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

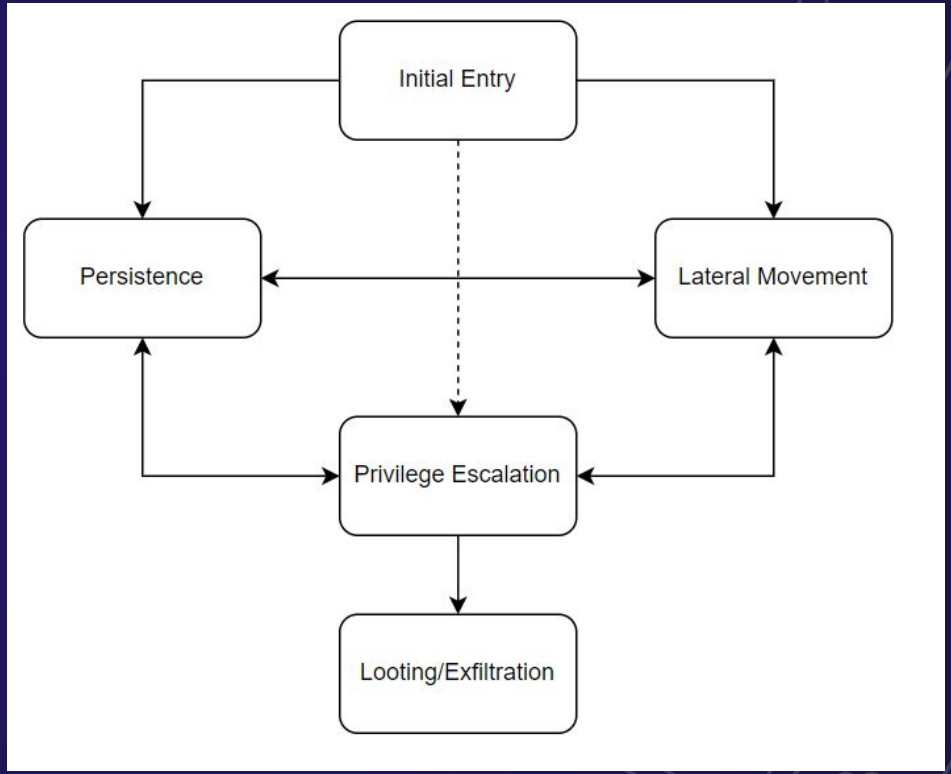
- Testing and Development
- Security
- Education

Design Complexity

- **Our Ground Zero**
  - What is AWS?
  - What makes up a Design Iteration?
  - Pivotal questions to deliver a valuable product
  - 2 Seperate Attack Paths

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



Project Requirements

- **Functional Requirements**
  - Incorporates common AWS-specific vulnerabilities and misconfigurations
  - Vulnerabilities should be actively exploitable
  - Logging and Monitoring to capture user and security events
  - Include an Incident Response Component that enables users to assess the impact
  - Attack path includes Non-Volatile persistence in the AWS account
  - Attack path should allow any user to fully compromise an AWS account (gain control over all aspects of the account through exploitation of given vulnerabilities)
- **Resource Constraints**
  - Utilize AWS CloudFormation for consolidated/static resource configuration and distribution
  - AWS API Gateway should be used as an interface between a user and other AWS resources
  - Utilize AWS Identity and Access Management for resource permissions
  - Cloud resource usage should be minimal, if not all in the free tier

