SAC303

How to Become an IAM Policy Ninja in 60 Minutes or Less

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": ["s3:Get*", "s3:List*"],
      "Resource": "*"
    }
  ]
}
```

Jeff Wierer, Senior Manager – AWS IAM

November 30, 2016

What to Expect from the Session

- Know more about securing your AWS resources
- Deeper understanding of AWS IAM permissions
- Tips and tricks
- Debugging, testing, and other policy foo
- A lively session via demos

Amazon S3

AWS IAM

Amazon EC2

- Goal: Limit a user from starting an instance unless the instance is `t1.*`, `t2.*`, `m3.*`
- **Limit Amazon EC2 instance types**

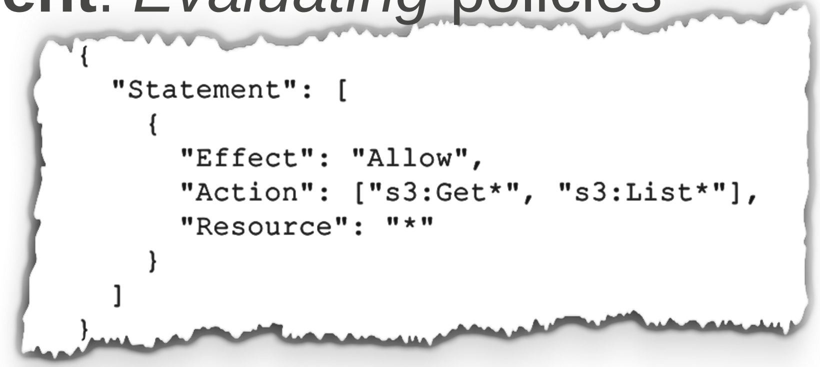
Let's try to:

Demo

  - Create a managed policy that attempts to limit starting an EC2 instance except for these instance types.
  - Attach that policy to an IA

The policy language

- Provides authorization
- Two facets:
  - **Specification**: *Defining* access policies
  - **Enforcement**: *Evaluating* policies



```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": ["s3:Get*", "s3:List*"],
      "Resource": "*"
    }
  ]
}
```

Principal – Examples

- An entity that is allowed or denied access to a resource
- Indicated by an Amazon Resource Name (ARN)
- With IAM policies, the principal element is implicit (i.e., the user, group, or role attached)

```
<!-- Everyone (anonymous users) -->
"Principal":"AWS":"*.*"

<!-- Specific account or accounts -->
"Principal":{"AWS":"arn:aws:iam::123456789012:root" }
"Principal":{"AWS":"123456789012"}

<!-- Individual IAM user -->
"Principal":"AWS":"arn:aws:iam::123456789012:user/username"

<!-- Federated user (using web identity federation) -->
"Principal":{"Federated":"www.amazon.com"}
"Principal":{"Federated":"graph.facebook.com"}
"Principal":{"Federated":"accounts.google.com"}

<!-- Specific role -->
"Principal":{"AWS":"arn:aws:iam::123456789012:role/rolename"}

<!-- Specific service -->
 "Principal":{"Service":"ec2.amazonaws.com"}
```

Replace with your account number

Action – Examples

- Describes the type of access that should be allowed or denied
- You can find these in the docs or use the policy editor to get a drop-down list
- Statements must include either an `Action` or `NotAction` element

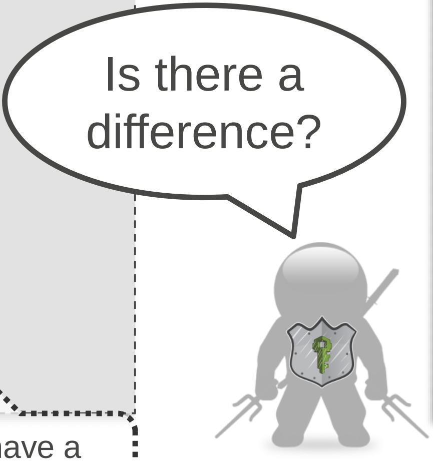
```
<!-- EC2 action -->
"Action":"ec2:StartInstances"

<!-- IAM action -->
"Action":"iam:ChangePassword"

<!– Amazon S3 action -->
"Action":"s3:GetObject"

<!-- Specify multiple values for the Action element-->
"Action":["sqs:SendMessage","sqs:ReceiveMessage"]

<-- Wildcards (* or ?) in the action name. Below covers create/delete/list/update-->
"Action":"iam:*AccessKey*"
```

Understanding `NotAction`

- Lets you specify an exception to a list of actions
- Could result in shorter policies than using `Action` and denying many actions
- Example: Let's say you want to allow everything but IAM APIs

```
{
  "Version": "2012-10-17",
  "Statement": [ {
      "Effect": "Allow",
      "NotAction": "iam:*",
      "Resource": "*"
    }
  ]
}
```

This is not a **`Deny`**. A user could still have a separate policy that grants **`IAM:*`**



```
{
  "Version": "2012-10-17",
  "Statement": [{
      "Effect": "Allow",
      "Action": "*",
      "Resource": "*"
    },
    {
      "Effect": "Deny",
      "Action": "iam:*",
      "Resource": "*"
    }
  ]
}
```

If you want to prevent the user from ever being able to call IAM APIs, use an explicit **`Deny`**.

Resource – Examples

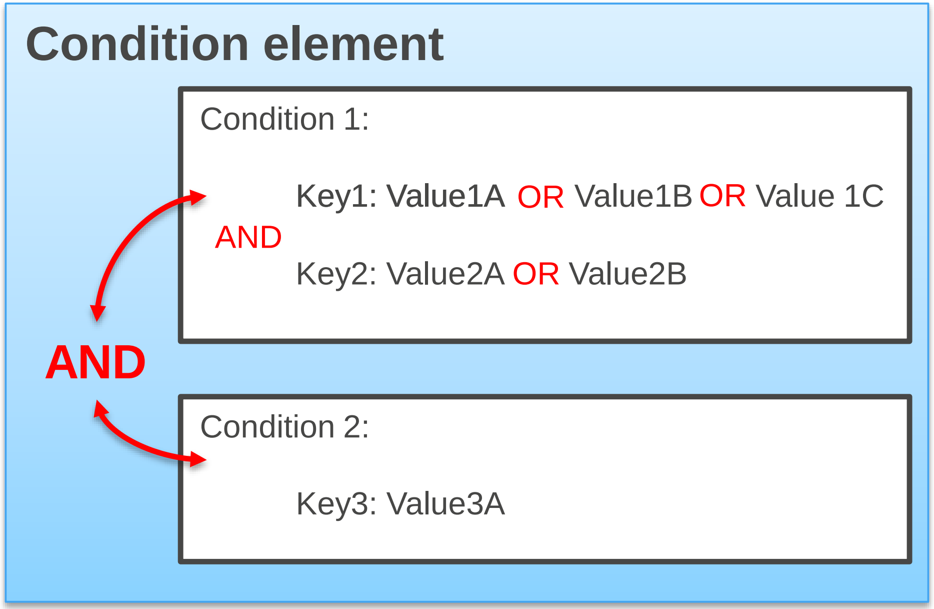
- The object or objects that are being requested
- Statements must include either a `Resource` or a `NotResource` element

