Damn Vulnerable AWS API

sdmay24-11

Attack Path 1: Ashler Benda, Karthik Kasarabada, Andrew Bowen

Attack Path 2: Garrett Arp, Ahmed Nasereddin, Ayo Ogunsola, Ethan Douglass

Industry Client: Jon Schnell on behalf of RSM US

Faculty Advisor: Julie Rursch

The Problem

- Over 90% of organizations in 2021 were using the cloud for some IT functionality (Source: O-Reilley)
- Threat actors are mastering exploitation of common oversights in cloud security. (Source: Palo-Alto)
- Personal data breaches were the 2nd most common Cyber Crime reported in 2022 (Source: FBI Internet Crime Report)
- Overall estimated losses due to *reported* Cyber Crimes in the last five years was $27.6 Billion (Source: FBI Internet Crime Report)
- Common free to access cyber security content providers, such as HTB and TryHackMe, have little to no cloud focused training exercises that are available for free

Who cares? What difference will it make?

Users

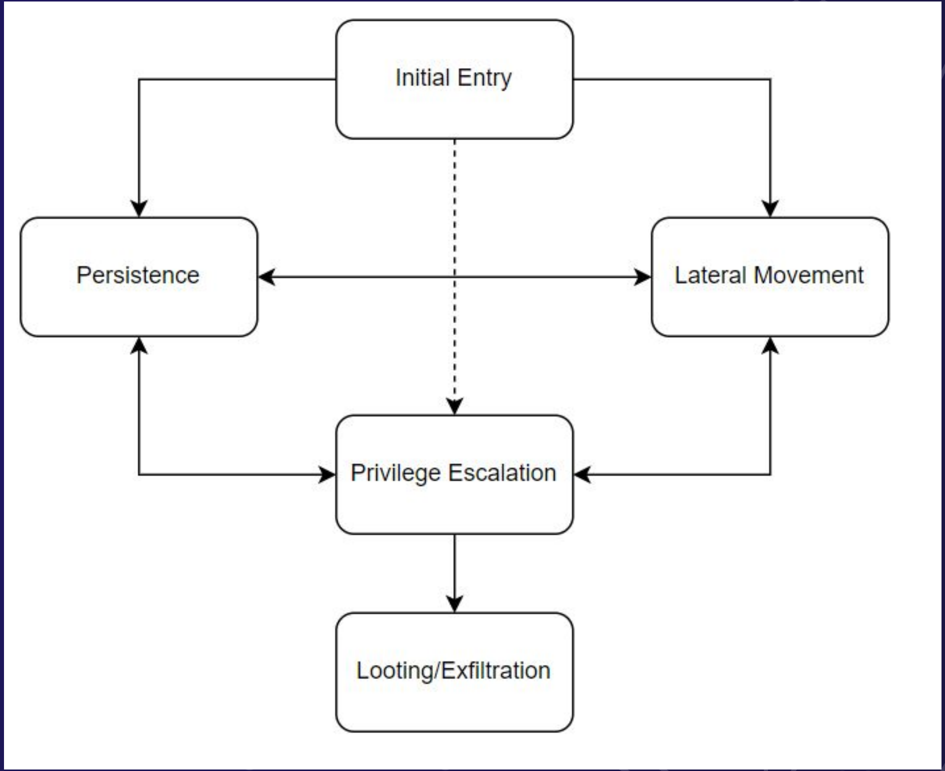
- IT Administrators
- Software Architects
- Cybersecurity Students
- Risk Consultants
- Application Developers

Use Cases

- Testing and Development
- Security
- Education

A Holistic Approach

- Based on typical cyber attack flow
- Two seperate attack paths
- Narrative for each component to model real world systems
- Four design iterations
    - Feedback from client
    - Feedback from industry professionals



