AWS re:Invent

SEC301: Top 10 IAM best practices

Anders Samuelsson, AWS Identity and Access

November 13, 2013

amazon web services

What we will cover today

- Quick overview of AWS Identity and Acces Management (IAM)
- Top 10 IAM best practices to secure your AWS environment (with a lot of demos)

AWS Identity and Access Management (IAM)

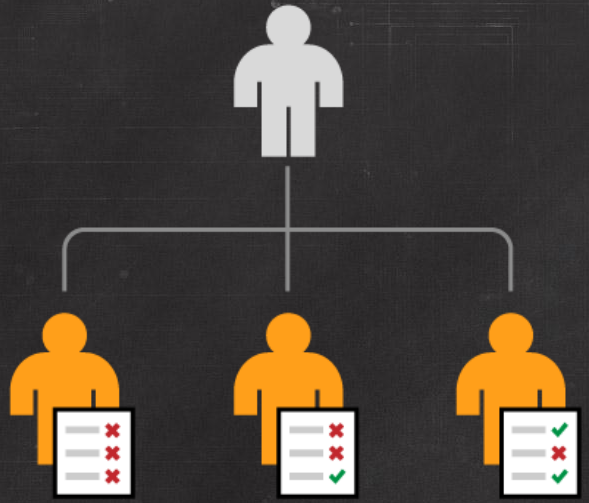
*IAM enables **you** to **control** who can do what in your AWS account*

- Users, Groups, Roles, Permissions
- Control…
  - Centralized
  - Fine-grained - APIs, resources and AWS Management Console
- Security…
  - Secure by default
  - Multiple users, individual security credentials and permissions

Top 10 IAM best practices

Top 10 IAM best practices

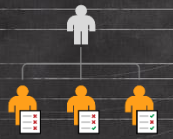
1. Users
2. Groups
3. Permissions
4. Passwords
5. MFA
6. Roles
7. Sharing
8. Rotation
9. Conditions
10. Root



1. Users

Create individual users

1. Create individual users

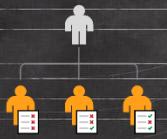


Benefits

- Unique credentials
- Individual credential rotation
- Individual permissions

How to steps

- Identify which IAM users you want to create ☺
- Use the IAM Console, CLI or API to:
  - Create user
  - Assign credentials
  - Assign permissions



1. Create individual users

2. Groups

Manage permissions with groups

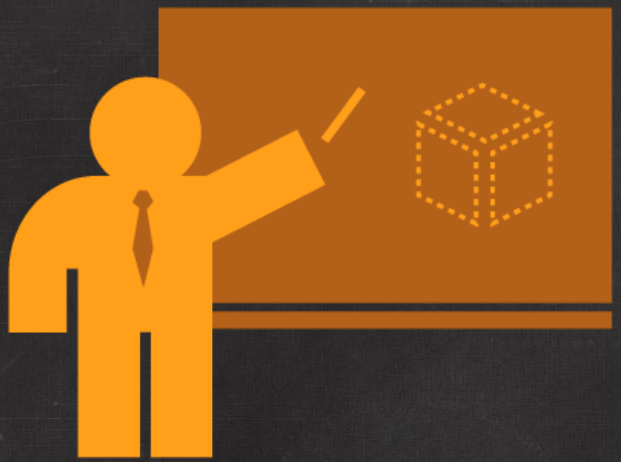
2. Manage permissions with groups

Benefits

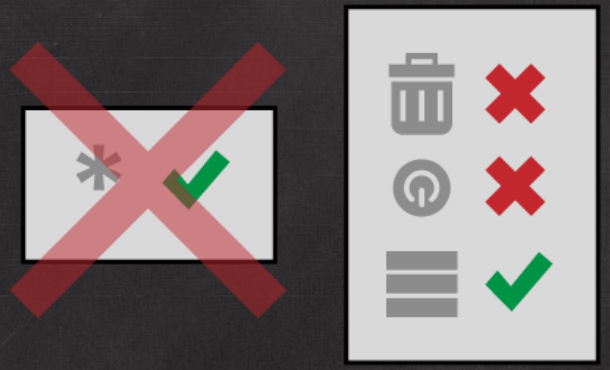
- Easier to assign the same permissions to multiple users
- Simpler to re-assign permissions based on change in responsibilities
- Only one change to update permissions for multiple users

How to steps

- Map permissions to a specific business function
- Assign users to that function
- Manage groups in the *Group* section of the IAM Console