```
<-- S3 Bucket -->
"Resource":"arn:aws:s3:::my_corporate_bucket/*"

<-- Amazon SQS queue-->
"Resource":"arn:aws:sqs:us-west-2:123456789012:queue1"

<-- Multiple Amazon DynamoDB tables -->
"Resource":["arn:aws:dynamodb:us-west-2:123456789012:table/books_table",
         "arn:aws:dynamodb:us-west-2:123456789012:table/magazines_table"]

<-- All EC2 instances for an account in a region -->
 "Resource": "arn:aws:ec2:us-east-1:123456789012:instance/*"
```

Conditions

- Optional criteria that must evaluate to true for the policy to evaluate as true

  (ex: restrict to an IP address range)
- Can contain multiple conditions
- Condition keys can contain multiple values
- If a single condition includes multiple values for one key, the condition is evaluated using logical OR
- Multiple conditions (or multiple keys in a single condition): the conditions are evaluated using logical AND



Condition example

AND

```
"Condition" :  {
  "DateGreaterThan" : {"aws:CurrentTime" : "2016-11-30T11:00:00Z"},
  "DateLessThan": {"aws:CurrentTime" : "2016-11-30T15:00:00Z"},
  "IpAddress" : {"aws:SourceIp" : ["192.0.2.0/24", "203.0.113.0/24"]}
}
```

OR

- Allows a user to access a resource under the following conditions:
  - The time is after 11:00 A.M. on 11/30/2016 AND
  - The time is before 3:00 P.M. on 11/30/2016 AND
  - The request comes from an IP address in the 192.0.2.0 /24 OR 203.0.113.0 /24 range

All of these conditions <u>must be met</u> in order for the statement to evaluate to TRUE.

Policy variables

${aws:userid}

- Predefined variables based on service request context
  - Existing keys (`aws:SourceIP`, `aws:CurrentTime`, etc.)
  - Principal-specific keys (`aws:username, aws:userid, aws:principaltype`)
  - Provider-specific keys (`graph.facebook.com:id, www.amazon.com:user_id`)
  - SAML keys (`saml:aud, saml:iss`)
  - See documentation for service-specific variables
- Benefits
  - Simplifies policy management
  - Reduces the need for hard-coded, user-specific policies
- Use cases we'll look at
  - Set up user access to "home folder" in S3
  - Limit access to specific EC2 resources

The anatomy of a policy with variables

```
{
  "Version": "2012-10-17",
  "Statement": [{
     "Effect": "Allow",
     "Action": ["s3:ListBucket"],
     "Resource": ["arn:aws:s3:::myBucket"],
     "Condition":
       {"StringLike":
          {"s3:prefix":["home/${aws:username}/*"]}
        }
    },
    {
      "Effect":"Allow",
      "Action":["s3:*"],
      "Resource": ["arn:aws:s3:::myBucket/home/${aws:username}",
                   "arn:aws:s3:::myBucket/home/${aws:username}/*"]
    }
  ]
}
```

Version is required

Variable in conditions

Variable in resource ARNs

Grants a user access to a home directory in S3 that can be accessed programmatically

Managing your policies

- IAM policies
  - Managed policies
  - Inline polices
- Resource-based policies

IAM policies

- Managed policies (newer way)
  - Can be attached to multiple users, groups, and roles
  - AWS managed policies: Created and maintained by AWS
  - Customer managed policies: Created and maintained by you
    - Up to 5K per policy
    - Up to 5 versions of a policy so you can roll back to a prior version
  - You can attach 10 managed policies per user, group, or role
  - You can limit who can attach which managed policies

- Inline policies (older way)
  - You create and embed directly in a single user, group, or role
  - Variable policy size (2K per user, 5K per group, 10K per role)

Resource-based policies

IAM policies live with:

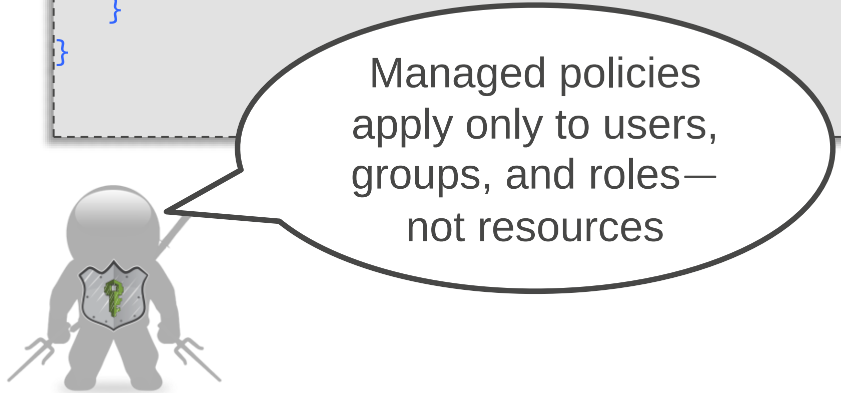
- IAM users
- IAM groups
- IAM roles

Some services allow storing policy with resources:

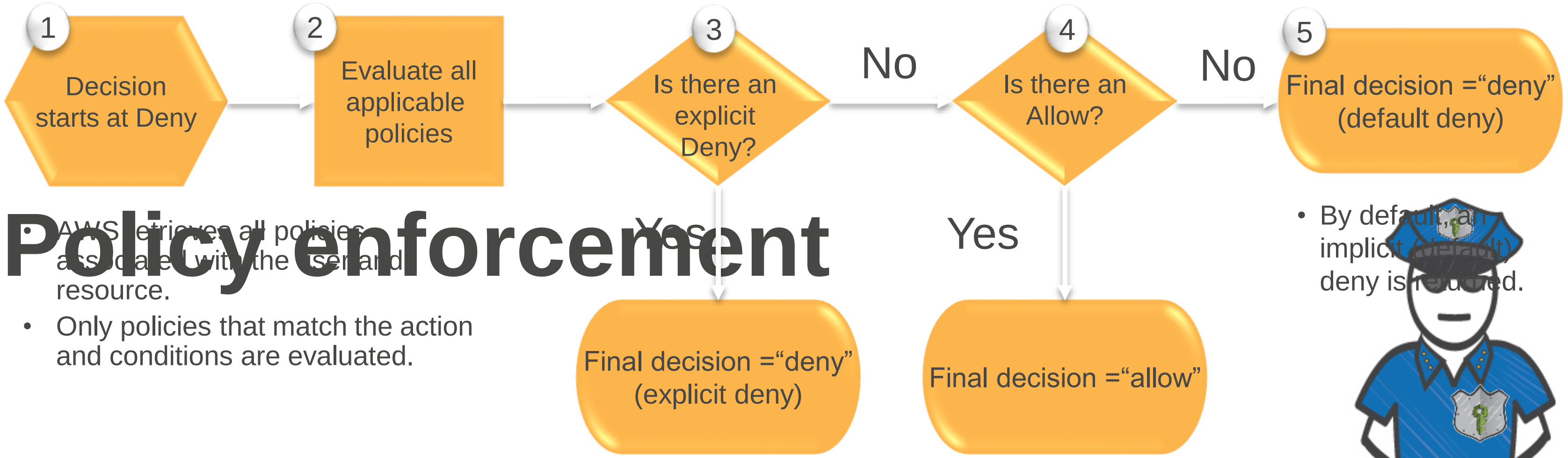
- S3 (bucket policy)
- Amazon Glacier (vault policy)
- Amazon SNS (topic policy)
- Amazon SQS (queue policy)

Principal required here

