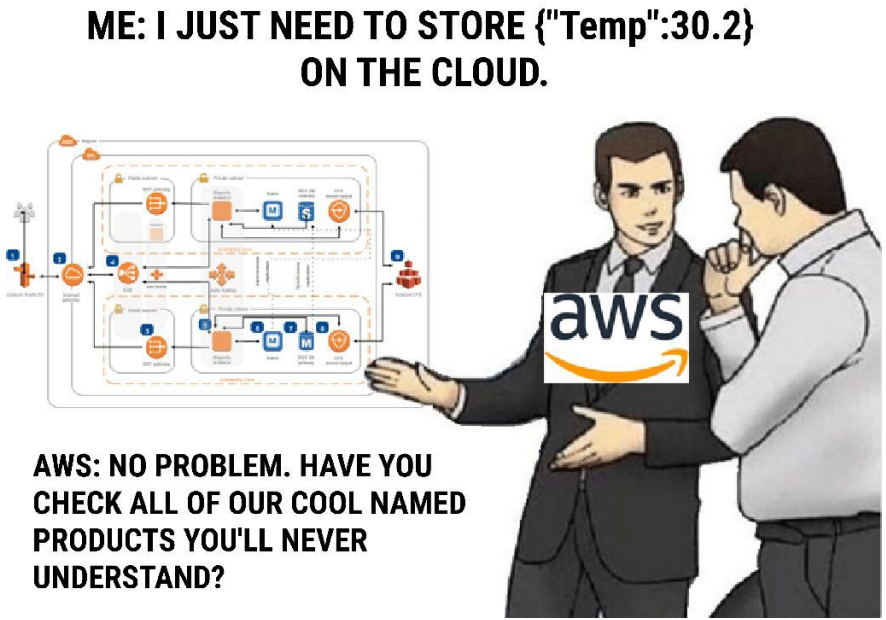
02 **Attack Methodology** - Same but different.

03 **Demos** - Privilege Escalation in practice.

04 **The problem** - A real problem.

05 **nuvola** - Helping to solve the problem.

06 **Conclusions**



AWS.

Basic knowledge and how it works.

S3 Buckets

S3 buckets can be used a container to store objects (files, logs, apps data, etc.)

- encryption
- access logs
- tiering
- access policies

EC2 instance

Amazon EC2 instance is a multi purpose virtual server

- impersonates an AWS role
- can scale up/down
- admin can login using SSM
- different OSes

Lambda Function

A Lambda function executes some code after an event is triggered.

- impersonates an AWS role
- execution time is short
- totally managed by AWS

IAM

Identity and Access Management is a **fundamental and critical cybersecurity capability**, especially on cloud environments where cloud users rely on services, like AWS Identity and Access Management (IAM), to **secure** and **manage access** across the variety of services and resources.

Users' group membership:

- *Jon* is a new hire and it is assigned to the *ReadOnlyUsers* group

User ARN

User name

Groups membership

Group ARN

Group name


```
"Arn": "arn:aws:iam::111111111111:user/Jon",
"CreateDate": "2021-09-18T11:15:41Z",
"Path": "/",
"UserId": "AIDAWFKZ7XCLPEWT3YLPD",
"UserName": "Jon",
"PasswordLastUsed": "2022-09-10T14:03:20Z",
"PermissionsBoundary": null,
"Tags": null,
"PasswordEnabled": "false",
"PasswordLastChanged": "N/A",
"MFAStatus": "false",
"Groups": [
    {
        "Arn": "arn:aws:iam::111111111111:group/ReadOnlyUsers",
        "CreateDate": "2021-09-18T11:15:33Z",
        "GroupId": "AGPAWFKZ7XCLL53Q5FOMR",
        "GroupName": "ReadOnlyUsers",
        "Path": "/"
    }
],
```

Users' group membership:

- *Jon* is a new hire and it is assigned to the *ReadOnlyUsers* group

User ARN

User name

Groups membership

Group ARN

Group name

```
"Arn": "arn:aws:iam::111111111111:user/Jon",
"CreateDate": "2021-09-18T11:15:41Z",
"Path": "/",
"UserId": "AIDAWFKZ7XCLPEWT3YLPD",
"UserName": "Jon",
"PasswordLastUsed": "2022-09-10T14:03:20Z",
"PermissionsBoundary": null,
"Tags": null,
"PasswordEnabled": "false",
"PasswordLastChanged": "N/A",
"MFAStatus": "false",
"Groups": [
    {
        "Arn": "arn:aws:iam::111111111111:group/ReadOnlyUsers",
        "CreateDate": "2021-09-18T11:15:33Z",
        "GroupId": "AGPAWFKZ7XCLL53Q5FOMR",
        "GroupName": "ReadOnlyUsers",
        "Path": "/"
    }
],
```

Users' group membership:

- *Jon* is a new hire and it is assigned to the *ReadOnlyUsers* group

User ARN

User name

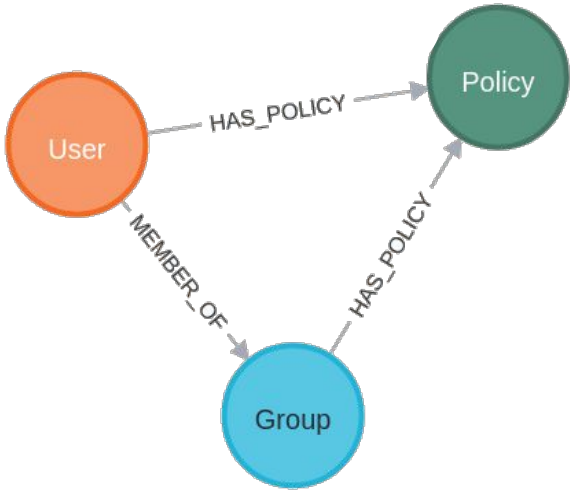
Groups membership

Group ARN

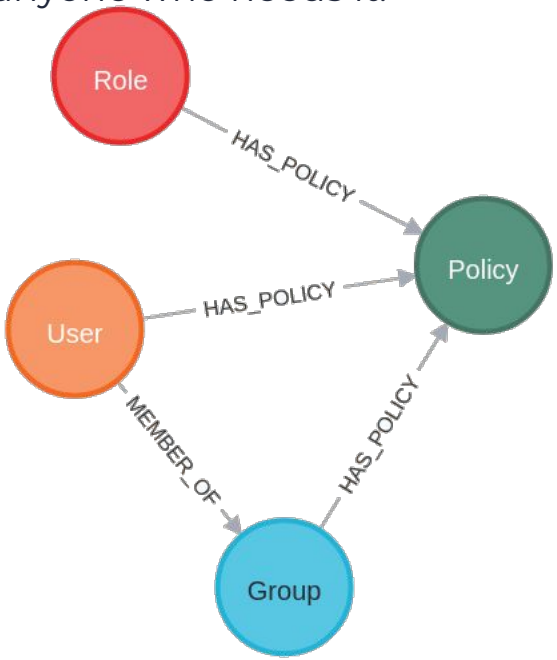
Group name


```
"Arn": "arn:aws:iam::111111111111:user/Jon",
"CreateDate": "2021-09-18T11:15:41Z",
"Path": "/",
"UserId": "AIDAWFKZ7XCLPEWT3YLPD",
"UserName": "Jon",
"PasswordLastUsed": "2022-09-10T14:03:20Z",
"PermissionsBoundary": null,
"Tags": null,
"PasswordEnabled": "false",
"PasswordLastChanged": "N/A",
"MFAStatus": "false",
"Groups": [
    {
        "Arn": "arn:aws:iam::111111111111:group/ReadOnlyUsers",
        "CreateDate": "2021-09-18T11:15:33Z",
        "GroupId": "AGPAWFKZ7XCLL53Q5FOMR",
        "GroupName": "ReadOnlyUsers",
        "Path": "/"
    }
],
```

Jon can then be enabled to execute some ***actions*** using a ***policy*** directly or using the Group membership



Instead of being uniquely associated with one person, a **role** is intended to be assumable by anyone who needs it.



