_# PRJ-CISA-049: Automated GRC Control Testing with AWS Config

Certification: Certified Information Systems Auditor (CISA) **Domain:** Domain 2 - Governance and Management of IT

---

1. Project Overview

This project bridges the gap between high-level GRC (Governance, Risk, and Compliance) frameworks and technical implementation. Auditors often test controls described in frameworks like NIST, COBIT, or ISO 27001. This project demonstrates how to take a specific control from such a framework and translate it into a fully automated, continuously monitored test using **AWS Config Rules**.

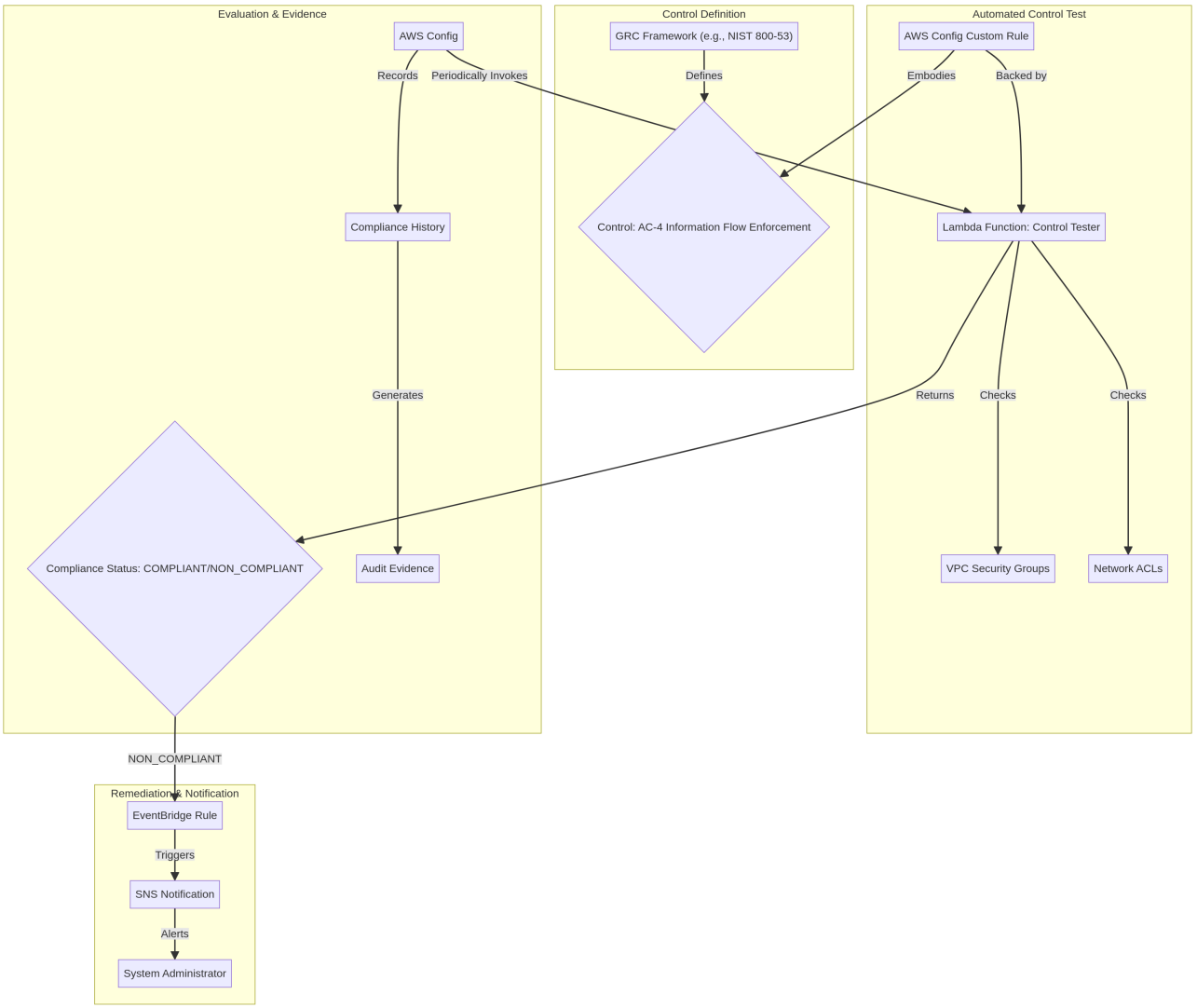
We will select a common technical control—for example, **NIST 800-53 AC-4: Information Flow Enforcement**, which mandates that systems enforce approved authorizations for controlling the flow of information. In a cloud context, this often translates to “unrestricted inbound traffic to sensitive ports (like SSH or RDP) should not be allowed.” We will create a **custom AWS Config rule**, backed by an **AWS Lambda function**, that automatically checks every security group in the account to see if it violates this policy. This transforms a manual, point-in-time audit test into a continuous, automated compliance verification mechanism.