```
{
 "Statement":
  {
    "Effect": "Allow",
    "Principal": {"AWS": "111122223333"},
    "Action": "sqs:SendMessage",
    "Resource":
       "arn:aws:sqs:us-east-1:444455556666:queue1"
  }
}
```


Managed policies apply only to users, groups, and roles—not resources

Policy enforcement


1
Decision starts at Deny
2
Evaluate all applicable policies
3
Is there an explicit Deny?
No
4
Is there an Allow?
No
5
Final decision =“deny” (default deny)
Yes
Final decision =“deny” (explicit deny)
Yes
Final decision =“allow”

- AWS retrieves all policies associated with the user and resource.
- Only policies that match the action and conditions are evaluated.

- If a policy statement has a deny, it trumps all other policy statements.

- Access is granted if there is an explicit allow and no deny.

- By default, an implicit (default) deny is returned.

Enough already…
Let's look at some examples

S3

IAM

EC2

Creating a home directory using S3

Demo

- Goal: Create a managed policy that:
  - Limits access to a prefix in an S3 bucket
  - For example, `arn:aws:s3:::my_bucket/home/Bob/*`
- We'll examine how to:
  - Create a managed policy that uses variables.
  - Enable users to list buckets in the S3 console.
  - Limit users' access to specific folders in a bucket.

Giving a user a home directory from the S3 console

```
{
"Version": "2012-10-17",
  "Statement": [
    {"Sid": "AllowGroupToSeeBucketListInTheManagementConsole",
      "Action": ["s3:ListAllMyBuckets", "s3:GetBucketLocation"],
      "Effect": "Allow",
      "Resource": ["arn:aws:s3:::*"]},
    {"Sid": "AllowRootLevelListingOfThisBucketAndHomePrefix",
      "Action": ["s3:ListBucket"],
      "Effect": "Allow",
      "Resource": ["arn:aws:s3:::myBucket"],
      "Condition":{"StringEquals":{"s3:prefix":["","home/"],"s3:delimiter":["/"]}}},
    {"Sid": "AllowListBucketofASpecificUserPrefix",
      "Action": ["s3:ListBucket"],
      "Effect": "Allow",
      "Resource": ["arn:aws:s3:::myBucket"],
      "Condition":{"StringLike":{"s3:prefix":["home/${aws:username}/*"]}}},
    {"Sid":"AllowUserFullAccesstoJustSpecificUserPrefix",
       "Action":["s3:*"],
       "Effect":"Allow",
       "Resource": ["arn:aws:s3:::myBucket/home/${aws:username}",
                    "arn:aws:s3:::myBucket/home/${aws:username}/*"]}
  ]
}
```

- Necessary to access the S3 console.
- Allows listing all objects in a folder and its subfolders.
- Allows modifying objects in the folder and subfolders.

Creating a “limited” IAM administrator

Demo

- Goal:
  - Create a limited administrator who can use the IAM console to manage users, but only using certain policies
- We’ll examine group policies that use variables to:
  - Grant admin full access to Amazon DynamoDB and read/write access to an S3 bucket.
  - Grant admin access to the IAM console to be able to create users and generate access keys.
  - Limit admin to a well-defined set of managed policies.

Create a “limited” IAM administrator

