AWS CLOUDFORMATION – AWS IAM: COME DELEGARE IL DEPLOY IN MODO SICURO

AWS CloudFormation | AWS Identity and Access Management (IAM) | DevSecOps | Infrastructure as Code (IaC)

beSharp | 30 Aprile 2020

Giorno dopo giorno sempre più aziende si affacciano con interesse al mondo del cloud computing, molte delle quali si interrogano su come effettivamente possa aiutare il proprio business e su quali problemi di sicurezza si possono introdurre. Una delle maggiori preoccupazioni dei più tecnici è come garantire il maggior grado di libertà possibile offerto dall'agilità che tali tecnologie mettono a disposizione senza introdurre però falle nella sicurezza.

Tra gli argomenti più caldi troviamo sicuramente il concetto di Infrastructure as Code (IaC) che di fatto permette di gestire le proprie risorse in cloud tramite sistemi di templating, aprendo quindi il vaso di Pandora di astrazione, ripetibilità e riproducibilità dell'infrastruttura stessa ma fornendo anche strumenti a supporto di attività di Disaster Recovery.

Cos'è AWS CloudFormation?

In questo articolo andremo a porre la lente di ingrandimento sul servizio di AWS chiamato AWS CloudFormation (che per brevità chiameremo... CloudFormation).

CloudFormation permette di creare, aggiornare e gestire le risorse presenti all'interno del proprio AWS Cloud Environment tramite l'utilizzo di template scritti in JSON o YAML in cui è possibile descrivere risorsa per risorsa la propria infrastruttura.

Solo da questa breve descrizione, una serie di domande è immediata: come posso gestire chi deploya cosa? Ma soprattutto, se posso gestire tutte le risorse, posso gestire anche i permessi? Se posso gestire i permessi, come posso delegare l'utilizzo di CloudFormation prevenendo fenomeni di privilege escalation?

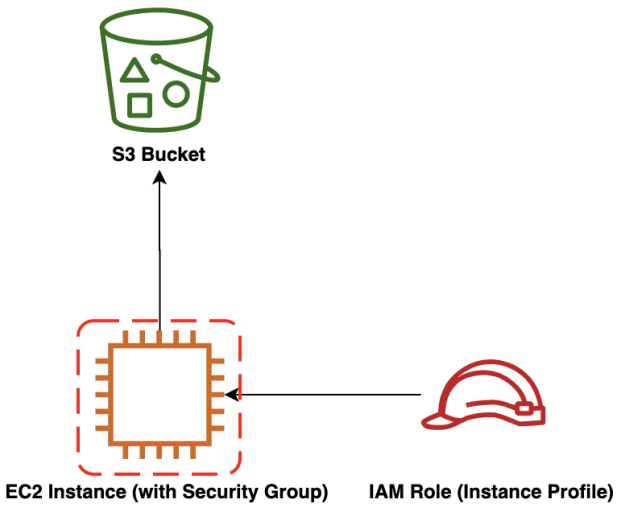
Per rispondere a queste domande occorre prima introdurre un altro servizio: AWS Identity Access Management.

Cos'è AWS IAM?

AWS Identity and Access Management (IAM) è il servizio di AWS che permette di gestire la totalità dei permessi all'interno del proprio AWS Cloud Environment. Per farla breve, affinchè vada a buon fine una qualsiasi azione (che sia su Console, da CLI o tramite SDK) è necessario che sia esplicitamente scritta all'interno di una policy collegata alla propria "utenza" (che sia un effettivo AWS IAM User piuttosto che un AWS IAM Role cambia poco in questa declinazione. Discorso differente se si parla di validità delle credenziali in uso, dove l'utilizzo del secondo ne permette anche un'automatica rotazione).

Use Case

Ora che conosciamo quali sono gli strumenti a nostra disposizione, caliamoci in una situazione verosimile: abbiamo creato un template per CloudFormation che genera un'infrastruttura composta da un'istanza EC2 pre-configurata (attraverso un Amazon Machine Image) che deve poter accedere ad un bucket S3. Essendo una infrastruttura semplice, vogliamo rendere indipendenti gli sviluppatori in modo tale che possano deployare questo template senza disturbare il team di Security ma al contempo garantendo la totale sicurezza.