Attack Path 1

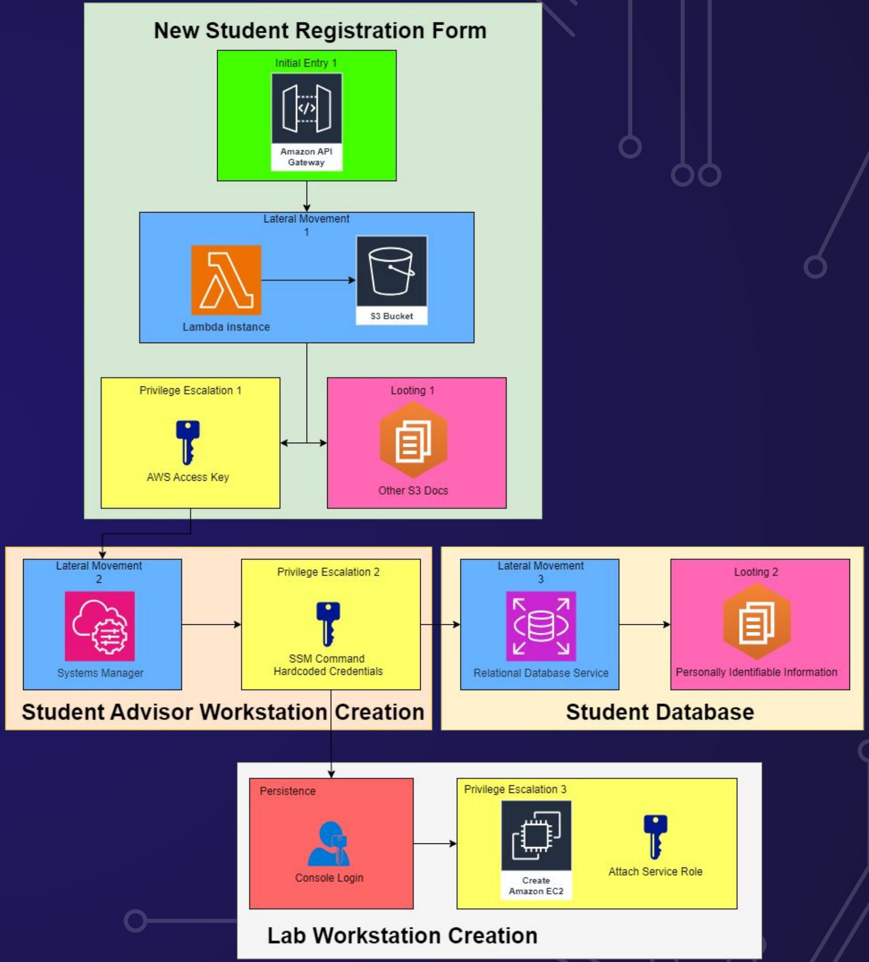
Attack Path based on University network scenario

Common AWS services

- API Gateway
- Lambda Function
- EC2 Instances
- S3 Bucket
- RDS

Common mistakes

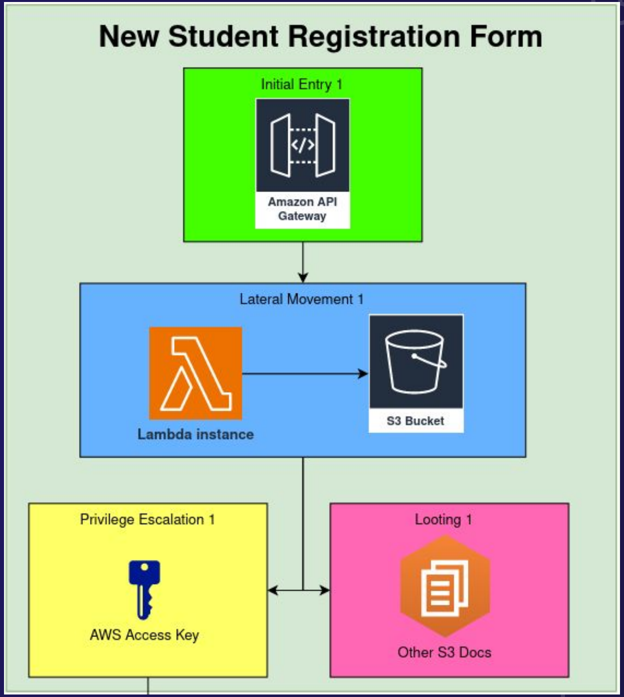
- Reusing passwords across services
- Not deleting old services/tools
- Hardcoding passwords
- Incorrect read/write permissions