```
{
  "Version": "2012-10-17",
  "Statement": [{
      "Sid": "ManageUsersPermissions",
      "Effect": "Allow",
      "Action": ["iam:ChangePasword", "iam:CreateAccessKey", "iam:CreateLoginProfile",
                 "iam:CreateUser", "iam:DeleteAccessKey", "iam:DeleteLoginProfile",
                 "iam:DeleteUser", "iam:UpdateAccessKey", "iam:ListAttachedUserPolicies",
                 "iam:ListPolicies"],
      "Resource": "*"
    },
    {
      "Sid": "LimitedAttachmentPermissions",
      "Effect": "Allow",
      "Action": ["iam:AttachUserPolicy","iam:DetachUserPolicy"],
      "Resource": "*",
      "Condition": {
        "ArnEquals": {
          "iam:PolicyArn": [
            "arn:aws:iam::123456789012:policy/reInvent_S3_Home_Folder",
            "arn:aws:iam::aws:policy/AmazonDynamoDBFullAccess"
          ]
        }
      }
    }
  ]
}
```

See AWS Security Blog post http://amzn.to/1Hf2XRl

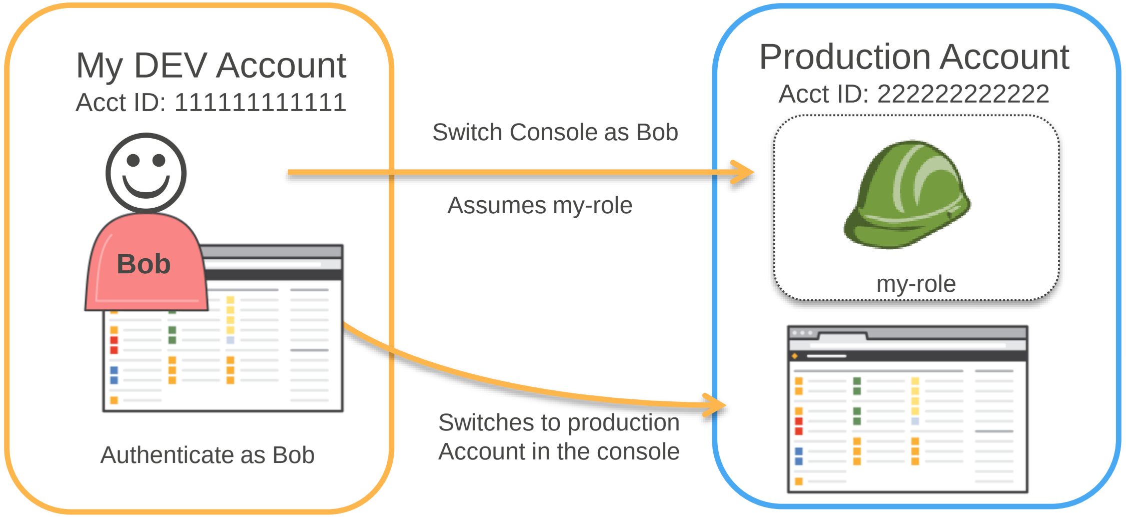
- Allows creating users, managing keys, and setting passwords.
- Limits attaching only these two policies.

Grant a user access to the IAM console

```
{
  "Version": "2012-10-17",
  "Statement": [{
     "Sid": "ViewListOfAllUsers",
     "Action": "iam:ListUsers",
     "Effect": "Allow",
     "Resource": "arn:aws:iam::123456789012:user/*"
    },
    {
     "Sid": "AllowAdmintoAccessUser",
     "Effect": "Allow",
     "Action": ["iam:GetUser","iam:GetLoginProfile",
         "iam:ListGroupsForUser","iam:ListAccessKeys"],
     "Resource": "arn:aws:iam::123456789012:user/${aws:username}"
    }
  ]
}
```

- Underneath the covers, the IAM console calls these APIs to view user settings.
- The user will be able to view details about all users.
- Doesn’t enable adding/removing MFA.

Cross-Account Console Access



Permissions assigned to my-role

```
{ "Statement": [
 {
   "Effect": "Allow",
   "Action": "ACTIONS",
   "Resource": "*"
  }
 ]
}
```

```
{ "Statement": [
  {
   "Effect": "Allow",
   "Action": “sts:AssumeRole",
   "Resource": "arn:aws:iam::222222222222:role/my-role"
  }
 ]
}
```

Policy assigned to Bob granting him permission to assume my-role in the production account

```
{ "Statement": [
  {
   "Effect":"Allow",
   "Principal":{"AWS":"arn:aws:iam::111111111111:root"},
   "Action":"sts:AssumeRole"
  }
 ]
}
```

Policy assigned to my-role defining who (trusted entities) can assume the role

How to Grant Conditional Cross-Account Access

Demo

- Goal: Create a managed policy that:
  - Limits cross-account access to specific users.
  - E.g., IAM users and federated users.
- We’ll examine ho
  - Create a manage   uses IAM/STS ARNs.
  - Enable specific u   n between AWS accounts.
  - Deny switching b   unts to other users.

Grant cross-account conditional IAM user access

```
{
     "Version": "2012-10-17",
     "Statement": [{
          "Effect": "Allow",
          "Action": "sts:AssumeRole",
          "Principal": {
               "AWS": ["arn:aws:iam::790891838339:root"]
               ]
          }
     }]
}
```

This is what is put in by default

Grant cross-account conditional IAM user access

```
{
     "Version": "2012-10-17",
     "Statement": [{
          "Effect": "Allow",
          "Action": "sts:AssumeRole",
          "Principal": {
               "AWS": ["arn:aws:iam::111111111111:user/Bob"]
               ]
          }
     }]
}
```

Specify the exact IAM user you want to grant access.

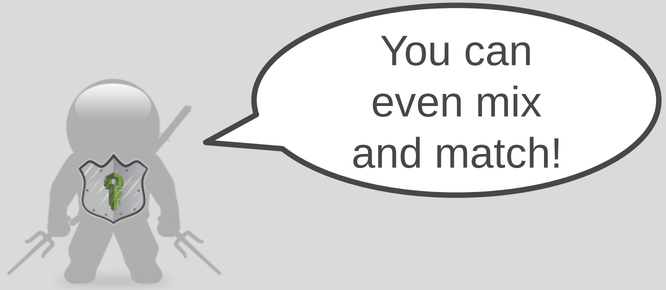
Grant cross-account conditional federated access

```
{
     "Version": "2012-10-17",
     "Statement": [{
          "Effect": "Allow",
          "Action": "sts:AssumeRole",
          "Principal": {
               "AWS": ["arn:aws:sts::111111111111:assumed-role/ROLENAME/ROLESESSIONNAME"]
               ]
          }
     }]
}
```

This will grant access to the specified federated user

Grant cross-account conditional federated access

```
{
     "Version": "2012-10-17",
     "Statement": [{
          "Effect": "Allow",
          "Action": "sts:AssumeRole",
          "Principal": {
               "AWS": [
                    "arn:aws:iam::111111111111:user/Bob",
                    "arn:aws:sts::111111111111:assumed-role/ROLENAME/ROLESESSIONNAME"
               ]
          }
     }]
}
```



EC2 resource-level permissions

- Previously, policies applied to all EC2 resources
- Permissions can be set per resource
- Ex: assign which users can stop, start, or terminate a particular instance

EC2 policies before resource-level permissions

```
{
   "Statement": [{
      "Effect": "Allow",
      "Action": ["ec2:TerminateInstances"],
      "Resource":"*"
   }
  ]
}
```

EC2 policies after resource-level permissions

```
{
   "Statement": [{
      "Effect": "Allow",
      "Action": ["ec2:TerminateInstances"],
      "Resource":
       "arn:aws:ec2:us-east-1:123456789012:instance/i-abc12345"
     }
  ]
}
```

EC2 policies after resource-level permissions