2. Manage permissions with groups



3. Permissions

Grant least privilege

3. Grant least privilege

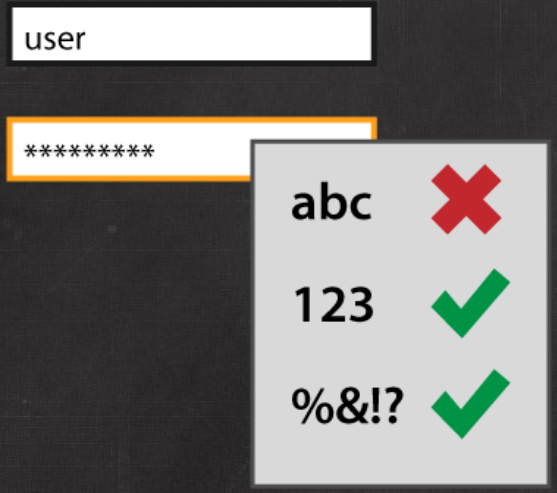
Benefits

- More granular control
- Less chance of people making mistakes
- Easier to relax than to tighten up

How to steps

- Identify what permissions are required
- Password/Access keys?
- Avoid assigning *:* policy
- Use policy templates

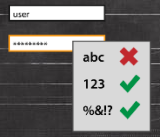
3. Grant least privilege



4. Passwords

Configure a strong password policy

4. Enforce a strong password policy

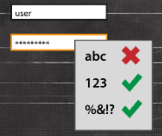


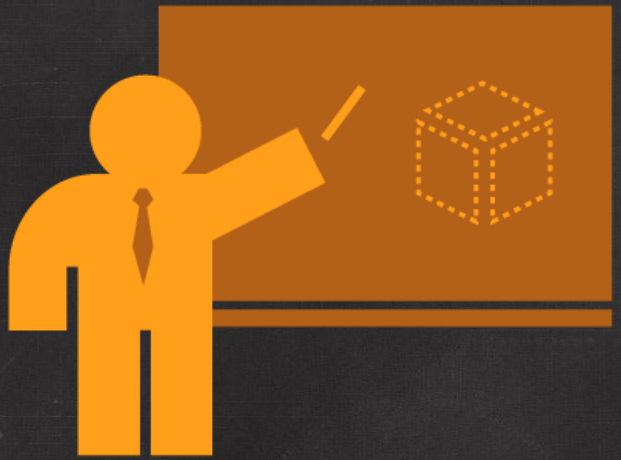
Benefits

- Ensures your users and your data are protected

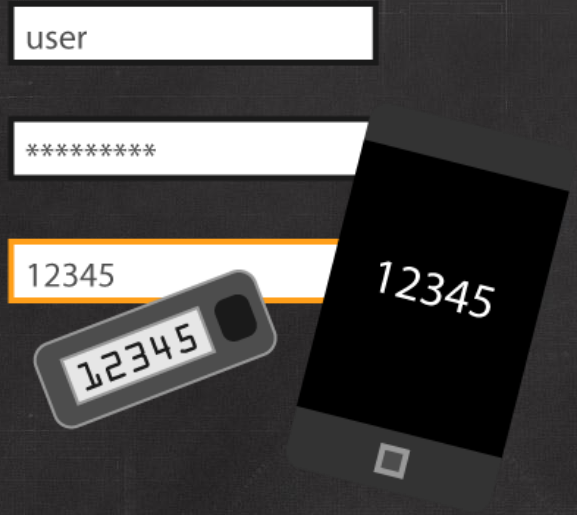
How to steps

- What is your company's password policy?
- You can configure
  - Minimum password length
  - Require any combination of:
    - One uppercase letter
    - One lowercase letter
    - One number
    - One non-alphanumeric character





4. Configure a strong password policy



5. MFA

Enable multi-factor authentication for privileged users

5. Enable Multi-Factor Authentication for privileged users

Benefits

- Supplements user name and password to require a one-time code during authentication

How to steps

- Choose type of MFA
  - Virtual MFA
  - Hardware
- Use IAM Console to assign MFA device