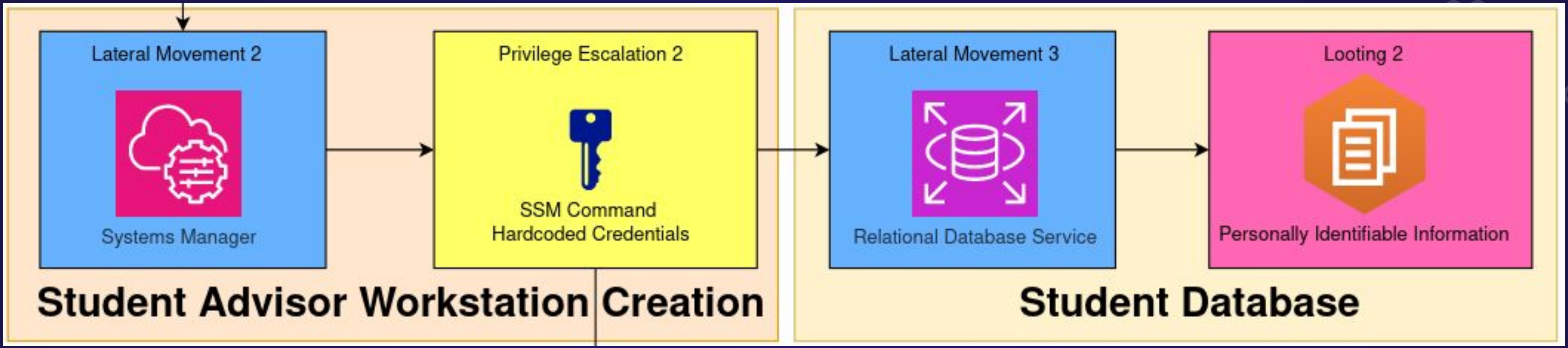
Attack Path 1

New Student Registration Form

- Misconfigured S3 Bucket Permissions
  - Accessible through Lambda Function and API Gateway
- Find AWS Access Key
- Loot other new student data

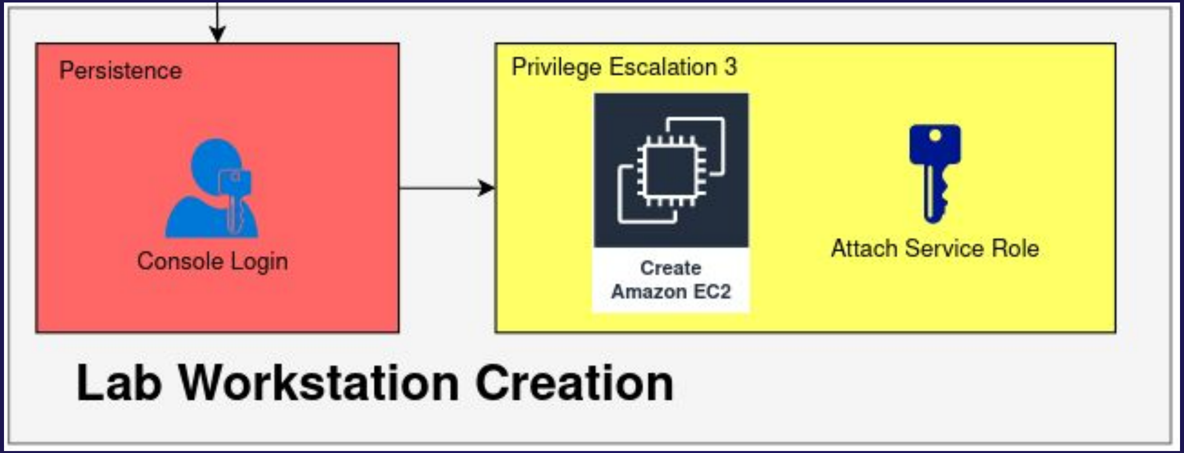


Attack Path 1



- Student Advisor Workstation Creation
  - Access Systems Manager with AWS Access Key
  - Find SSM command in the Systems Manager with hardcoded credentials
  - Login to RDS Database
  - Loot Student’s personal information, financial info, etc.