```
{
   "Statement": [{
      "Effect": "Allow",
      "Action": ["ec2:TerminateInstances"],
      "Resource":
       "arn:aws:ec2:us-east-1:123456789012:instance/*"
     }
  ]
}
```

EC2 policies after resource-level permissions

```
{
   "Statement": [{
      "Effect": "Allow",
      "Action": ["ec2:TerminateInstances"],
      "Resource":
       "arn:aws:ec2:us-east-1:123456789012:instance/*",
      "Condition": {
       "StringEquals": {"ec2:ResourceTag/department": "dev"}
      }
    }
  ]
}
```

Supported EC2 resource types

Supports many different resource types, including:

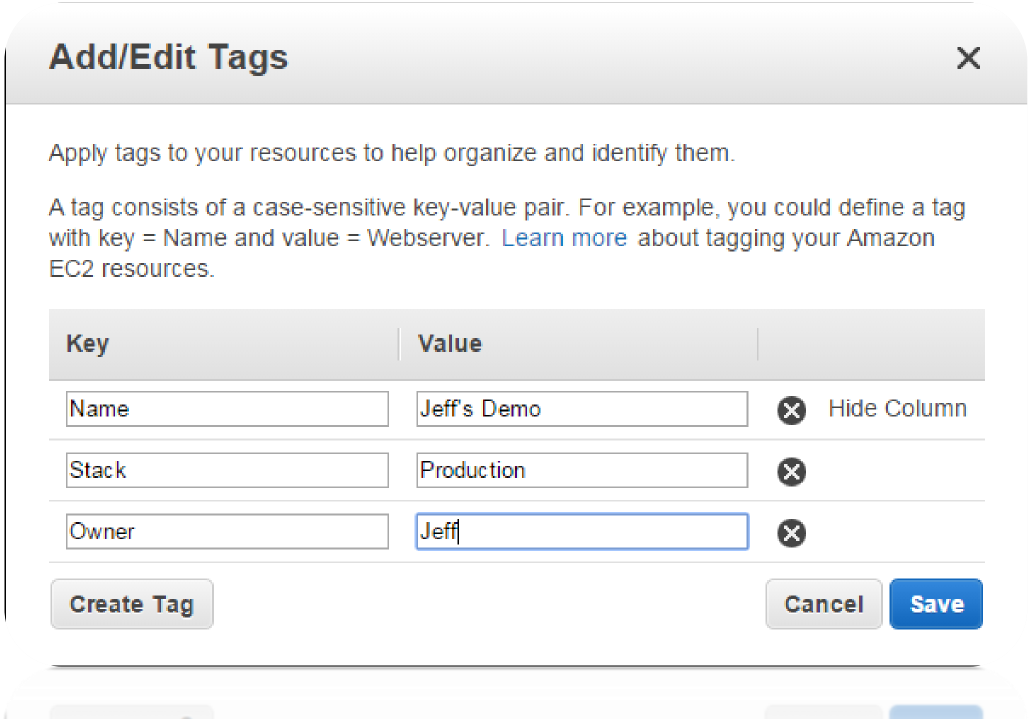
- Customer gateway
- DHCP options set
- Image
- Instance
- Instance profile
- Internet gateway
- Key pair
- Network ACL
- Network interface
- Placement group
- Route table
- Security group
- Snapshot
- Subnet
- Volume
- VPC
- VPC peering connection

Supported EC2 actions

Type of resource	Actions Accurate as of 11/16/2016
EC2 instances	RebootInstances, RunInstance, StartInstances, StopInstances, TerminateInstances, AttachClassicLinkVpc, AttachVolume, DetachClassicLinkVpc, DetachVolume, GetConsoleScreenshot
Customer gateway	DeleteCustomerGateway
DHCP options sets	DeleteDhcpOptions
Internet gateways	DeleteInternetGateway
Network ACLs	DeleteNetworkAcl, DeleteNetworkAclEntry
Route tables	DeleteRoute, DeleteRouteTable
Security groups	AuthorizeSecurityGroupEgress, AuthorizeSecurityGroupIngress, DeleteSecurityGroup, RevokeSecurityGroupEgress, RevokeSecurityGroupIngress, AttachClassicLinkVpc, RunInstances
Volumes	AttachVolume, DeleteVolume, DetachVolume, RunInstances
VPC peering connections	AcceptVpcPeeringConnection, CreateVpcPeeringConnection, DeleteVpcPeeringConnection, RejectVpcPeeringConnection, DisableVpcClassicLink, EnableVpcClassicLink

Categorize your EC2 resources

Use tags as a resource attribute


Add/Edit Tags

Apply tags to your resources to help organize and identify them.

A tag consists of a case-sensitive key-value pair. For example, you could define a tag with key = Name and value = Webserver. Learn more about tagging your Amazon EC2 resources.

Key	Value	
Name	Jeff's Demo	Hide Column
Stack	Production	
Owner	Jeff	

- Allows user-defined models
- “Prod”/”Dev”
- “Cost Center X”
- “Department Y”
- “Project reInvent”

EC2 resource-level permissions

Demo

- Goal: Limit a user from starting, stopping, or terminating an instance unless the instance is owned by the user.
- We'll examine:
  - Adding an owner tag to a nce.
  - A policy that grants a use he EC2 console.
  - A policy that uses variab cess based on an owner tag.

Locking down access to EC2 instances

```
{
  "Version": "2012-10-17",
  "Statement": [{
      "Effect": "Allow",
      "Action": "ec2:Describe*",
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": "elasticloadbalancing:Describe*",
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "cloudwatch:ListMetrics","cloudwatch:GetMetricStatistics","cloudwatch:Describe*"
      ],
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": "autoscaling:Describe*",
      "Resource": "*"
    }]
}
```

Allows seeing everything from the EC2 console

This is the AWS managed policy named AmazonEC2ReadOnlyAccess

Locking down access to EC2 instances

```
{
      "Version": "2012-10-17",
      "Statement": [{
            "Sid": "THISLIMITSACCESSTOOWNINSTANCES",
            "Effect": "Allow",
            "Action": ["ec2:RebootInstances",
                           "ec2:StartInstances",
                           "ec2:StopInstances",
                           "ec2:TerminateInstances"
            ],
            "Resource": "arn:aws:ec2:*:123456789012:instance/*",
            "Condition": {
                  "StringEquals": {
                        "ec2:ResourceTag/Owner": "${aws:username}"
                  }
            }
      }
    ]
}
```

Version is required here because we're using variables

Only allowed if this tag condition is true

Specify the tag key and value here

Limit EC2 instance types

Demo

- Goal: Limit a user from starting an instance unless the instance is `t1.*`, `t2.*`, `m3.*`
- We'll do the following:
  - – Create a new IAM group.
  - – Create a managed policy tarting EC2 instances to specific instance types.
  - – Attach that managed pol M group.

Locking down access to instance types