Role ARN

Role name

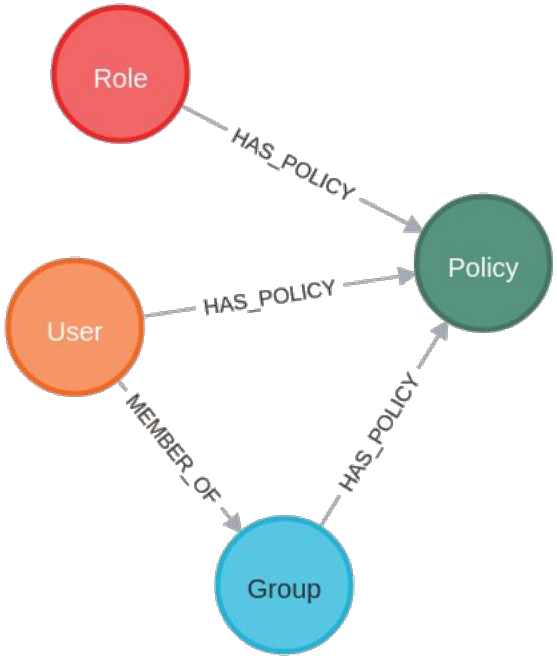
List of assigned policies

Policy name

List of allowed permissions


```
"Arn": "arn:aws:iam::111111111111:role/role1",
"CreateDate": "2022-09-08T08:12:30Z",
"Path": "/",
"RoleId": "AROAWFKZ7XCLIDPHUPWGO",
"RoleName": "demo1-cloudformation-deployer",
"MaxSessionDuration": 3600,
"PermissionsBoundary": null,
"RoleLastUsed": null,
"Tags": null,
"Description": "",
"InlinePolicies": [
    {
        "PolicyName": "Role1Policy",
        "Version": "2012-10-17",
        "Statement": [
            {
                "Effect": "Allow",
                "Action": [
                    "s3:ListBuckets",
                    "s3:PutObject",
                    "iam:CreateUser",
                ],
                "Resource": "*"
            }
        ]
    }
],
```

A **policy** defines what a role, group or user can or can't perform.



Role ARN

Role name

List of assigned policies

Policy name

List of allowed permissions

```
"Arn": "arn:aws:iam::111111111111:role/role1",
"CreateDate": "2022-09-08T08:12:30Z",
"Path": "/",
"RoleId": "AROAWFKZ7XCLIDPHUPWGO",
"RoleName": "demo1-cloudformation-deployer",
"MaxSessionDuration": 3600,
"PermissionsBoundary": null,
"RoleLastUsed": null,
"Tags": null,
"Description": "",
"InlinePolicies": [
    {
        "PolicyName": "Role1Policy",
        "Version": "2012-10-17",
        "Statement": [
            {
                "Effect": "Allow",
                "Action": [
                    "s3:ListBuckets",
                    "s3:PutObject",
                    "iam:CreateUser",
                ],
                "Resource": "*"
            }
        ]
    }
],
```

Action = *ServiceName* + ':' + *Operation*

Effect of the statement

Target Service

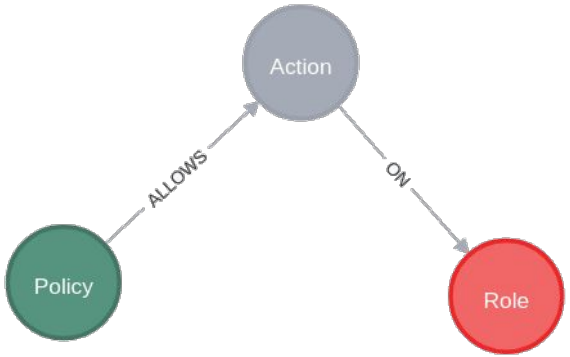
Target permission

Target objects

```
"Statement": [
    {
        "Effect": "Allow",
        "Action": [
            "iam:DeleteRole",
            "iam:ListRoles",
            "iam:AddUserToGroup",
            "s3:PutObject",
            "s3:ListBuckets",
            "ec2:*",
        ],
        "Resource": "*"
    },
```

Action = *ServiceName* + ':' + *Operation*

The *iam:DeleteRole* action, for example, must also be logically connected to the roles in AWS



Effect of the statement

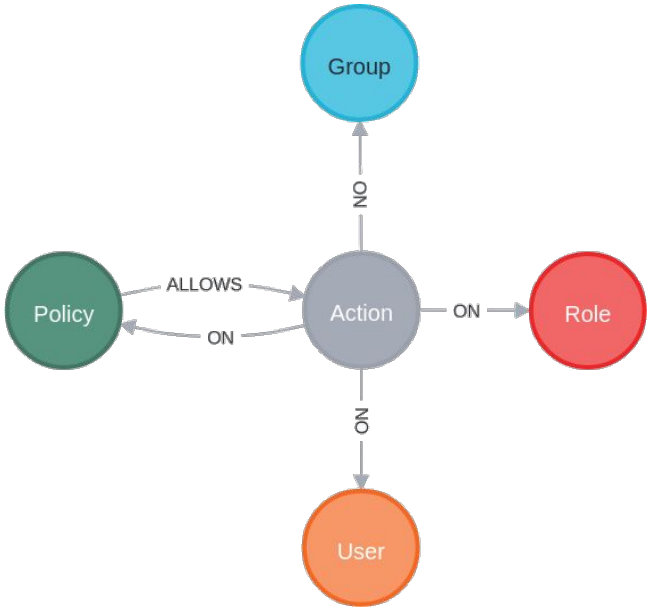
Target Service

Target permission

Target objects

```
"Statement": [
    {
        "Effect": "Allow",
        "Action": [
            "iam:DeleteRole",
            "iam:ListRoles",
            "iam:AddUserToGroup",
            "s3:PutObject",
            "s3:ListBuckets",
            "ec2:*",
        ],
        "Resource": "*"
    },
```

With *“iam:” actions* you can target any IAM object



Effect of the statement

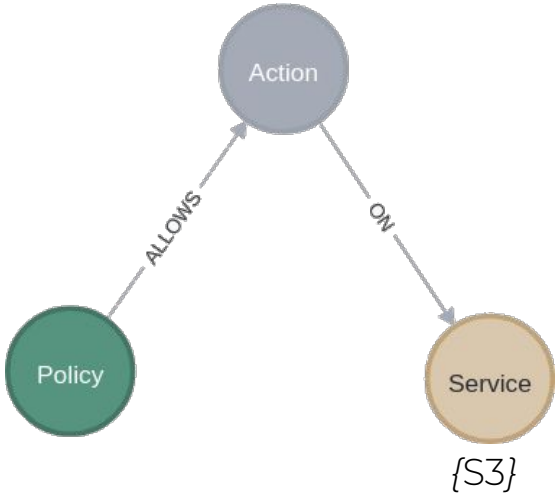
Target Service

Target permission

Target objects

```
"Statement": [
    {
        "Effect": "Allow",
        "Action": [
            "iam:DeleteRole",
            "iam:ListRoles",
            "iam:AddUserToGroup",
            "s3:PutObject",
            "s3:ListBuckets",
            "ec2:*",
        ],
        "Resource": "*"
    },
```