Attack Path 1



- Lab Workstation Creation
    - Access AWS Console with hardcoded credentials
    - Misconfigured Role policies
        - Can attach a higher role to a EC2 Instance
    - Create an EC2 Instance and attach a role that grant complete AWS control

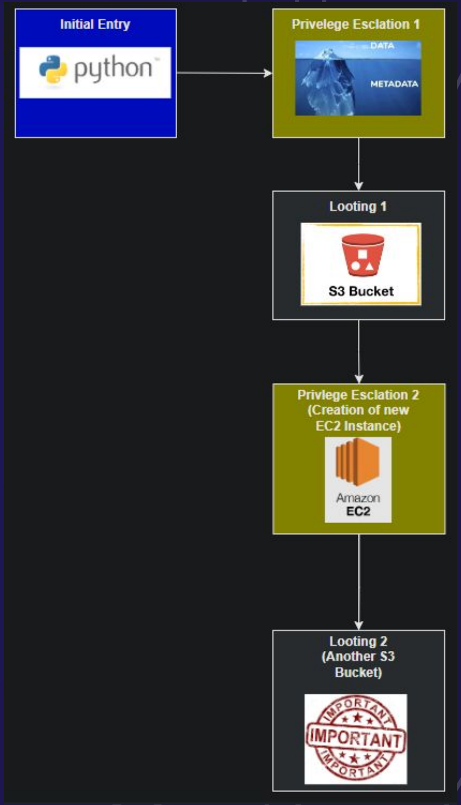
Final Design - Attack Path 2

Based on Red Team Methodology

5 sections follow methodology
- Initial Entry SSRF
- Privilege Escalation 1
- Looting 1
- Privilege Escalation 2
- Looting 2

Components leverage IMDS
- Wildly used metadata service in AWS
- Based on misconfigured services

Certain components based on real life attacks
- Initial Entry based on Capital one attack
- Priv Esc 2 based on United hack



System Design - Attack Path 2

Initial Entry SSRF

- Utilizing an SSRF attack on an Django web server
- Attackers will then find an admin page

Privilege Escalation 1

- Metadata api
- Temp credentials used to rollback policies

Looting 1

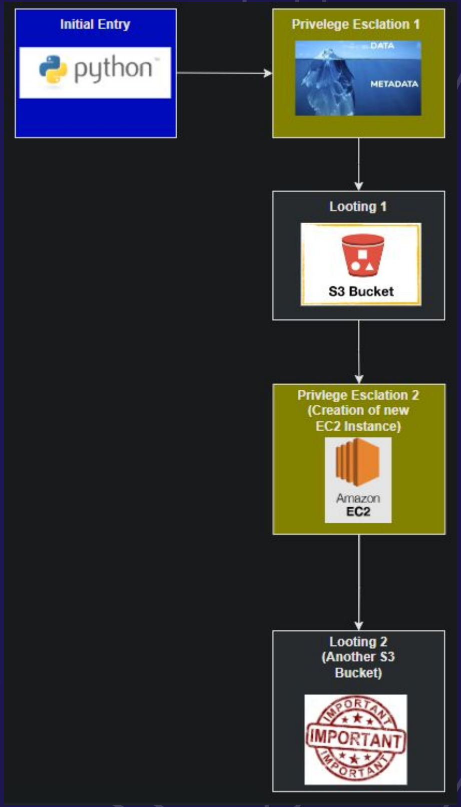
- Loot credentials from S3
- Using passrole permissions

Privilege Escalation 2

- Create new Ec2 instance, passrole more privileged role to ec2

Looting 2

- Loot sensitive information from second S3 with new privileged role



Testing

- Unit Testing
  - Run against each component in attack path
  - View state before attack, run attack, check state after
  - Primarily checked using logs