5. Enable MFA for privileged users

6. Roles

Use IAM roles for Amazon EC2 instances

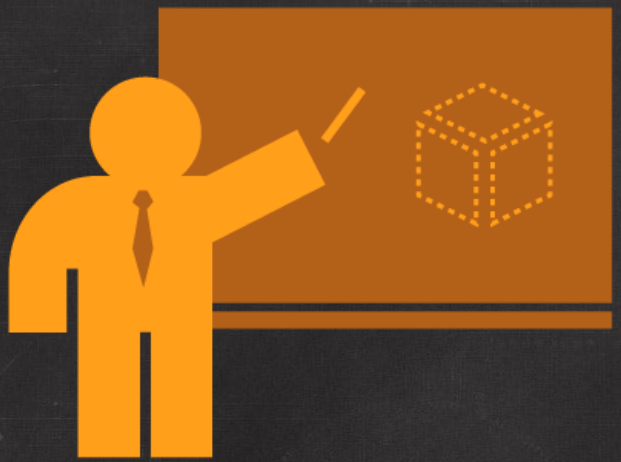
6. Use IAM roles for Amazon EC2 instances

Benefits

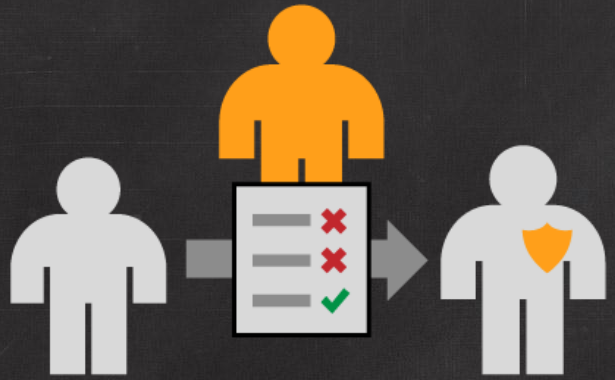
- Easy to manage access keys on EC2 instances
- Automatic key rotation
- Assign least privilege to the application
- AWS SDKs fully integrated

How to steps

- Create a role
- Launch instances with the role
- If not using SDKs, sign all requests to AWS services with the roles' temporary credentials



6. Use IAM roles for EC2 instances



7. Sharing

Use IAM roles to share access

7. Use IAM roles to share access

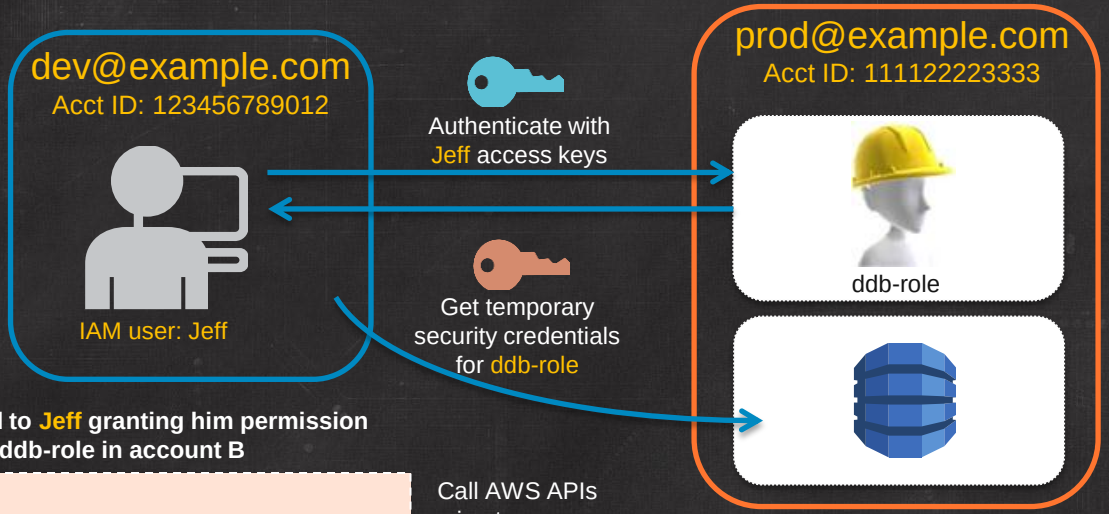
Benefits

- No need to share security credentials
- Easy to break sharing relationship
- Use cases
  - Cross-account access
  - Intra-account delegation
  - Federation

How to steps

- Create a role
  - Specify who you trust
  - Describe what the role can do
- Share the name of the role

Cross-account access – How does it work?