Key Objectives

- Understand how to interpret a GRC control and translate it into a testable, technical requirement.
- Create a custom AWS Config rule using an AWS Lambda function.
- Write a Lambda function that programmatically inspects the configuration of AWS resources (in this case, Security Groups).
- Evaluate the resource against the defined control and report its compliance status (COMPLIANT or NON_COMPLIANT) back to AWS Config.
- Use the AWS Config dashboard to view the compliance status of all resources against your custom rule.
- Generate evidence for auditors that demonstrates the control is being continuously monitored.

2. Architecture

The architecture uses AWS Config as the engine to trigger and record the results of our custom, Lambda-powered control test.



Architectural Flow:

1. **Control Definition:**
    - A control is selected from a **GRC Framework** (e.g., NIST 800-53).
    - This control is translated into a specific, measurable rule: “No security group should allow inbound traffic from `0.0.0.0/0` on port 22 (SSH).”
2. **Automated Control Test:**

- An **AWS Config Custom Rule** is created. Instead of using a managed rule provided by AWS, this rule is configured to invoke a specific **AWS Lambda function**.
- The Lambda function (`Control-Tester`) contains the logic to perform the actual check.

3. **Evaluation and Evidence Generation:**
   - **AWS Config** acts as the trigger. Whenever a security group is created or changed (or on a periodic schedule), Config invokes the `Control-Tester` Lambda function.
   - It passes the details of the resource that needs to be evaluated to the Lambda function in the event payload.
   - The Lambda function inspects the IP permissions of the security group. If it finds a rule allowing `0.0.0.0/0` on port 22, it returns a `NON_COMPLIANT` status to AWS Config. Otherwise, it returns `COMPLIANT`.
   - AWS Config records this compliance status, creating a detailed **compliance history** for that resource. This history serves as powerful **audit evidence**.
4. **Remediation and Notification:**
   - When AWS Config receives a `NON_COMPLIANT` status, it generates an event.
   - An **EventBridge Rule** can be configured to listen for these specific non-compliant events.
   - The rule can then trigger an **SNS Notification** to alert the system administrator or security team that a misconfiguration has occurred, enabling rapid remediation.

---

3. Prerequisites

- An AWS account with administrative permissions.
- AWS Config enabled in your region.

---

4. Step-by-Step Implementation Guide

Step 4.1: Create the Lambda Function (The Control Tester)

1. **Create an IAM Role:**
    - Go to the **IAM Console -> Roles -> Create role**.
    - **Trusted entity:** AWS service -> Lambda.
    - **Permissions:** Attach the following policies:
        - `AWSLambdaBasicExecutionRole`
        - `ReadOnlyAccess` (for simplicity in this project; in production, you would scope this down to the specific `ec2:DescribeSecurityGroups` permission).
        - `AWSConfigRulesExecutionRole` (allows the function to send results back to AWS Config).
2. **Create the Lambda Function:**
    - Go to the **Lambda Console -> Create function**.
    - **Name:** `SecurityGroup-SSH-Checker`
    - **Runtime:** Python 3.9.
    - **Role:** Choose the IAM role you just created.
    - **Code:** Paste the following Python code.

```
import boto3
import json


def lambda_handler(event, context):
    invoking_event = json.loads(event['invokingEvent'])
    configuration_item = invoking_event['configurationItem']
    resource_id = configuration_item['resourceId']


    ec2 = boto3.client('ec2')
    config = boto3.client('config')


    # Check if the resource is a security group
    if configuration_item['resourceType'] !=
'AWS::EC2::SecurityGroup':
        return


    sg = ec2.describe_security_groups(GroupIds=[resource_id])
['SecurityGroups'][0]


    is_compliant = True
    # Check inbound rules for open SSH access
    for rule in sg.get('IpPermissions', []):
        if rule.get('FromPort') == 22 and rule.get('ToPort') == 22:
            for ip_range in rule.get('IpRanges', []):
                if ip_range.get('CidrIp') == '0.0.0.0/0':
                    is_compliant = False
                    break
        if not is_compliant:
            break


    compliance_status = 'COMPLIANT' if is_compliant else
'NON_COMPLIANT'


    # Send the evaluation back to AWS Config
    config.put_evaluations(
        Evaluations=[
            {
                'ComplianceResourceType':
```

```
configuration_item['resourceType'],
                'ComplianceResourceId': resource_id,
                'ComplianceType': compliance_status,
                'Annotation': 'Security group allows unrestricted SSH
access.' if not is_compliant else 'SSH access is restricted.',
                'OrderingTimestamp':
configuration_item['configurationItemCaptureTime']
            }
        ],
        ResultToken=event['resultToken']
    )
```