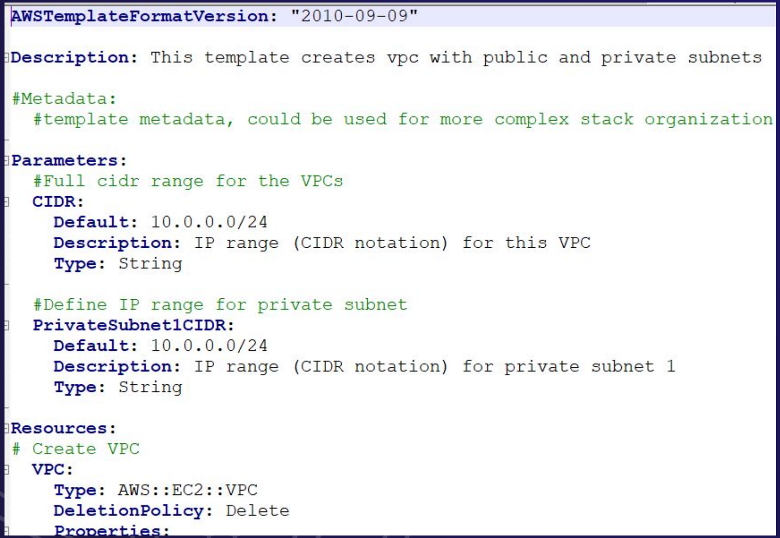
- Interface Testing
  - Component testing
  - Cross machine communication
  - Check unit testing across different components

- Security Testing
  - Project is purposely vulnerable
  - Testing is for unintended vulnerabilities
    - Still a learning opportunity
  - Check IAM policies to check for these routes

Current Progress

AP1

- API Gateway Created
- RDS and VPC Created
- Created Roles and User for Persistence



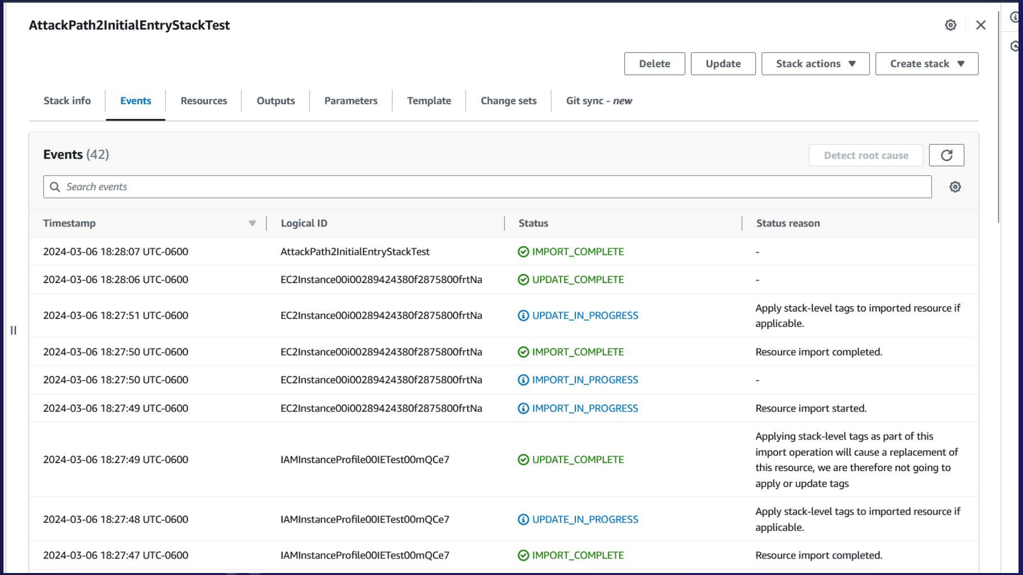
```
AWSTemplateFormatVersion: "2010-09-09"

Description: This template creates vpc with public and private subnets

#Metadata:
  #template metadata, could be used for more complex stack organization

Parameters:
  #Full cidr range for the VPCs
  CIDR:
    Default: 10.0.0.0/24
    Description: IP range (CIDR notation) for this VPC
    Type: String

  #Define IP range for private subnet
  PrivateSubnet1CIDR:
    Default: 10.0.0.0/24
    Description: IP range (CIDR notation) for private subnet 1
    Type: String

Resources:
# Create VPC
  VPC:
    Type: AWS::EC2::VPC
    DeletionPolicy: Delete
    Properties:
```

AP2

- Web Server for initial entry is near completion
- All resources needed have been created
- Looting 1/Priv Escalation 2 are a work in progress
  - Most Roles/Policies have been created, subject to modification
- Looting 2 is set up, just need to correctly set up priv escalation 2 for passing an existing role.
- Documenting steps for user as we go, also subject to change once we test.
- We have started creating and testing with cloud formation templates.