```
{
  "Version": "2012-10-17",
  "Statement": [{
      "Effect": "Allow",
      "NotAction": ["iam:*","ec2:RunInstances"],
      "Resource": "*"},
    {
      "Effect": "Allow",
      "Action": "ec2:RunInstances",
      "NotResource": [
        "arn:aws:ec2:us-east-2:012345678912:instance/*",
        "arn:aws:ec2:eu-west-1:012345678912:instance/*"]},
    {
      "Effect": "Allow",
      "Action": "ec2:RunInstances",
      "Resource": [
        "arn:aws:ec2:us-east-2:012345678912:instance/*",
        "arn:aws:ec2:eu-west-1:012345678912:instance/*"],
      "Condition": {
        "StringLike": {"ec2:InstanceType": ["t1.*","t2.*","m3.*"]}
      }
    }
  ]
}
```

Include all services/actions you want to exclude!

Grants access to everything you need to launch an instance, except the actual instance

Lock down types here

Take advantage of `IfExists` conditional operator

- Many condition keys only exist for certain resource types.
- If you test for a nonexistent key, your policy will fail to evaluate (in other words, access denied).
- You can add `IfExists` at the end of any condition operator except the `Null` condition (for example, `StringLikeIfExists`).
- Allows you to create policies that “don’t care” if the key is not present.

`StringNotLikeIfExists` example

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "ec2:*",
            "Resource": "*"
        },
        {
            "Effect": "Deny",
            "Action": "ec2:RunInstances",
            "Resource": "arn:aws:ec2:*:012345678901:instance/*",
            "Condition": {
                "StringNotLikeIfExists": {
                    "ec2:InstanceType": [
                        "t1.*", "t2.*", "m3.*"
                    ]
                }
            }
        }
    ]
}
```

Granting access to all EC2 actions on all resources in all regions

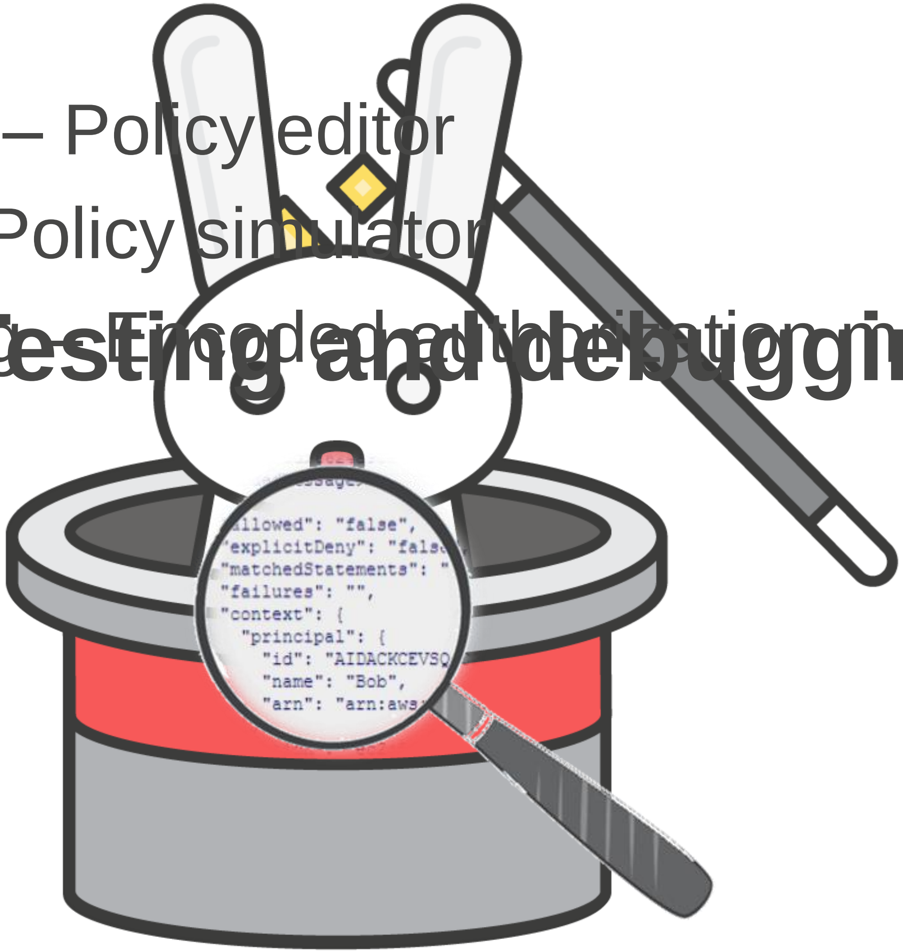
Explicitly deny access to `RunInstances`, for all regions and resources if condition is met

Only apply this condition if this `InstanceType` key exists

That sounds great. What if I get stuck? Can I call you?

Magic: Testing and debugging

- Authoring – Policy editor
- Testing – Policy simulator
- Debugging – Encoded authorization message (EC2)



```
allowed": "false",
'explicitDeny": "fals
"matchedStatements": "
"failures": "",
"context": {
  "principal": {
    "id": "AIDACKCEVSQ
    "name": "Bob",
    "arn": "arn:aws
```