s3 actions targets only the S3 service



Effect of the statement

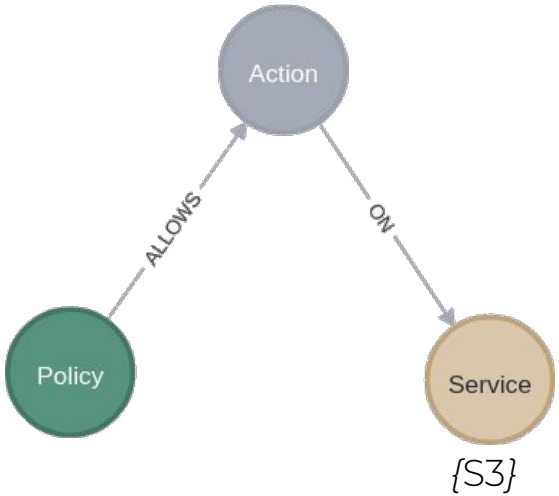
Target Service

Target permission

Target objects

```
"Statement": [
    {
        "Effect": "Allow",
        "Action": [
            "iam:DeleteRole",
            "iam:ListRoles",
            "iam:AddUserToGroup",
            "s3:PutObject",
            "s3:ListBuckets",
            "ec2:*",
        ],
        "Resource": "*"
    },
```

If “*” is specified all objects in the S3 service can be targeted



Effect of the statement

Target Service

Target permission

Target objects

```
"Statement": [
    {
        "Effect": "Allow",
        "Action": [
            "iam:DeleteRole",
            "iam:ListRoles",
            "iam:AddUserToGroup",
            "s3:PutObject",
            "s3:ListBuckets",
            "ec2:*",
        ],
        "Resource": "*"
    },
```

Attack Methodology.

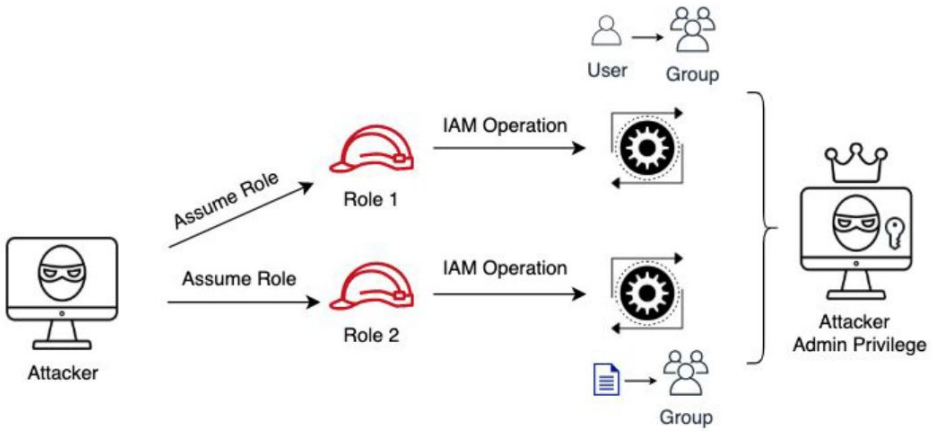
Same but different.

- Publicly **exposed cloud services**, like **web services**, aren't different to classical targets that attackers use to gain initial access on a company using **common or known vulnerabilities**

‘); EXEC xp_cmdshell ‘whoami’; ---

- **Misconfigurations**, **vulnerabilities** or **insider threats** highly increase the risk that unauthorised access is performed on data, users' accounts and other AWS services.
- The **tactics** remain the same whether is a cloud environment or an AD network; what changes are the **specific techniques**.
- The overall attack methodology is the same:
    a. **harvest** credentials/information
    b. **lateral movements** and **privileges escalation**
    c. repeat
    d. ...
    e. **profit**?

- Most the attacks on cloud environments are due to misconfigurations generated by:
  - lack of **awareness** of cloud security and policies
  - lack of adequate **controls** and **oversight**
  - **too many** APIs and interfaces to adequately govern
  - negligent **insider behavior**



- *iam:CreatePolicy*
  - the attackers creates a new policy that permits all AWS actions to himself

- *iam:PassRole* and *ec2:RunInstances*
  - the attackers creates an EC2 instance, pass a role to the instance with permissions that the user does not have currently

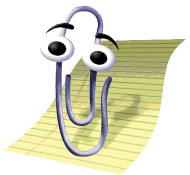
- *iam:PassRole*, *lambda:CreateFunction*, and *lambda:InvokeFunction*
  - the attackers *pass* an existing IAM role to a new Lambda function that includes code to import the relevant AWS library to perform actions of its choice

- *iam:CreatePolicyVersion*
  - the attackers cr… …hat permits all AWS actions
- *iam:PassRole* an…
  - the attackers cr… …le to the instance with permissions tha… …tly
- *iam:PassRole*, *la*… …d *lambda:InvokeFunction*
  - the attackers *p*… …v Lambda function that includes code to impo… …e relevant AWS library to perform actions of its choice

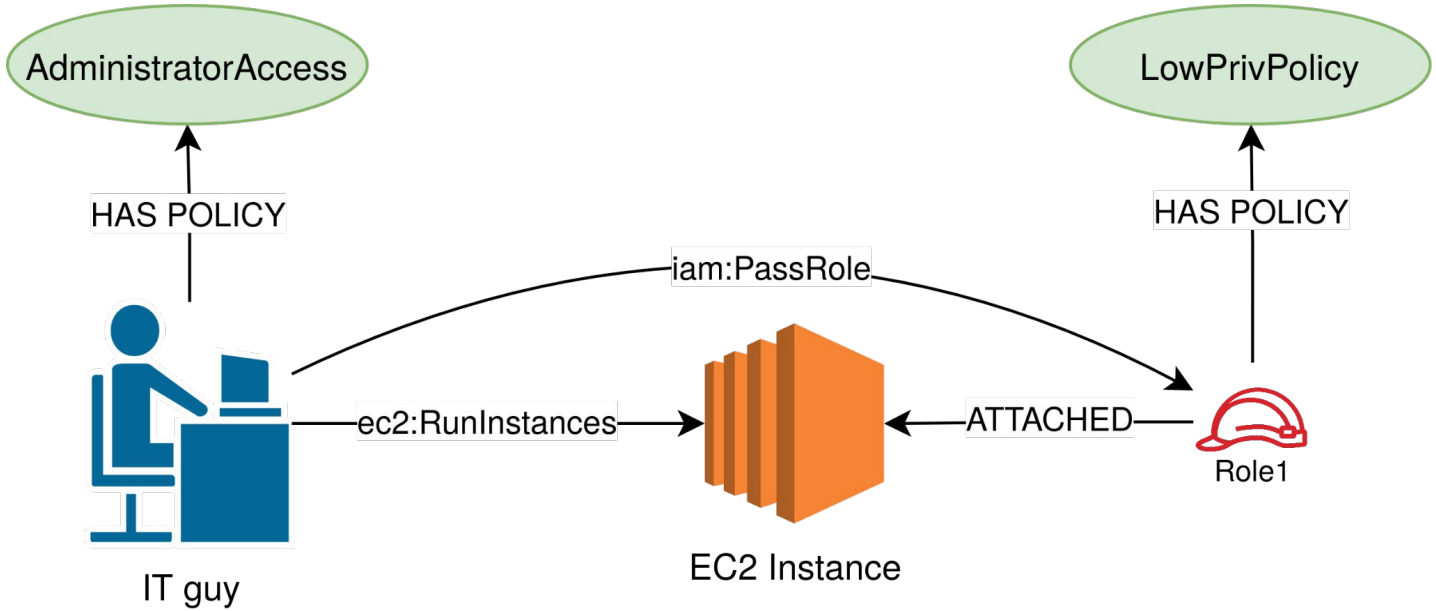
It looks like you have encountered the action “iam:PassRole” for the first time.