Project Requirements

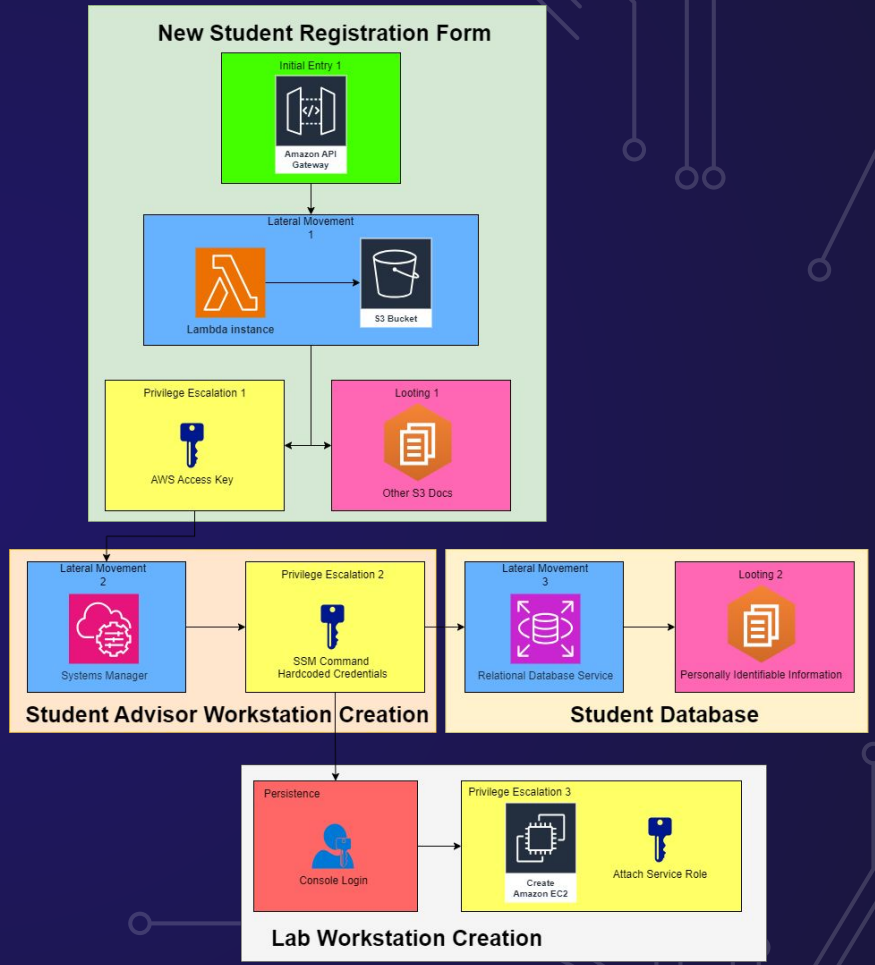
- **Qualitative Requirements**
  - Design a unique Service using existing AWS services and common configurations
  - Identity Management roles and policies should reflect professional roles and use cases
  - Documentation on the intended exploits and incident response components must be available to users
  - Implements safeguards to prevent unintended damage to AWS resources
  - Attack path follows the standard flow of a penetration test

Attack Path 1

Attack Path based on University network scenario

Common AWS services
- API Gateway
- Lambda Function
- EC2 Instances
- S3 Bucket
- RDS

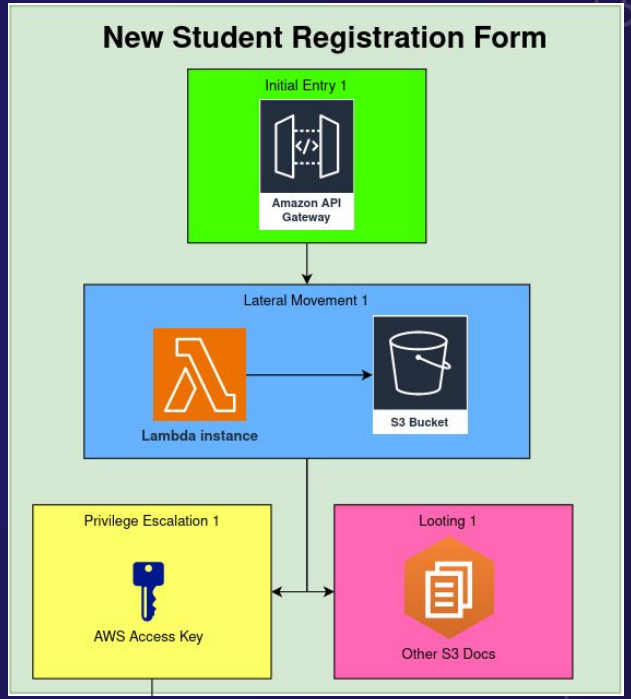
Common mistakes
- Reusing passwords across services
- Not deleting old services/tools
- Hardcoding passwords
- Incorrect read/write permissions