Permissions assigned to Jeff granting him permission to assume ddb-role in account B

```
{ "Statement": [
  {
   "Effect": "Allow",
   "Action": "sts:AssumeRole",
   "Resource":
       "arn:aws:iam::111122223333:role/ddb-role"
}]}
```

Call AWS APIs using temporary security credentials of ddb-role

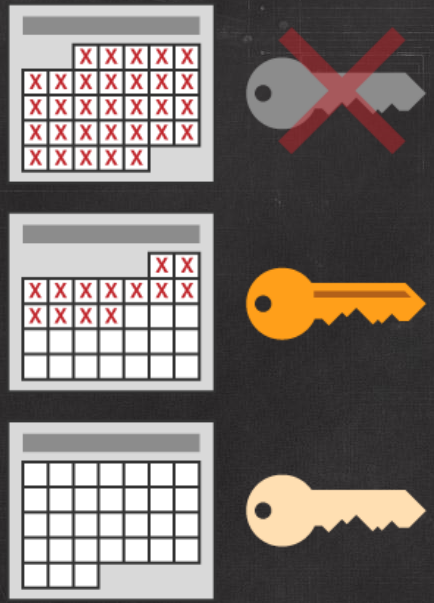
Permissions assigned to ddb-role

```
{ "Statement": [
  {
    "Action": [
      "dynamodb:GetItem",
      "dynamodb:BatchGetItem",
      "dynamodb:Query",
      "dynamodb:Scan",
      "dynamodb:DescribeTable",
      "dynamodb:ListTables"
    ],
    "Effect": "Allow",
    "Resource": "*"
}]}
```

```
{ "Statement": [
  {
   "Effect":"Allow",
   "Principal":{"AWS":"123456789012"},
   "Action":"sts:AssumeRole"
}]}
```

ddb-role trusts IAM users from the AWS account dev@example.com (123456789012)

7. Use IAM roles to share access



8. Rotation

Rotate security credentials regularly

8. Rotate security credentials regularly

Benefits

- Normal best practice

How to steps

- Grant IAM user permission to rotate credentials
- Change password in IAM console
- IAM roles for EC2 automatically rotate credentials

Enabling credential rotation for IAM users

(enable password rotation sample policy)

Password

```
{ "Version":"2012-10-17",
  "Statement": [{
  "Effect": "Allow",
  "Action": "iam:ChangePassword",
  "Resource":
    "arn:aws:iam::123456789012:user/${aws:username}"
 }
]}
```

Enforcing a password policy will automatically enable IAM users to manage their passwords

Enabling credential rotation for IAM users

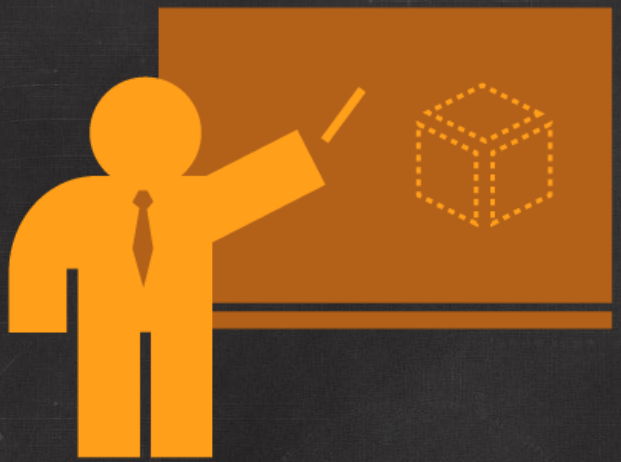
(enable access key rotation sample policy)

Access Keys

```
{ "Version":"2012-10-17",
  "Statement": [{
  "Effect": "Allow",
  "Action": [
      "iam:CreateAccessKey",
      "iam:DeleteAccessKey",
      "iam:ListAccessKeys",
      "iam:UpdateAccessKey"],
  "Resource":
     "arn:aws:iam::123456789012:
user/${aws:username}"
 }
]}
```

Steps to rotate access keys

1. While the first set of credentials is still active, create a second set of credentials, which will also be active by default.
2. Update all applications to use the new credentials.
3. Change the state of the first set of credentials to Inactive.
4. Using only the new credentials, confirm that your applications are working well.
5. Delete the first set of credentials



8. Rotate security credentials regularly

9. Conditions

Restrict privileged access further with conditions

9. Restrict privileged access further with conditions

Benefits

- Additional granularity when defining permissions
- Can be enabled for any AWS service API
- Minimizes chances of accidentally performing privileged actions

How to steps

- Use conditions where applicable
- Two types of conditions
  - AWS common
  - Service-specific

Restrict privileged access further with conditions

MFA

```
{
   "Statement":[{
   "Effect":"Deny",
   "Action":["ec2:TerminateInstances"],
   "Resource":["*"],
   "Condition":{
     "Null":{"aws:MultiFactorAuthAge":"true"}
}}]}
```

Enables a user to terminate EC2 instances only if the user has authenticated with their MFA device.