Step 4.2: Create the Custom AWS Config Rule

1. Go to the **AWS Config Console -> Rules -> Add rule**.
2. Select **Add custom rule**.
3. **Name:** `Unrestricted-SSH-Access-Check`
4. **Description:** "Checks for security groups that allow unrestricted inbound SSH traffic."
5. **AWS Lambda function ARN:** Select the `SecurityGroup-SSH-Checker` Lambda function.
6. **Trigger:**
   - **Trigger type:** Configuration changes.
   - **Scope of changes:** Resources.
   - **Resource type:** `AWS::EC2::SecurityGroup`.
7. Save the rule.

Step 4.3: Test the Rule

1. **Create a Non-Compliant Resource:**
   - Go to the **EC2 Console -> Security Groups -> Create security group**.
   - **Name:** `Test-NonCompliant-SG`
   - **Inbound rules:** Add a rule with:
     - **Type:** SSH
     - **Source:** Anywhere (`0.0.0.0/0`)

    - Create the security group.
2. **Evaluate Compliance:**
    - Go back to the **AWS Config Console -> Rules**.
    - Find your `Unrestricted-SSH-Access-Check` rule. It may take a few minutes to evaluate the new resource.
    - Click on the rule. You should see the `Test-NonCompliant-SG` listed as a **Non-compliant** resource.
3. **Create a Compliant Resource:**
    - Create another security group, but this time, set the source for the SSH rule to **My IP** or a specific, restricted IP range.
    - After a few minutes, AWS Config will evaluate this new security group and mark it as **Compliant**.

---

5. Evidence for the Auditor

As a CISA, you can now use the AWS Config console to demonstrate that the control is operating effectively:

- **Compliance Timeline:** Select a resource (like the non-compliant security group) and view its compliance timeline. This shows exactly when it was created, when it was evaluated, and its compliance status over time. This is a powerful piece of audit evidence.
- **Rule Details:** The custom rule itself, along with the Lambda code, serves as documentation for the automated test being performed.
- **Resource Inventory:** AWS Config provides a complete inventory of all security groups and their compliance status against this rule.

---

6. Cleanup

1. **Delete the test security groups** (`Test-NonCompliant-SG` and the compliant one).
2. Go to the **AWS Config Console**, select your custom rule, and **delete** it.

3. **Delete the Lambda function** (`SecurityGroup-SSH-Checker`).
4. **Delete the IAM role** created for the Lambda function.

Business Context

The Problem

Organizations need to demonstrate compliance with audit requirements but lack automated evidence collection. Manual audit preparation is time-consuming and error-prone. Auditors struggle to verify security controls and compliance in dynamic cloud environments.

The Solution

Automated compliance monitoring and audit evidence collection system. Continuously assesses AWS environment against compliance frameworks, generates audit reports, and maintains evidence repository. Provides real-time compliance dashboards and automated remediation for non-compliant resources.

Business Value

- **Audit Efficiency:** Reduces audit preparation time from months to days
- **Continuous Compliance:** Real-time monitoring vs. point-in-time assessments
- **Cost Reduction:** Eliminates manual evidence collection, saving 200+ hours per audit
- **Risk Visibility:** Executive dashboards show compliance posture at a glance

Risk Mitigation

Prevents compliance violations, identifies control gaps before audits, ensures evidence availability, and reduces audit findings and remediation costs.

GRC Mapping

Compliance Frameworks

- **COBIT 2019:** APO13 (Manage security), DSS05 (Manage security services)
- **ISO 27001:** A.18.1 (Compliance with legal requirements), A.18.2 (Information security reviews)
- **NIST CSF:** ID.GV-3 (Legal and regulatory requirements), PR.IP-1 (Baseline configuration)
- **ITIL v4:** Service validation and testing

Security Controls Implemented

- Automated compliance assessments
- Continuous control monitoring
- Audit trail and evidence collection
- Configuration compliance checking
- Automated remediation workflows

Audit Evidence

- Compliance assessment reports
- Control effectiveness evidence
- Configuration snapshots and change logs
- Remediation records and timelines

Regulatory Alignment

- **SOX:** Section 404 (Internal controls assessment)
- **PCI DSS:** Requirement 11.3 (Penetration testing), Requirement 12.9 (Service provider compliance)
- **HIPAA:** § 164.308(a)(8) (Evaluation of security measures)
- **SOC 2:** All trust service criteria