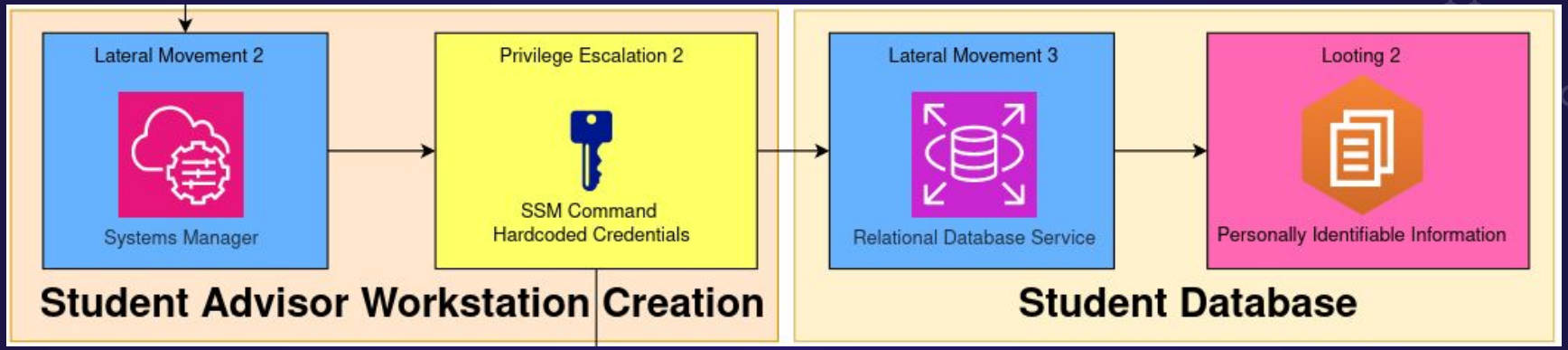
Attack Path 1

New Student Registration Form

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

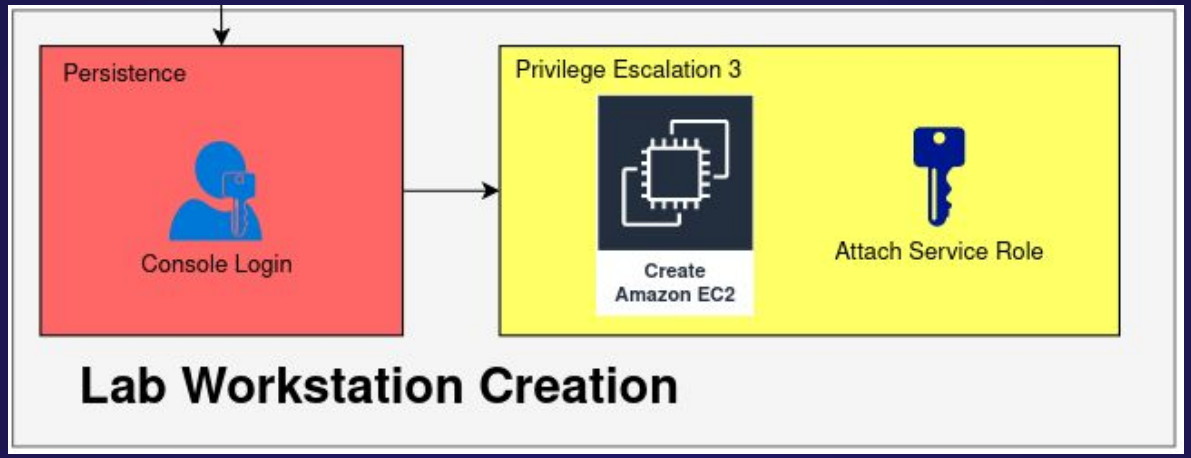


Attack Path 1



- 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.