Would you like some help with that?

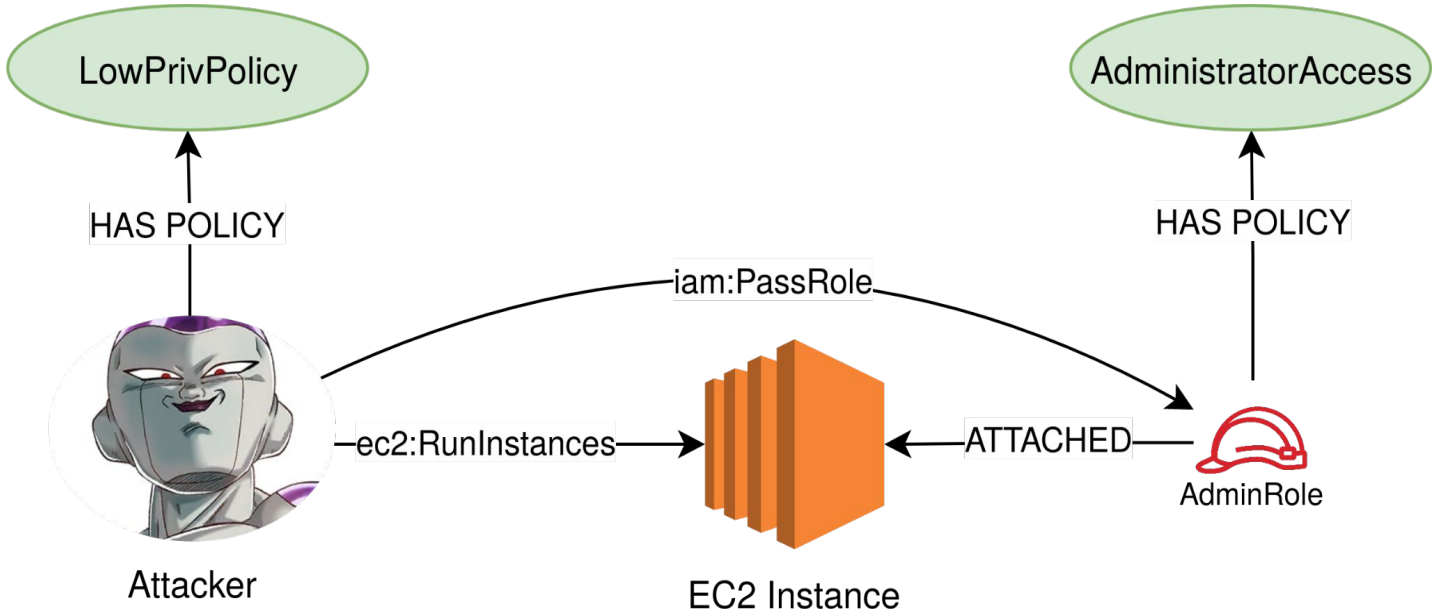
- Get help on “iam:PassRole”
- Do you think you have a choice?

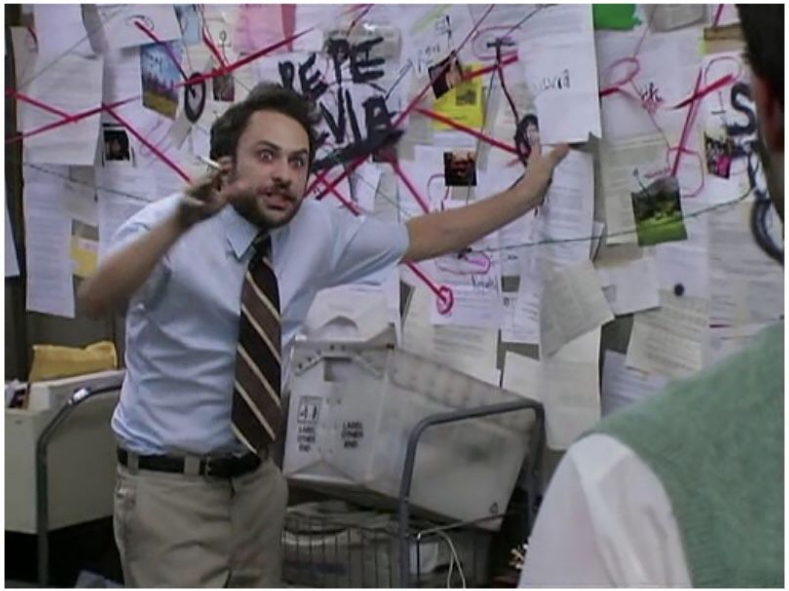


- *iam:PassRole is a permission granted to IAM Principals and resources that permits them to use an IAM Role on specific services* ***to perform actions on the Role behalf****.*



- *iam:PassRole is a permission **abused by attackers** to permits them to use an IAM Role on specific services to perform actions on the Role behalf.*





Demos.

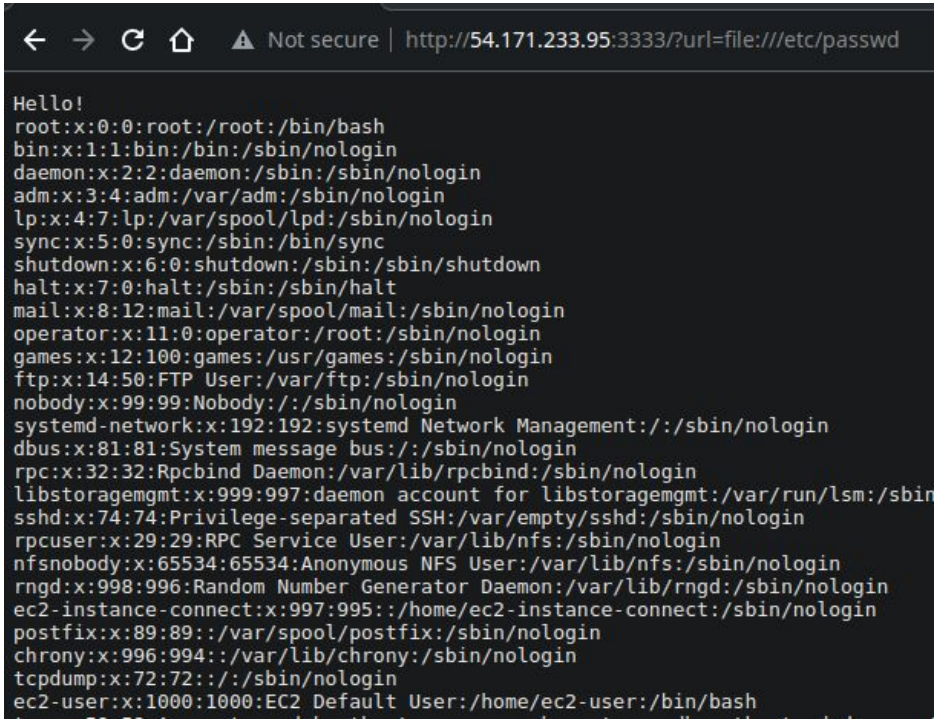
Privilege Escalations in practice.