Current Progress

- Initial Entry Cloudformation stack test



AttackPath2InitialEntryStackTest

Delete | Update | Stack actions | Create stack

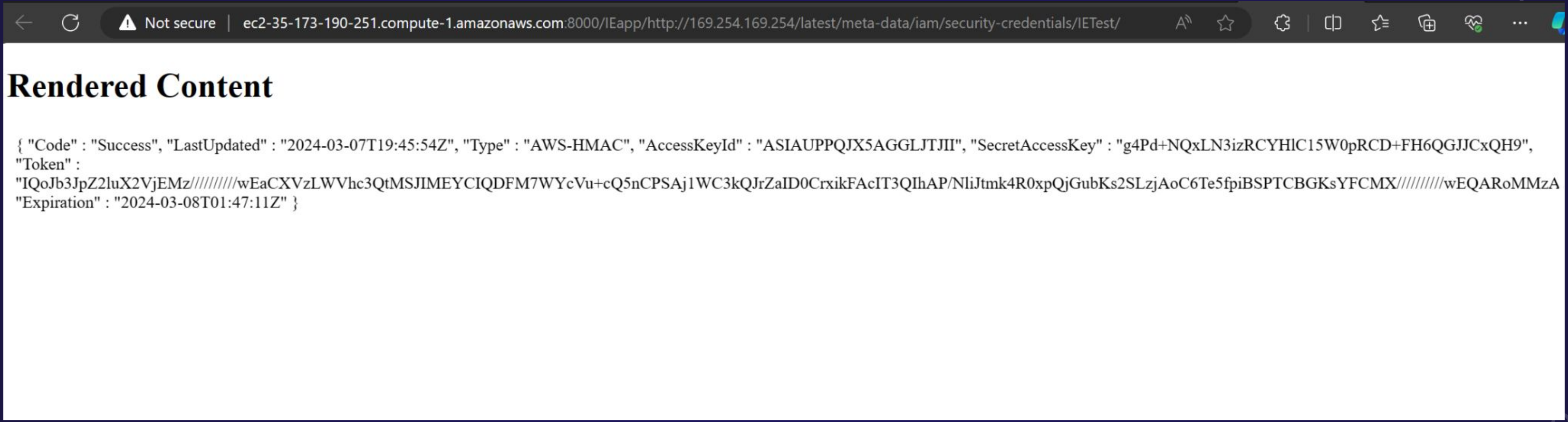
Stack info | Events | Resources | Outputs | Parameters | Template | Change sets | Git sync - *new*

Events (42)

Timestamp	Logical ID	Status	Status reason
2024-03-06 18:28:07 UTC-0600	AttackPath2InitialEntryStackTest	IMPORT_COMPLETE	-
2024-03-06 18:28:06 UTC-0600	EC2Instance00i00289424380f2875800frtNa	UPDATE_COMPLETE	-
2024-03-06 18:27:51 UTC-0600	EC2Instance00i00289424380f2875800frtNa	UPDATE_IN_PROGRESS	Apply stack-level tags to imported resource if applicable.
2024-03-06 18:27:50 UTC-0600	EC2Instance00i00289424380f2875800frtNa	IMPORT_COMPLETE	Resource import completed.
2024-03-06 18:27:50 UTC-0600	EC2Instance00i00289424380f2875800frtNa	IMPORT_IN_PROGRESS	-
2024-03-06 18:27:49 UTC-0600	EC2Instance00i00289424380f2875800frtNa	IMPORT_IN_PROGRESS	Resource import started.
2024-03-06 18:27:49 UTC-0600	IAMInstanceProfile00IETest00mQCe7	UPDATE_COMPLETE	Applying stack-level tags as part of this import operation will cause a replacement of this resource, we are therefore not going to apply or update tags
2024-03-06 18:27:48 UTC-0600	IAMInstanceProfile00IETest00mQCe7	UPDATE_IN_PROGRESS	Apply stack-level tags to imported resource if applicable.
2024-03-06 18:27:47 UTC-0600	IAMInstanceProfile00IETest00mQCe7	IMPORT_COMPLETE	Resource import completed.