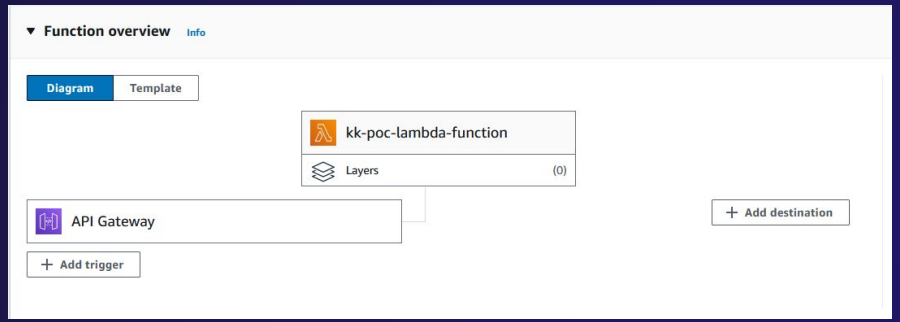
Attack Path 1



- 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

Prototype Implementations

Attack_Path_1_Intial_Entry:




```python
def lambda_handler(event, context):
    bucket_name = 'kk-poc'

    try:
        # Create an S3 client
        s3 = boto3.client('s3')

        response = s3.get_object(Bucket='kk-poc', Key='lambda-week7.txt')
        contents = response['Body'].read().decode('utf-8')
        print(contents)

        return {
            'statusCode': 200,
            'body': contents
        }
        return {
            'statusCode': 200,
            'body': json.dumps(response, default=str)
        }

    except Exception as e:
        print(f"Error: {e}")
        return {
            'statusCode': 500,
            'body': json.dumps({'message': 'Internal Server Error'})
        }
```

Prototype Implementations

Attack_Path_1_Priviliege_Escalation_2:

```
ashlerbenda@pop-os:~$ aws ssm list-commands
{
    "Commands": [
        {
            "CommandId": "7f7f7f7b-48eb-42ee-b90a-2fda11144bc2",
            "DocumentName": "OLD-get-student-files",
            "DocumentVersion": "2",
            "Comment": "",
            "ExpiresAfter": "2023-12-01T22:54:17.974000-06:00",
            "Parameters": {
                "pass": [
                    "password!"
                ],
                "user": [
                    "AshlerB"
                ]
            },
```