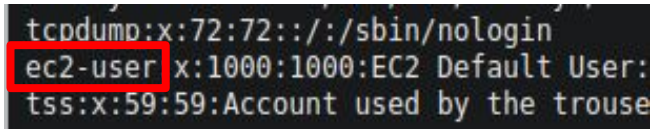
Ultima is slowly shifting to the cloud, porting most its **Java** on-premise services to AWS, using a **web portal** where logged and authorized users are able to upload and deploy **CloudFormation** stacks (IaC).

For each deployment file, **extensive security checks** are performed to avoid deploying **misconfigurations** or allowing **malicious activities**.

You, the attacker, were able to find an **SSRF vulnerability** that allowed you to perform exploration of the **local network**.

SSRF - Internal file system *http://<ip>:3333/?url=file:///etc/passwd*
SSRF - Internal network http://*<ip>*:3333/?url=http://localhost:3333



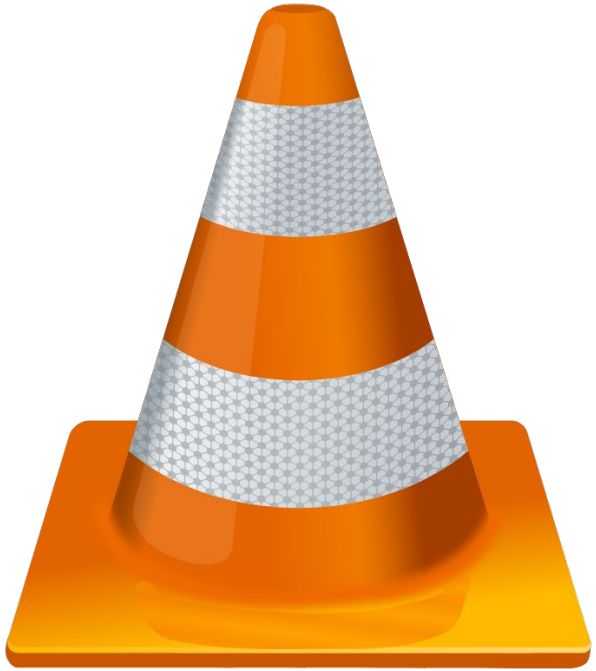


- EC2 instances have a metadata endpoint (**IMDS**) that is used to interact with the machine.

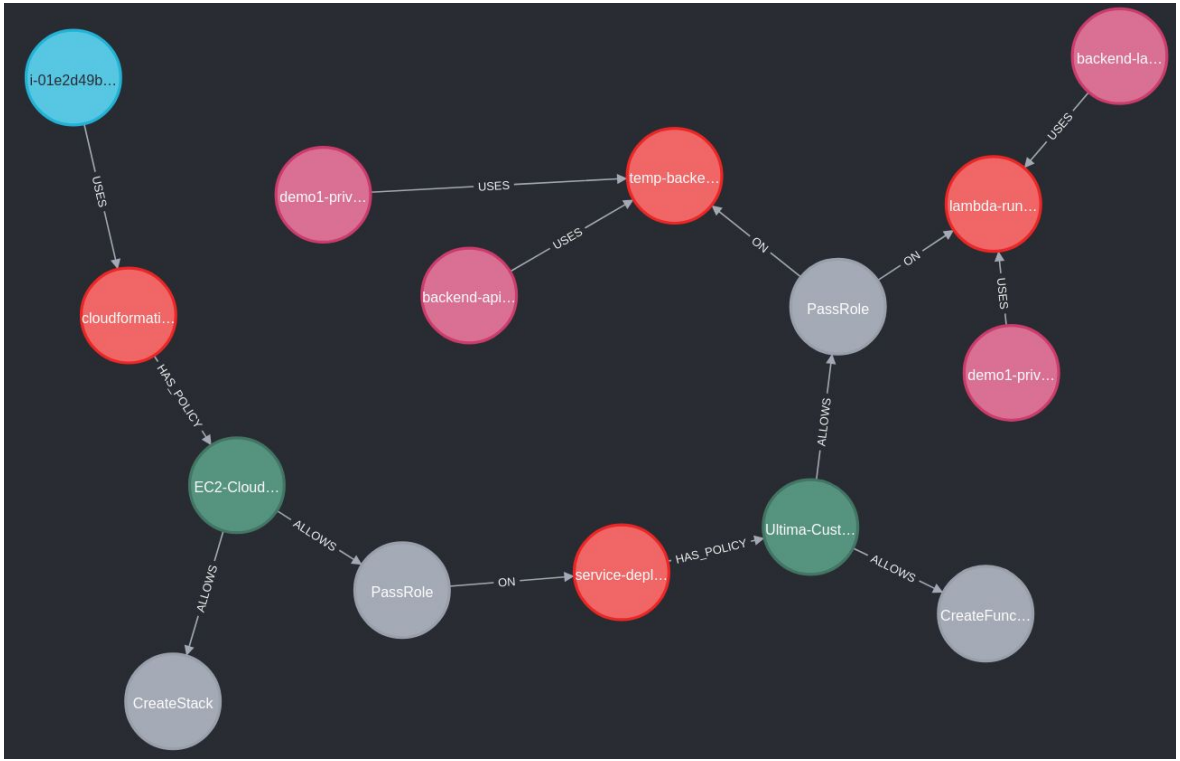
- *IMDSv1*, if not upgraded to use the version 2, can be used with SSRFs to **dump the credentials of the role** associated to the EC2.
  - http://169.254.169.254/latest/meta-data/iam/security-credentials/<**roleName**>
  - the endpoint is in ***Directory Listing*** *mode*

- https://github.com/SummitRoute/imdsv2_wall_of_shame

DEMO1



1. **EC2** access from **SSRF**
2. *cloudformation-deployer* credentials **dump**
3. CloudFormation stack to create a **lambda** using *lambda-runner* role passing the *service-deployer* role
4. **enumeration** of Lambdas
5. discovery of a new role: *temp-backend-api-role-runner*
6. CloudFormation stack to create a **lambda** using this new role with **Admin privs**



- **GuardDuty** will notify the administrators of an authorized access with the following signatures:
  - *UnauthorizedAccess:IAMUser/InstanceCredentialExfiltration.OutsideAWS*
  - UnauthorizedAccess:IAMUser/InstanceCredentialExfiltration.InsideAWS

```
"You have a severity 8 GuardDuty finding type UnauthorizedAccess:IAMUser/InstanceCredentialExfiltration.InsideAWS in the eu-west-1 region."
"Finding Description:"
"Credentials created exclusively for an EC2 instance using instance role demo1-role have been used from a remote AWS account 111111111111.. "
"For more details open the GuardDuty console at https://console.aws.amazon.com/guardduty/home?region=eu-west-1#/findings?search=id%3D01111111111111111111deadbeef11111"
```

- https://notdodo.medium.com/aws-guardduty-exfiltration-bypass-4720f6ed16a4

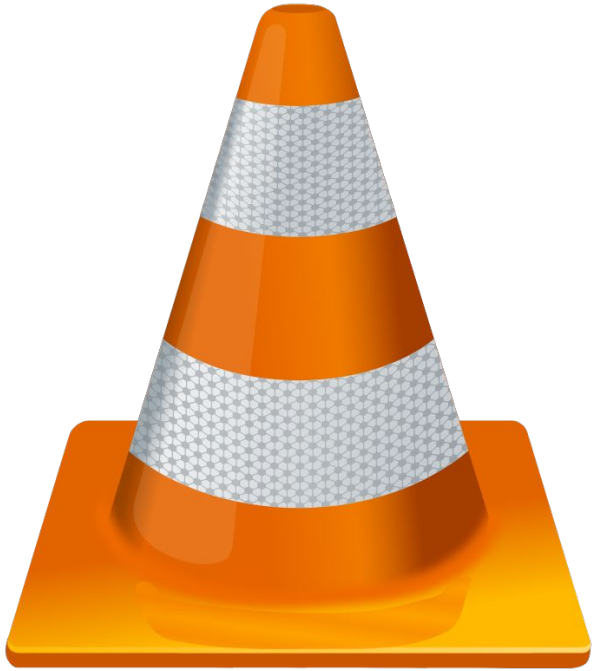
- Ultima learned a lot from the last incident and started creating **ad-hoc *Deny* policies** to prevent privilege escalations.