I più lungimiranti sapranno certamente che per deployare tale infrastruttura è necessaria la creazione di uno IAM Role, ma dobbiamo escludere la possibilità che lo sviluppatore possa crearsi un ruolo con permessi da amministratore da assumere per evitare i soliti "noiosi" messaggi di "Access Denied" e "Forbidden".

Senza Permission Boundaries

Il template che andremo a deployare è il seguente:

```
---
AWSTemplateFormatVersion: '2010-09-09'
Description: |
 CloudFormation and IAM Permission Boundaries Demo


##############################################################################
#                                Metadata                                    #
```

```
##############################################################################
Metadata:

 AWS::CloudFormation::Interface:
   ParameterGroups:
     - Label: {default: 'Required parameters'}
       Parameters:
         - VpcId
         - AmiId
         - KeyName
         - SubnetId
     - Label: {default: 'Optional parameters'}
       Parameters:
         - NameSpace
         - ProjectName
         - Environment


##############################################################################
#                                  Parameters                                #
##############################################################################
Parameters:

 NameSpace:
   Type: String
   Default: 'besharp'

 ProjectName:
   Type: String
   Default: 'permission-boundaries-demo'

 Environment:
   Type: String
   Default: 'dev'
  VpcId:
   Type: AWS::EC2::VPC::Id

 SubnetId:
   Type: AWS::EC2::Subnet::Id

 AmiId:
   Type: AWS::EC2::Image::Id

 KeyName:
   Type: AWS::EC2::KeyPair::KeyName


##############################################################################
#                                  Conditions                                #
##############################################################################
Conditions: {}


##############################################################################
#                                   Mappings                                 #
##############################################################################
Mappings: {}


##############################################################################
```

```yaml
#                                   Resources                                  #
################################################################################
Resources:

 ################################### S3 ###################################
 S3Bucket:
   Type: AWS::S3::Bucket
   Properties:
     BucketName: !Sub 'com.${NameSpace}.${ProjectName}'
     Tags:
       - Key: Name
         Value: !Sub '${NameSpace}-${ProjectName}'
       - Key: Owner
         Value: 'name.surname@besharp.it'

 ################################### EC2 ###################################
 EC2Instance:
   Type: AWS::EC2::Instance
   Properties:
     IamInstanceProfile: !Ref IAMInstanceProfile
     ImageId: !Ref AmiId
     InstanceType: t3a.micro
     KeyName: !Ref KeyName
     NetworkInterfaces:
       - AssociatePublicIpAddress: 'true'
         DeviceIndex: '0'
         GroupSet:
           - !Ref EC2SecurityGroup
         SubnetId: !Ref SubnetId
     Tags:
       - Key: Name
         Value: !Sub '${NameSpace}-${ProjectName}'
       - Key: Owner
         Value: 'name.surname@besharp.it'

 EC2SecurityGroup:
   Type: AWS::EC2::SecurityGroup
   Properties:
     GroupName: !Sub '${NameSpace}-${ProjectName}-ec2'
     GroupDescription: !Sub 'Security Group for ${NameSpace}-${ProjectName}-ec2'
     VpcId: !Ref VpcId
     Tags:
       - Key: Name
         Value: !Sub '${NameSpace}-${ProjectName}-ec2'
       - Key: Owner
         Value: 'name.surname@besharp.it'

 ################################### IAM ###################################
 IAMInstanceProfile:
   Type: AWS::IAM::InstanceProfile
   Properties:
     Roles:
       - !Ref IAMRole
     InstanceProfileName: !Sub '${NameSpace}-${ProjectName}'

 IAMRole:
   Type: AWS::IAM::Role
   Properties:
     RoleName: !Sub '${NameSpace}-${ProjectName}'
     AssumeRolePolicyDocument:
```

```
      Version: '2012-10-17'
      Statement:
        - Effect: Allow
          Principal:
            Service: ec2.amazonaws.com
          Action: sts:AssumeRole
    Path: '/'
    Policies:
      - PolicyName: 'EC2Access'
        PolicyDocument:
          Version: '2012-10-17'
          Statement:
            - Effect: 'Allow'
              Action:
                - 's3:GetObject'
              Resource: !Sub '${S3Bucket.Arn}/*'
    Tags:
      - Key: Name
        Value: !Sub '${NameSpace}-${ProjectName}'
      - Key: Owner
        Value: 'name.surname@besharp.it'


##############################################################################
#                                  Outputs                                   #
##############################################################################
Outputs:

 StackName:
   Description: 'Stack name.'
   Value: !Sub '${AWS::StackName}'
```