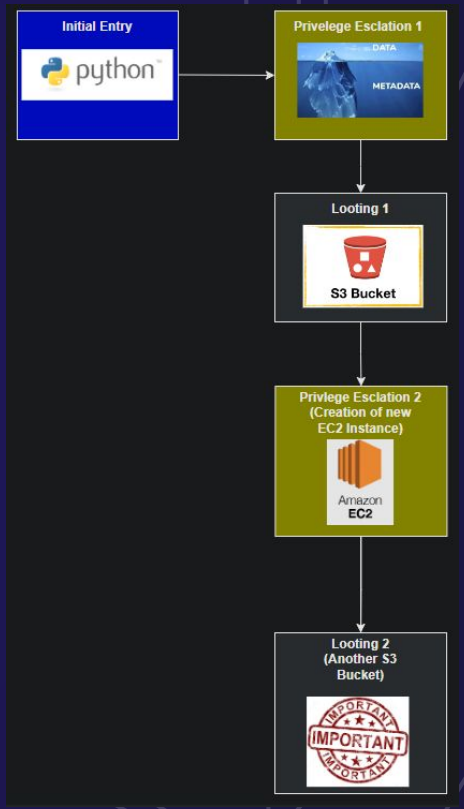
Final Design - Attack Path 2

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

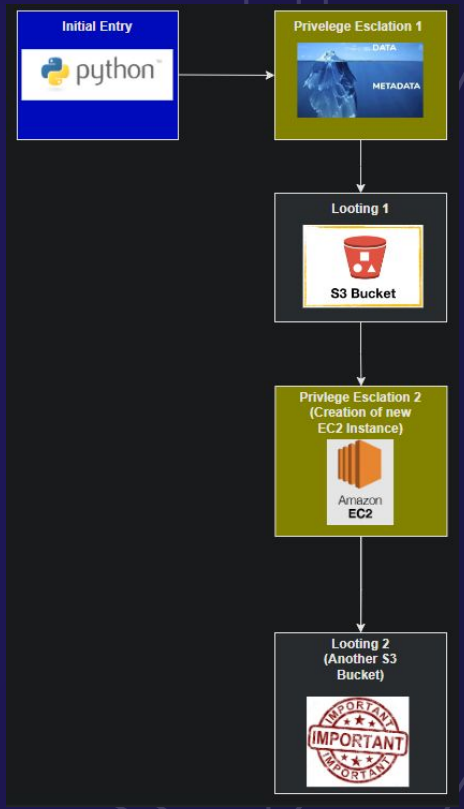
Initial Entry SSRF
- Utilizing an SSRF attack on an Django web server
- Attackers will then find an admin page

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



```
etd@EDSurface4:~$ aws iam list-attached-user-policies --user-name etd@iastate.edu
{
    "AttachedPolicies": [
        {
            "PolicyName": "IAMUserChangePassword",
            "PolicyArn": "arn:aws:iam::aws:policy/IAMUserChangePassword"
        }
    ]
}
etd@EDSurface4:~$ aws iam list-policy-versions --policy-arn arn:aws:iam::aws:policy/IAMUserChangePassword
{
    "Versions": [
        {
            "VersionId": "v2",
            "IsDefaultVersion": true,
            "CreateDate": "2016-11-15T23:18:55Z"
        },
        {
            "VersionId": "v1",
            "IsDefaultVersion": false,
            "CreateDate": "2016-11-15T00:25:16Z"
        }
    ]
}
```

```
etd@EDSurface4:~$ aws iam get-policy-version --policy-arn arn:aws:iam::aws:policy/IAMUserChangePassword --version-id v1
{
    "PolicyVersion": {
        "Document": {
            "Version": "2012-10-17",
            "Statement": [
                {
                    "Effect": "Allow",
                    "Action": [
                        "iam:ChangePassword"
                    ],
                    "Resource": [
                        "arn:aws:iam::*:user/${aws:userid}"
                    ]
                },
                {
                    "Effect": "Allow",
                    "Action": [
                        "iam:GetAccountPasswordPolicy"
                    ],
                    "Resource": "*"
                }
            ]
        },
        "VersionId": "v1",
        "IsDefaultVersion": false,
        "CreateDate": "2016-11-15T00:25:16Z"
    }
}
```

Project Plan

- Initial Set of Exploits Defined
  - Complete flAWS Level 1-2
  - Complete flAWS Level 3-4
  - Complete flAWS Level 5-6
  - Define an exploit that could be used in an attack path and where it could be used

- Attack Path Designed
  - Recon Defined
  - Initial Exploitation Defined
  - Persistence and Privilege Escalation Defined
  - Lateral Movement Defined
  - Looting (Exfiltration) defined
  - Define a use case for each AWS service/resource

Project Plan

- Remediation of Attack Path Defined
  - Vulnerability/misconfiguration remediation defined
  - Events and Actions defined