- The company started also to use all AWS services, even the ones to perform **data analytics and science** to keep pace with the other modern companies.

- Java was abandoned

You, the ~~attacker~~ disgruntled Ultima data scientist, want to **delete all** the work created in this years of hard work.

DEMO2



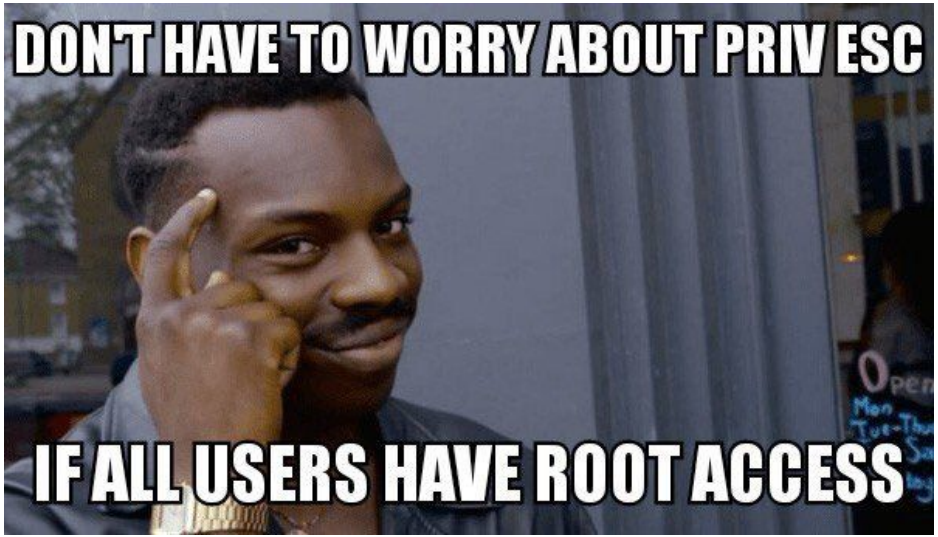
- Privilege escalation are not only meant to reach Administrator permissions
- Defenders needs to protect the company **crown jewels**
- **Backup all the things** (BCP and DR)

The Problem.

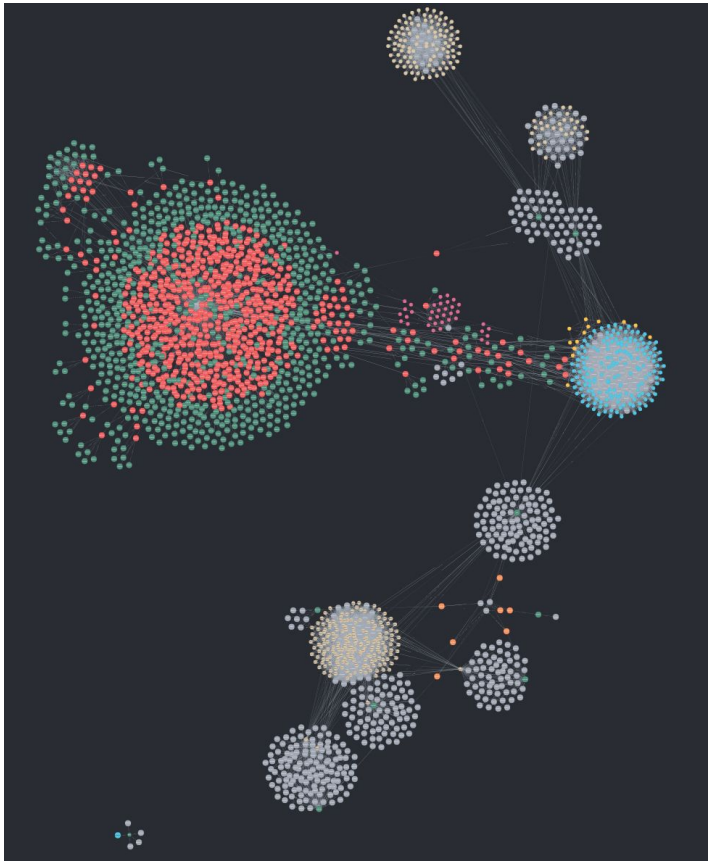
A real problem.

- In general, **defending against these attacks** is (*in theory*) relatively simple:
  - *apply the **least privilege** principle*
  - *use the **permission boundaries***
  - *use the AWS **Access Analyzer***
  - *do not use AWS **managed policies***
  - *create specific **resource policies***
  - ***train** the users*
  - ***harden** the exposed services*

- The complication comes in when trying to defend against these kinds of attacks on **our own environment** that may change quickly, and have a variety of services, roles, resources, etc.



- In **large** and **complex** cloud **ecosystems** reviewing all permissions can be quite **difficult**
- **84% of organisations*** have **no automation**
- **lack of knowledge and expertise**** was consistently identified as:
  - primary barrier to cloud security (59%)
  - primary cause of misconfigurations (62%)
  - a barrier to proactively preventing or fixing misconfigurations (59%)
  - the primary barrier to implementing auto-remediation (56%)



** report: Technology and Cloud Security Maturity, 2022 | Cloud Security Alliance*
*** The State of Cloud Security Risk, Compliance, and Misconfigurations, 2022 | Cloud Security Alliance*

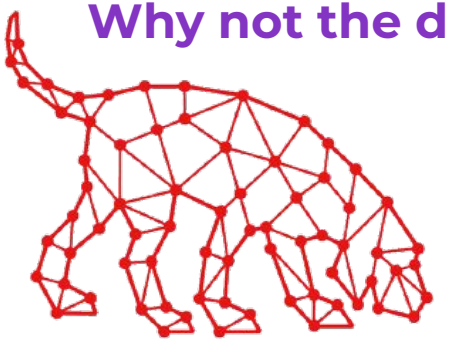
- In general, we need to have a **global overview** of the ecosystem. When **attacking** or **defending** an environment the **knowledge of all aspects** is a key point for a successful operation

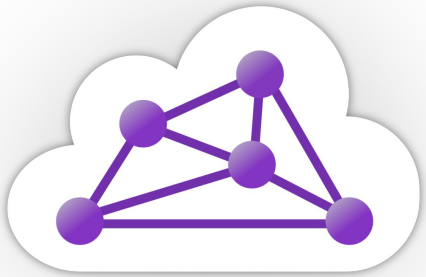
- Attackers think in graphs

- In general, we need to have a **global overview** of the ecosystem. When **attacking** or **defending** an environment the **knowledge of all aspects** is a key point for a successful operation

- Attackers think in graphs

Why not the defenders too?