Current Progress

- Simple web server login page



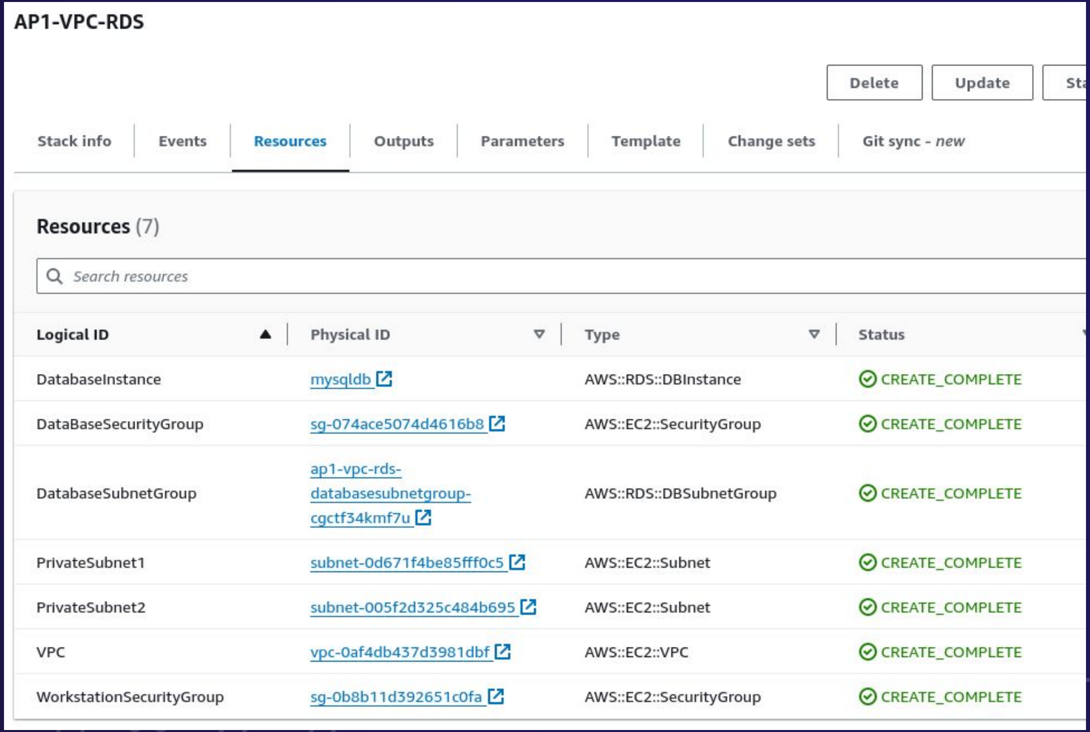
```
ec2-35-173-190-251.compute-1.amazonaws.com:8000/IEapp/http://169.254.169.254/latest/meta-data/iam/security-credentials/IETest/
```

Rendered Content

```
{ "Code" : "Success", "LastUpdated" : "2024-03-07T19:45:54Z", "Type" : "AWS-HMAC", "AccessKeyId" : "ASIAUPPQJX5AGGLJTJII", "SecretAccessKey" : "g4Pd+NQxLN3izRCYHlC15W0pRCD+FH6QGJJCxQH9",
"Token" :
"IQoJb3JpZ2luX2VjEMz//////////wEaCXVzLWVhc3QtMSJIMEYCIQDFM7WYcVu+cQ5nCPSAj1WC3kQJrZaID0CrxikFAcIT3QIhAP/NliJtmk4R0xpQjGubKs2SLzjAoC6Te5fpiBSPTCBGKsYFCMX//////////wEQARoMMzA
"Expiration" : "2024-03-08T01:47:11Z" }
```