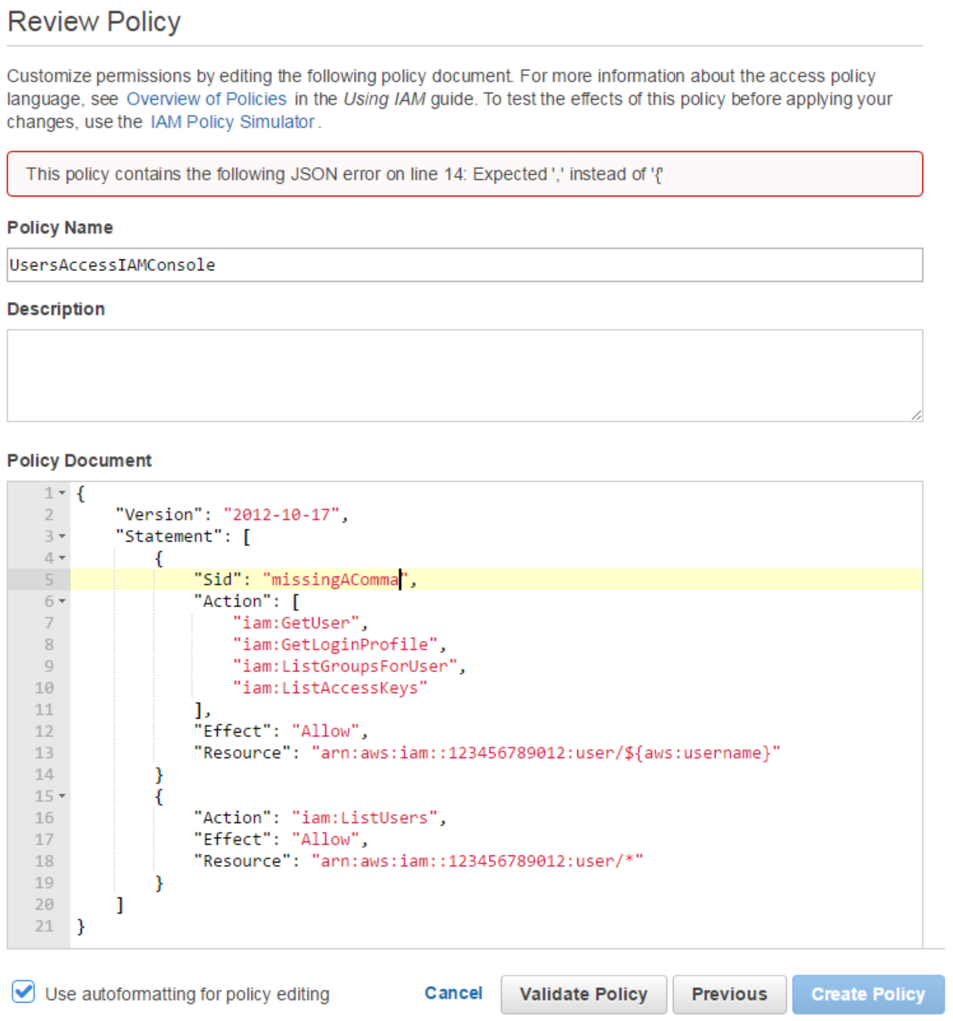
Policy editor

Policy validation checks:

- JSON errors
- Policy grammar errors

Policy formatting:

- On-demand
- Autoformatting



Review Policy

Customize permissions by editing the following policy document. For more information about the access policy language, see Overview of Policies in the *Using IAM* guide. To test the effects of this policy before applying your changes, use the IAM Policy Simulator.

This policy contains the following JSON error on line 14: Expected ',' instead of '{'

Policy Name

UsersAccessIAMConsole

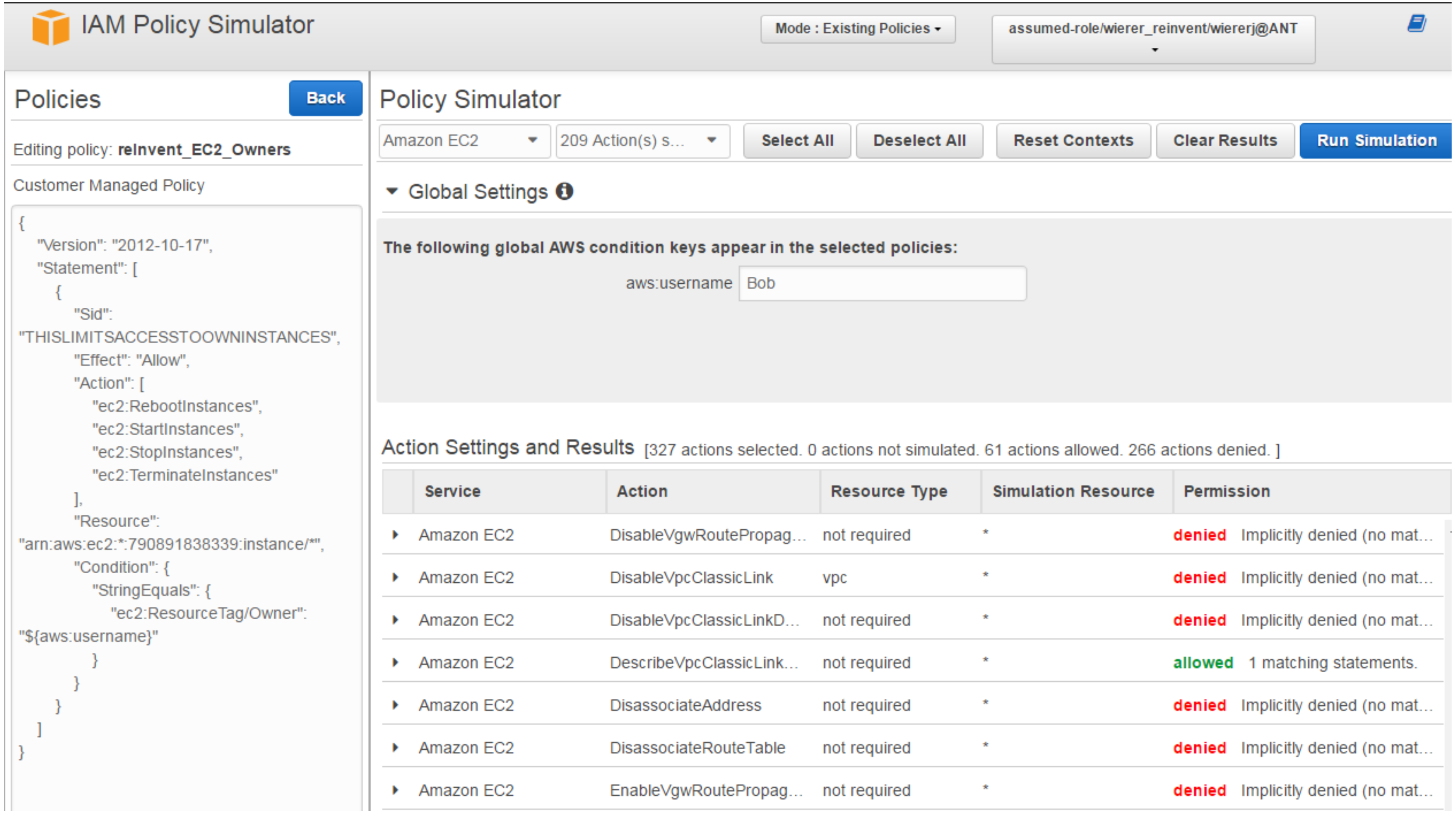
Description

Policy Document

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "missingAComma",
            "Action": [
                "iam:GetUser",
                "iam:GetLoginProfile",
                "iam:ListGroupsForUser",
                "iam:ListAccessKeys"
            ],
            "Effect": "Allow",
            "Resource": "arn:aws:iam::123456789012:user/${aws:username}"
        }
        {
            "Action": "iam:ListUsers",
            "Effect": "Allow",
            "Resource": "arn:aws:iam::123456789012:user/*"
        }
    ]
}
```

[✓] Use autoformatting for policy editing

Policy simulator



Editing policy: **reInvent_EC2_Owners**

Customer Managed Policy