Al suo interno sono presenti le informazioni per creare:

Bucket S3

Istanza EC2 + Security Group

Ruolo IAM + Instance Profile

Allo sviluppatore è stata attaccata la seguente policy per poter deployare questo template:

```
{
   "Version": "2012-10-17",
   "Statement": [
       {
           "Sid": "CloudFormationReadAccess",
           "Effect": "Allow",
           "Action": [
               "cloudformation:DescribeStacks",
               "cloudformation:ListChangeSets",
               "cloudformation:ListExports",
               "cloudformation:ListImports",
               "cloudformation:ListStacks"
           ],
           "Resource": "*"
       },
       {
           "Sid": "CloudFormationWriteAccess",
```

```
            "Effect": "Allow",
            "Action": [
                "cloudformation:CreateStack",
                "cloudformation:DeleteStack",
                "cloudformation:TagResource",
                "cloudformation:UpdateStack",
                "cloudformation:ValidateTemplate"
            ],
            "Resource": "*"
        },
        {
            "Sid": "EC2ReadAccess",
            "Effect": "Allow",
            "Action": [
                "ec2:DescribeImages",
                "ec2:DescribeInstances",
                "ec2:DescribeKeyPairs",
                "ec2:DescribeSecurityGroupReferences",
                "ec2:DescribeSecurityGroups",
                "ec2:DescribeSubnets",
                "ec2:DescribeVpcs"
            ],
            "Resource": "*"
        },
        {
            "Sid": "EC2WriteAccess",
            "Effect": "Allow",
            "Action": [
                "ec2:AssociateIamInstanceProfile",
                "ec2:AuthorizeSecurityGroupEgress",
                "ec2:AuthorizeSecurityGroupIngress",
                "ec2:CreateSecurityGroup",
                "ec2:CreateTags",
                "ec2:DeleteSecurityGroup",
                "ec2:DeleteTags",
                "ec2:RevokeSecurityGroupEgress",
                "ec2:RevokeSecurityGroupIngress",
                "ec2:RunInstances",
                "ec2:StartInstances",
                "ec2:StopInstances",
                "ec2:TerminateInstances"
            ],
            "Resource": "*"
        },
        {
            "Sid": "IAMReadAccess",
            "Effect": "Allow",
            "Action": [
                "iam:GetInstanceProfile",
                "iam:GetRole",
                "iam:GetRolePolicy"
            ],
            "Resource": "*"
        },
        {
            "Sid": "IAMWriteAccess",
            "Effect": "Allow",
            "Action": [
                "iam:AddRoleToInstanceProfile",
                "iam:CreateInstanceProfile",
```

```
                "iam:CreateRole",
                "iam:DeleteInstanceProfile",
                "iam:DeleteRole",
                "iam:DeleteRolePolicy",
                "iam:PassRole",
                "iam:PutRolePolicy",
                "iam:RemoveRoleFromInstanceProfile",
                "iam:TagRole",
                "iam:UntagRole"
            ],
            "Resource": "*"
        },
        {
            "Sid": "S3WriteAccess",
            "Effect": "Allow",
            "Action": [
                "s3:CreateBucket",
                "s3:DeleteBucket",
                "s3:PutBucketTagging"
            ],
            "Resource": "*"
        }
    ]
}
```