SSL

```
{
   "Statement":[{
   "Effect":"Allow",
   "Action":"iam:*AccessKey*",
   "Resource”:"arn:aws:iam::123456789012:user/*",
   "Condition":{
      "Bool":{“aws:SecureTransport":"true"},
}}]}
```

Enables a user to manage access keys for all IAM users only if the user is coming over SSL.

SourceIP

```
{
   "Statement":[{
   "Effect":"Allow",
   "Action":["ec2:TerminateInstances“],
   "Resource":["*“],
   "Condition":{
      "IpAddress":{"aws:SourceIP":"192.168.176.0/24"}
}}]}
```

Enables a user to terminate EC2 instances only if the user is accessing EC2 from the 192.168.176.0/24 address range.

9. Restrict privileged access further with conditions

10. Root

Reduce or remove use of root

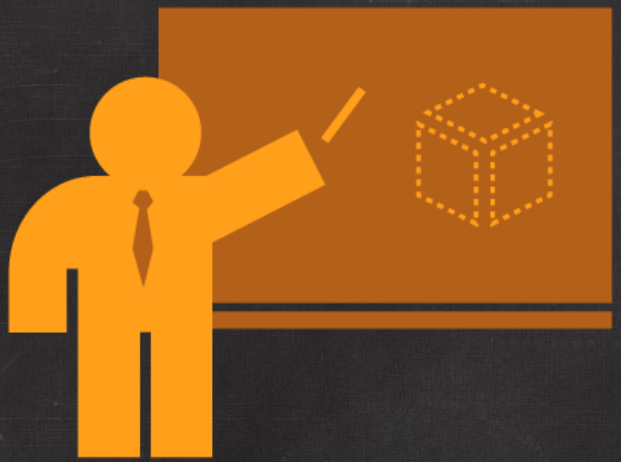
10. Reduce or remove use of root

Benefits

- Reduce potential for misuse of credentials

How to steps

- Security Credentials Page
  - Delete access keys
  - Activate a MFA device
- Ensure you have set a “strong” password



10. Reduce or remove use of root

Top 10 IAM best practices

1. **Users** – Create individual users
2. **Groups** – Manage permissions with groups
3. **Permissions** – Grant least privilege
4. **Password** – Configure a strong password policy
5. **MFA** – Enable MFA for privileged users
6. **Roles** – Use IAM roles for EC2 instances
7. **Sharing** – Use IAM roles to share access
8. **Rotate** – Rotate security credentials regularly
9. **Conditions** – Restrict privileged access further with conditions
10. **Root** – Reduce or remove use of root

Top 10 IAM best practices

1. **Users** – Create individual users
2. **Groups** – Manage permissions with groups
3. **Permissions** – Grant least privilege
4. **Password** – Configure a strong password policy
5. **MFA** – Enable MFA for privileged users
6. **Roles** – Use IAM roles for EC2 instances
7. **Sharing** – Use IAM roles to share access
8. **Rotate** – Rotate security credentials regularly
9. **Conditions** – Restrict privileged access further with conditions
0. **Root** – Reduce or remove use of root

Additional resources

- IAM detail page: http://aws.amazon.com/iam
- AWS forum: https://forums.aws.amazon.com/forum.jspa?forumID=76
- Documentation: http://aws.amazon.com/documentation/iam/
- AWS Security Blog: http://blogs.aws.amazon.com/security
- Twitter: @AWSIdentity

All IAM related sessions at re:Invent

ID	Title	Time, Room
CPN205	Securing Your Amazon EC2 Environment with AWS IAM Roles and Resource-Based Permissions	Wed 11/13 11am, Delfino 4003
SEC201	Access Control for the Cloud: AWS Identity and Access Management (IAM)	Wed 11/13 1.30pm, Marcello 4406
SEC301	TOP 10 IAM Best Practices	Wed 11/13 3pm, Marcello 4503
SEC302	Mastering Access Control Policies	Wed 11/13 4.15pm, Venetian A
SEC303	Delegating Access to Your AWS Environment	Thu 11/14 11am, Venetian A

GA23	**Come talk security with AWS**	**Thu 11/14 4pm, Toscana 3605**

AWS re:Invent

Please give us your feedback on this presentation

SEC301

As a thank you, we will select prize winners daily for completed surveys!

amazon web services