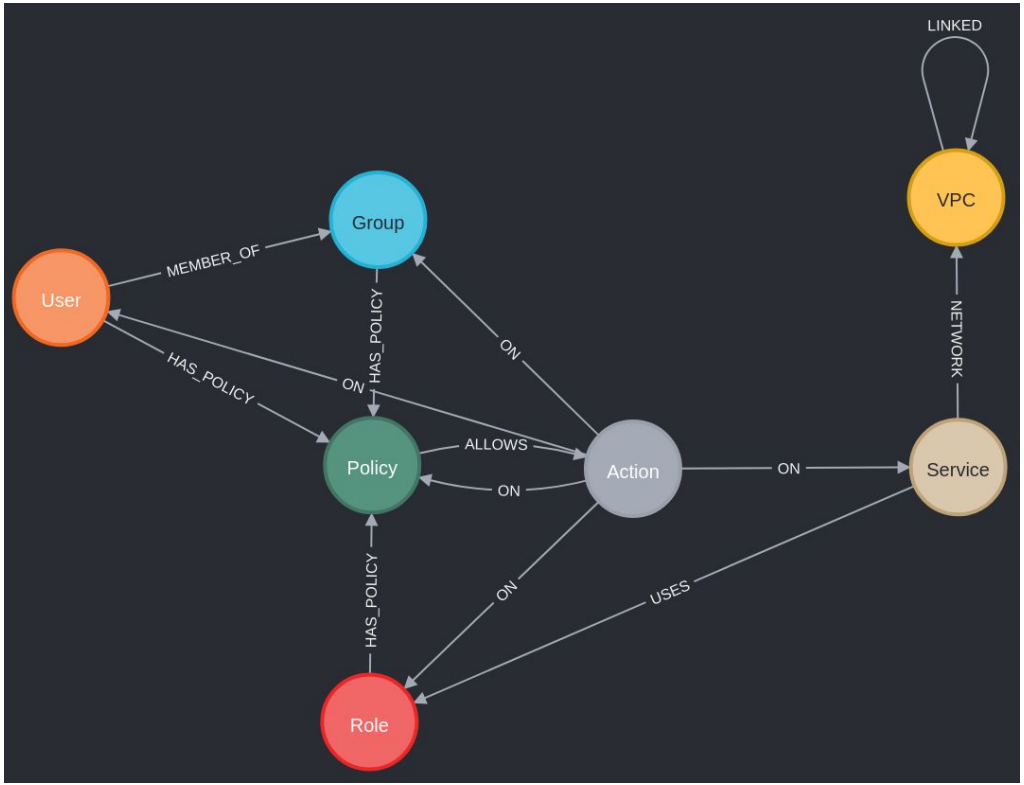


nuvola.

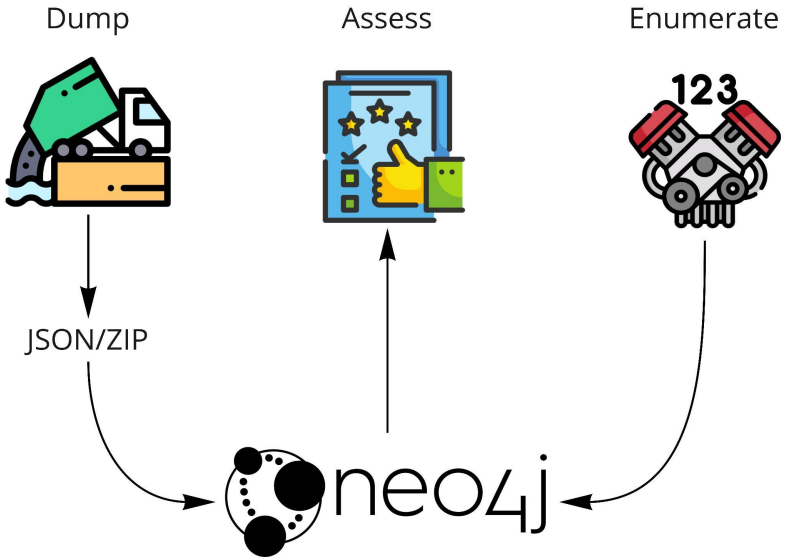
Helping to solve the problem.

- **nuvola** is an open source tool, by **Prima Assicurazioni**, to perform automatic and manual security analysis on **AWS environments configurations and services** using predefined, extensible and custom **rules** created using a simple Yaml syntax.
- The general idea behind this project is to create an abstracted **digital twin** of a cloud platform using **graphs**.
- github.com/primait/nuvola

- The modelling of AWS resources and services can be simplified using **nodes** and **edges**.



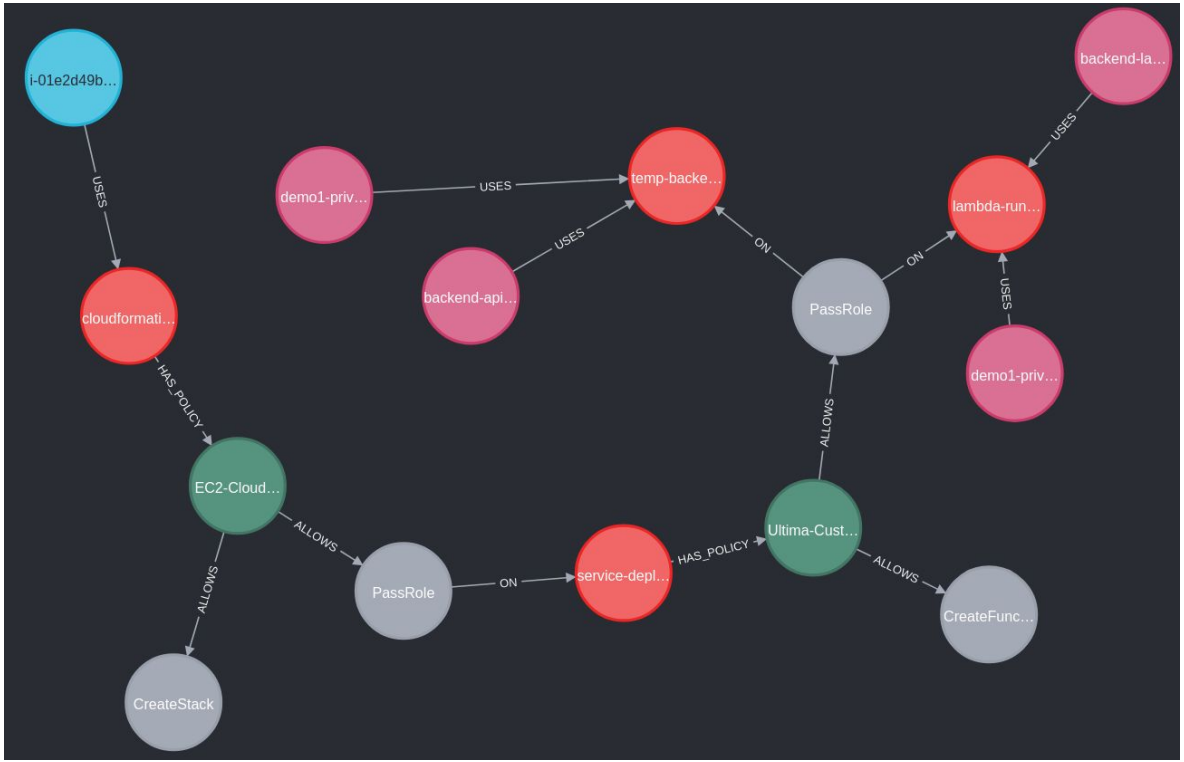
- nuvola is created with **three major** subset of **features**:
  - **Dump**
  - **Assess**
  - **Enumerate** (*TBD*)