Connettendoci all'istanza EC2 creata, possiamo verificare che effettivamente possa scaricare oggetti dal bucket S3 appena creato:

```
ubuntu@ec2-demo:~$ aws s3 cp s3://com.besharp.permission-boundaries-demo/if-you-download-me-you-are-fine .

download: s3://com.besharp.permission-boundaries-demo/if-you-download-me-you-are-fine
 to ./if-you-download-me-you-are-fine
ubuntu@ec2-demo:~$ cat if-you-download-me-you-are-fine
test-ok
```

E che non possa, ad esempio, creare altri bucket:

```
ubuntu@ec2-demo:~$ aws s3api create-bucket --bucket can-i-create-it

An error occurred (AccessDenied) when calling the CreateBucket operation: Access Denied
```

A questo punto, lo sviluppatore potrebbe (a prescindere dalla buona o cattiva fede) voler modificare i permessi del ruolo associato alla macchina per poter effettuare operazioni che di norma non è autorizzato ad eseguire (o addirittura crearsi un ruolo di amministrazione da assumere). La risorsa relativa allo IAM Role può essere modificata così:

```
 IAMRole:
   Type: AWS::IAM::Role
   Properties:
     RoleName: !Sub '${NameSpace}-${ProjectName}'
```

```
      AssumeRolePolicyDocument:
        Version: '2012-10-17'
        Statement:
          - Effect: Allow
            Principal:
              Service: ec2.amazonaws.com
            Action: sts:AssumeRole
      Path: '/'
      Policies:
        - PolicyName: 'AdministratorAccess'
          PolicyDocument:
            Version: '2012-10-17'
            Statement:
              - Effect: 'Allow'
                Action:
                  - '*'
                Resource: '*'
      Tags:
        - Key: Name
          Value: !Sub '${NameSpace}-${ProjectName}'
```

In tal modo, ad esempio, l'istanza può scaricare oggetti dal bucket, ma anche crearne di altri (senza contare che, con la policy modificata rozzamente come da esempio, di fatto, ha permessi di amministratore all'interno dell'account):

```
ubuntu@ec2-demo:~$ aws s3 cp s3://com.besharp.permission-boundaries-demo/if-you-download-me-you-are-fine .

download: s3://com.besharp.permission-boundaries-demo/if-you-download-me-you-are-fine
 to ./if-you-download-me-you-are-fine
ubuntu@ec2-demo:~$ cat if-you-download-me-you-are-fine
test-ok
ubuntu@ec2-demo:~$ aws s3api create-bucket --bucket can-i-create-it
{
    "Location": "/can-i-create-it"
}
```

Chiaramente la preoccupazione principale non è data solamente dal fatto che lo sviluppatore possa acquisire permessi non previsti, ma anche dall'effettivo aumento della superficie attaccabile da malintenzionati. Non è nuovo infatti il concetto di privilege escalation, nel quale una persona anche esterna all'azienda va a sfruttare dei permessi inutilmente ampi per predisporre il terreno fertile ad un attacco organizzato.

Con Permission Boundaries

I permission boundaries non sono altro che delle policy IAM da attaccare un'entità IAM per circoscriverne i permessi. Infatti, i permessi effettivi saranno l'intersezione di quelli garantiti dalle normali IAM policy con quelli garantiti dalla policy di permission boundary. Di per sé non vanno a risolvere particolari problemi, ma di fatto è possibile obbligare lo sviluppatore a dover agganciare il permission boundary specificato qualora voglia creare un ruolo.

La parte relativa a IAM della policy dello sviluppatore diventa quindi:

```
{
            "Sid": "IAMPermissionBoundaryWriteAccess",
            "Effect": "Allow",
            "Action": [
                "iam:CreateRole",
                "iam:PutRolePolicy",
                "iam:UpdateRole",
                "iam:UpdateRoleDescription"
            ],
            "Resource": "arn:aws:iam::111122223333:role/dev-namespace/*",
            "Condition": {
                "StringEquals": {
                    "iam:PermissionsBoundary": "arn:aws:iam::111122223333:policy/besharp-permission-boundary-demo"
                }
            }
        },
        {
            "Sid": "IAMPassRoleAccess",
            "Effect": "Allow",
            "Action": [
                "iam:PassRole"
            ],
            "Resource": "arn:aws:iam::111122223333:role/dev-namespace/*"
        },
        {
            "Sid": "IAMWriteAccess",
            "Effect": "Allow",
            "Action": [
                "iam:AddRoleToInstanceProfile",
                "iam:CreateInstanceProfile",
                "iam:DeleteInstanceProfile",
                "iam:DeleteRole",
                "iam:DeleteRolePolicy",
                "iam:RemoveRoleFromInstanceProfile",
                "iam:TagRole",
                "iam:UntagRole"
            ],
            "Resource": "*"
        },
```