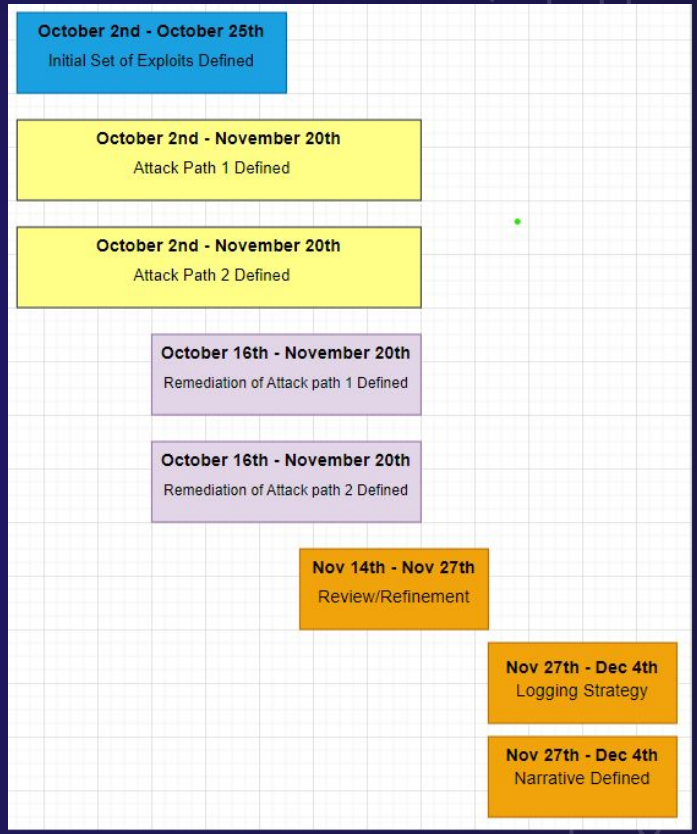
- Review/Refinement
  - Cross Review
  - Refinement

- Narrative Defined
  - Line of Business Defined
  - Use Case for Services Defined for Narrative
  - User Roles Defined

- Logging Strategies Defined

Project Timeline

- Metrics
  - Vulnerabilities per attack path
  - Number of resources used
  - Estimated Resource Monetary usage per hour
  - Complexity of setup
  - Realism



Testing

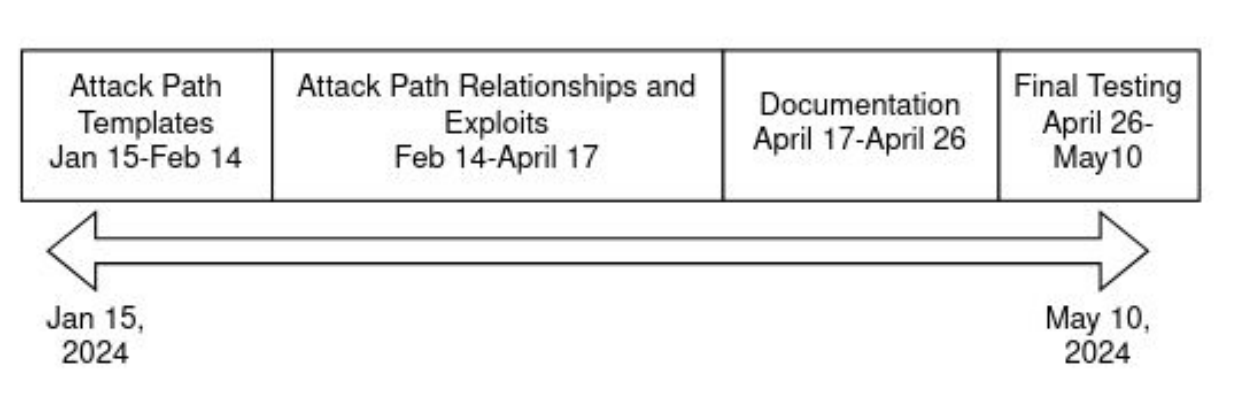
- Unit Testing
  - Run against each component in attack path
  - View state before attack, run attack, check state after
  - Primarily checked using logs

- 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

Conclusion

- Two completed designs for attack paths
- Implementation Plan
  - Create CloudFormation Templates
  - Connect individual AWS services and components together
  - Each member responsible for a part of the attack paths
- Work on learning CloudFormation Templates over winter break



Questions?