Current Progress

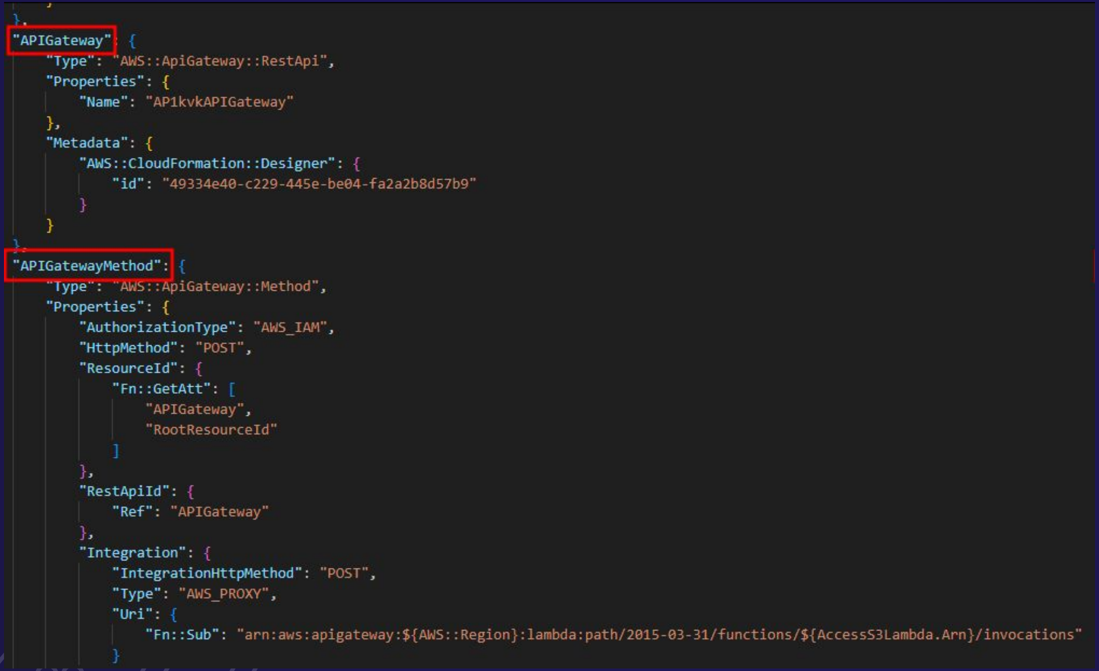
AP1 VPC RDS Stack



Logical ID	Physical ID	Type	Status
DatabaseInstance	mysqldb	AWS::RDS::DBInstance	CREATE_COMPLETE
DataBaseSecurityGroup	sg-074ace5074d4616b8	AWS::EC2::SecurityGroup	CREATE_COMPLETE
DatabaseSubnetGroup	ap1-vpc-rds-databasesubnetgroup-cgctf34kmf7u	AWS::RDS::DBSubnetGroup	CREATE_COMPLETE
PrivateSubnet1	subnet-0d671f4be85fff0c5	AWS::EC2::Subnet	CREATE_COMPLETE
PrivateSubnet2	subnet-005f2d325c484b695	AWS::EC2::Subnet	CREATE_COMPLETE
VPC	vpc-0af4db437d3981dbf	AWS::EC2::VPC	CREATE_COMPLETE
WorkstationSecurityGroup	sg-0b8b11d392651c0fa	AWS::EC2::SecurityGroup	CREATE_COMPLETE

Current Progress

- AP1 API Gateway Service setup in Cloud Formation



```
},
"APIGateway": {
    "Type": "AWS::ApiGateway::RestApi",
    "Properties": {
        "Name": "AP1kvkAPIGateway"
    },
    "Metadata": {
        "AWS::CloudFormation::Designer": {
            "id": "49334e40-c229-445e-be04-fa2a2b8d57b9"
        }
    }
},
"APIGatewayMethod": {
    "Type": "AWS::ApiGateway::Method",
    "Properties": {
        "AuthorizationType": "AWS_IAM",
        "HttpMethod": "POST",
        "ResourceId": {
            "Fn::GetAtt": [
                "APIGateway",
                "RootResourceId"
            ]
        },
        "RestApiId": {
            "Ref": "APIGateway"
        },
        "Integration": {
            "IntegrationHttpMethod": "POST",
            "Type": "AWS_PROXY",
            "Uri": {
                "Fn::Sub": "arn:aws:apigateway:${AWS::Region}:lambda:path/2015-03-31/functions/${AccessS3Lambda.Arn}/invocations"
            }
```

Project Plan

- Create SSM Command to login to database - March 10th
- Finish Individual parts with Units Test - by March 24th
- Integration Testing - by March 24th
- Full attack path testing - by April 7th
- Final Testing/Presentation Prep - by April 28th