Se al momento della creazione del ruolo da parte di CloudFormation non è presente il permission boundary, la creazione fallirà.

Per agganciare il permission boundary allo IAM role dell'istanza, la risorsa va modificata come segue:

```
IAMRole:
   Type: AWS::IAM::Role
   Properties:
     RoleName: !Sub '${NameSpace}-${ProjectName}'
     PermissionsBoundary: !Ref PermissionBoundaryArn
     AssumeRolePolicyDocument:
       Version: '2012-10-17'
```

```
      Statement:
        - Effect: Allow
          Principal:
            Service: ec2.amazonaws.com
          Action: sts:AssumeRole
    Path: '/dev-namespace/'
    Policies:
      - PolicyName: 'AdministratorAccess'
        PolicyDocument:
          Version: '2012-10-17'
          Statement:
            Effect: 'Allow'
            Action:
               - '*'
            Resource: '*'
    Tags:
      - Key: Name
        Value: !Sub '${NameSpace}-${ProjectName}'
      - Key: Owner
        Value: 'name.surname@besharp.it'
```

Nota: per rendere il tutto più sicuro e limitare l'azione particolarmente sensibile di iam:PassRole è stato aggiunto anche il concetto di Path, che essenzialmente va ad identificare un namespace per gli sviluppatori all'interno di IAM

Agganciando il permission boundary, la creazione avrà successo e i permessi dell'istanza (e di qualunque altro ruolo lo sviluppatore voglia creare) saranno limitati.

```
ubuntu@ec2-demo:~$ aws s3 cp s3://com.besharp.permission-boundaries-demo/if-you-download-me-you-are-fine .

download: s3://com.besharp.permission-boundaries-demo/if-you-download-me-you-are-fine
 to ./if-you-download-me-you-are-fine
ubuntu@ec2-demo:~$ cat if-you-download-me-you-are-fine
test-ok
ubuntu@ec2-demo:~$ aws s3api create-bucket --bucket can-i-create-it

An error occurred (AccessDenied) when calling the CreateBucket operation: Access Denied
```

Nella configurazione presentata è chiaro come il permission boundary diventi la risorsa sotto controllo del team di sicurezza, il quale può concordare l'insieme massimo di permessi attribuibili ad una determinata risorsa. Una volta individuati tali permessi, AWS IAM garantirà che l'entità IAM associata ad una risorsa non possa effettuare azioni non previste.

Conclusioni

Abbiamo visto come, grazie ad una semplice modifica dei permessi dello sviluppatore, si possa limitare forzatamente i permessi attribuibili ad una entità IAM. Tale approccio spesso identifica lo sviluppatore come **delegated admin**. In tal modo è possibile aumentare l'indipendenza degli sviluppatori, riducendo il numero di task giornalieri che le poche persone con ampi permessi

all’interno dell’AWS Cloud Environment devono evadere senza andare però a introdurre compromessi sulla sicurezza.

Soddisfatti? 🙂 se avete curiosità o desiderate fare due chiacchiere in merito a questo o a soluzioni simili, non esitate a scriverci nei commenti o a contattarci!

beSharp

Dal 2011 beSharp guida le aziende italiane sul Cloud. Dalla piccola impresa alla grande multinazionale, dal manifatturiero al terziario avanzato, aiutiamo le realtà più all'avanguardia a realizzare progetti innovativi in campo IT.

Get in touch

beSharp.it
proud2becloud@besharp.it