```
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Sid":
"THISLIMITSACCESSTOOWNINSTANCES",
         "Effect": "Allow",
         "Action": [
            "ec2:RebootInstances",
            "ec2:StartInstances",
            "ec2:StopInstances",
            "ec2:TerminateInstances"
         ],
         "Resource":
"arn:aws:ec2:*:790891838339:instance/*",
         "Condition": {
            "StringEquals": {
               "ec2:ResourceTag/Owner":
"${aws:username}"
            }
         }
      }
   ]
}
```

The following global AWS condition keys appear in the selected policies:

aws:username: Bob

Action Settings and Results [327 actions selected. 0 actions not simulated. 61 actions allowed. 266 actions denied.]

Service	Action	Resource Type	Simulation Resource	Permission
Amazon EC2	DisableVgwRoutePropag…	not required	*	denied Implicitly denied (no mat…
Amazon EC2	DisableVpcClassicLink	vpc	*	denied Implicitly denied (no mat…
Amazon EC2	DisableVpcClassicLinkD…	not required	*	denied Implicitly denied (no mat…
Amazon EC2	DescribeVpcClassicLink…	not required	*	allowed 1 matching statements.
Amazon EC2	DisassociateAddress	not required	*	denied Implicitly denied (no mat…
Amazon EC2	DisassociateRouteTable	not required	*	denied Implicitly denied (no mat…
Amazon EC2	EnableVgwRoutePropag…	not required	*	denied Implicitly denied (no mat…

Decoding the EC2 authorization message

- The decoded message includes:
  - Whether the request was denied due to an explicit deny or absence of an explicit allow.
  - The principal who made the request.
  - The requested action.
  - The requested resource.
  - The values of condition keys in the context of the user's request.
- Requires permissions to call

  `sts:DecodeAuthorizationMessage`

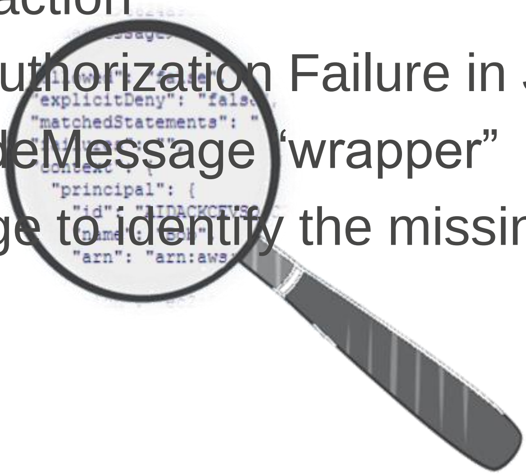
EjE8j1AEXAMPLEDOwukwv5KbOS2j0jiZTsl_ESOmbSFnq
Y91EIGRRQplweQ5CQDQmaS7DBMfJDqwpZAm
ORTOKHgeNZdcChNCDacLE6YGEAlVyTl8yoc8Ukcb8A8q
4i3a 4A5Izf4HGJ6VHoOYPExvwVcDy
C gM8nJDaxELFcgjOa4RxfsDcpPe5mONA
_dpA6Q6IJRjYNWxjNEEtky
52OMn8X7ai3SkRS7V33dpxcwpaKEHE
GoWbAPY7LuylJqtDysfP
aKkIGIPHXPjC4IT63ttMvTObDdDaOleR
TA gsb7pQTnqQAmqQBhvxWS
wDf5b2Vy3qeJ_LY oO7PwMfjuMK6SZjCL5tgwWRqu
_5UPxpZdY5DdGmKb FfrDPVENevHUe

```
"allowed": "false",
"explicitDeny": "false
"matchedStatements": "
"failures": "",
"context": {
  "principal": {
    "id": "AIDACKCEVSQ
    "name": "Bob",
    "arn": "arn:aws
```

Decoding the EC2 authorization message

Demo

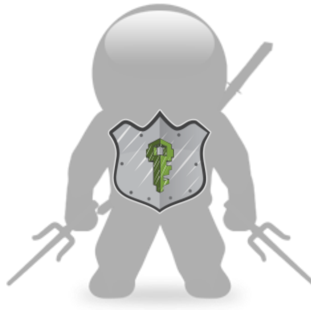
- Goal: Determine what caused an EC2 authorization failure
- We'll do the following:
  - Try to call an EC2 action
  - Capture the EC2 Authorization Failure in JSON format
  - Remove the DecodeMessage "wrapper"
  - Format the message to identify the missing permission



```
"explicitDeny": "false",
"matchedStatements": "
"principal": {
"id": "AIDACKCEVS
"arn": "arn:aws
```

Summary

- IAM provides access control for your AWS account.
- The policy language authorizes that access.
- All applicable policies are evaluated.
  - Users are denied access by default.
  - A deny always trumps an allow.
- Use policy variables and remember the version!
- Use managed policies: they make life easier.
- Keep in mind which EC2 actions or resources are currently supported.



Congratulations

You are now a certified

AWS IAM Policy Ninja

Disclaimer: Not really. This is not a real certification, but thank you for sticking around until the end. ☺

Remember to complete your evaluations!

Additional resources

- AWS IAM home page: http://aws.amazon.com/iam
- Documentation
  - http://aws.amazon.com/documentation/iam/
  - http://docs.aws.amazon.com/AWSEC2/latest/APIReference/ec2-api-permissions.html
- AWS Security Blog
  - https://aws.amazon.com/blogs/security/demystifying-ec2-resource-level-permissions/
  - https://aws.amazon.com/blogs/security/granting-users-permission-to-work-in-the-amazon-ec2-console/
  - https://aws.amazon.com/blogs/security/how-to-create-a-limited-iam-administrator-by-using-managed-policies/
- IAM forum: https://forums.aws.amazon.com/forum.jspa?forumID=76
- Twitter: @AWSIdentity

Thank you!