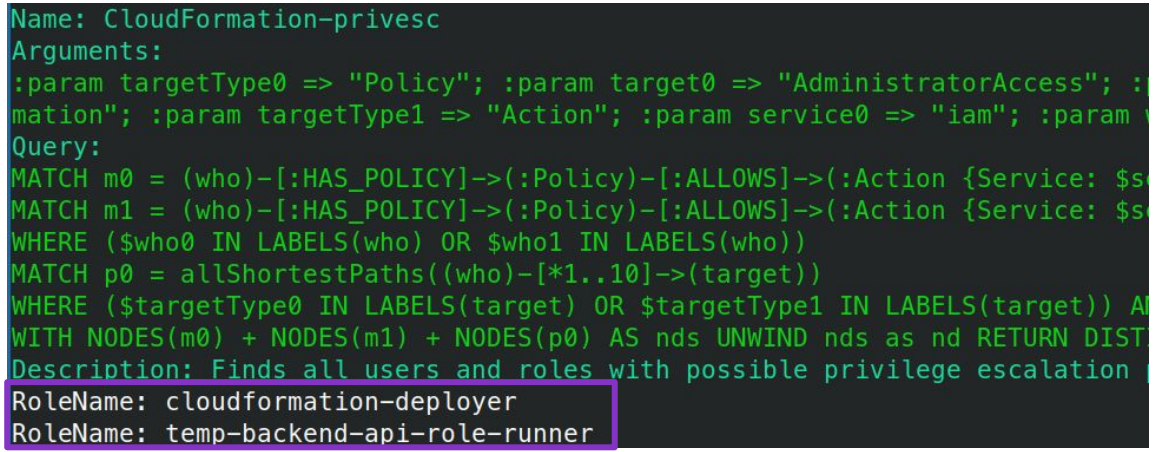
```yaml
name: ec2-IMDS
enabled: true
description: "Finds all EC2 with IMDSv1 enabled"
services:
  - ec2
properties:
  - MetadataOptions:
     - HttpTokens: "optional"
return:
  - InstanceId
  - Tag
  - IamInstanceProfile
```

```
Name: ec2-IMDS
Arguments:
:param name0 => "Ec2"; :param key0 => "MetadataOptions_HttpTokens"; :param value0 => "optional";
Query:
MATCH (s:Service)
WHERE $name0 IN LABELS(s)
WITH s
MATCH (s)
WHERE any(prop in keys(s) where toLower(prop) STARTS WITH toLower($key0) AND s[prop] = $value0)
RETURN s
Description: Finds all EC2 with IMDSv1 enabled
Tags_Value_1: demo1
Tags_Key_0: Name
Tags_Value_0: demo1-ec2
IamInstanceProfile_Arn: arn:aws:iam::573663372258:instance-profile/ec2-instance-role
Tags_Key_1: Stack
IamInstanceProfile_Id: AIPAYLEIAMPRAL56Q5IQT
InstanceId: i-01e2d49b81bb383ea
```

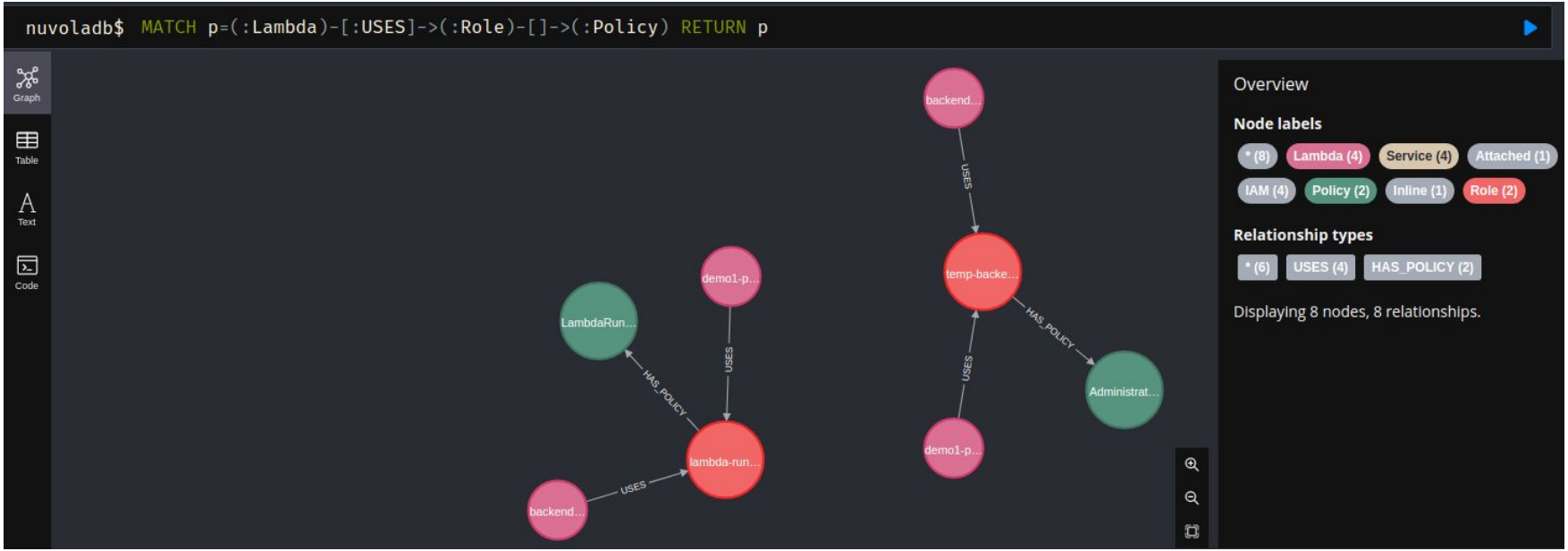
- **Overview of the privilege escalation *demo1* on nuvola**
  - from the EC2 (in blue) to the lambda *demo1-priv-admin* (in pink at the center)



```yaml
name: CloudFormation-privesc
enabled: true
description: "Finds all users and roles with possible privilege escalation permissions
using a CloudFormation stack"
find:
  who:
     - User
     - Role
  with:
     - iam:PassRole
     - cloudformation:CreateStack
  target:
     - policy: AdministratorAccess
     - action: CreateRole
return:
  - RoleName
  - UserName
```



- Using **Neo4j Browser** it's possible to use Cypher queries to **explore the graph** or manually expanding the nodes and relationships



- Using **Neo4j Browser** it's possible to use Cypher queries to **explore the nodes properties**

```
nuvoladb$ MATCH (l:Lambda) RETURN properties(l).FunctionName as FunctionName, properties(l).Runtime as Runtime
```

Table | Text | Code

	FunctionName	Runtime
1	"demo1-privesc-admin-Demo1LambdaFun-YWAxuG1Hghxh"	"python3.9"
2	"backend-api-temp"	"python3.9"
3	"demo1-privesc-Demo1LambdaFun-kgdEp5Rv2QxG"	"python3.9"
4	"backend-lambda-api"	"go1.x"

Conclusions.

- High level **overview** of the infrastructure
- Thinking in **graphs**
- **Community Call For Actions**: improving and expanding the **features** of **nuvola**
  - Pull requests
  - Issues
  - Tests
  - Reviews

Thanks to:

- RomHack incredible staff
- Prima Assicurazioni
- My colleagues
- Fabio
- Santa
- Pietro
- Lisa
- My parents
- You

Thank you.

ROMHACK CAMP

prima

https://prima.jobs

Q&A.

nuvola

github.com/primait/nuvola

Contacts

notdodo

*_notdodo_*