AWS Bedrock Guardrails for SecurityToolsSandbox

This repository contains a CloudFormation template that implements comprehensive security guardrails for AWS Bedrock in the SecurityToolsSandbox environment.

Overview

AWS Bedrock provides access to powerful foundation models, but these models require proper guardrails to ensure secure, cost-effective, and responsible use. This template implements multiple layers of protection to safeguard your Bedrock usage.

Architecture

Show Image

The solution integrates with your existing VPC infrastructure while adding multiple security layers:

1. **IAM Access Controls** - Principle of least privilege
2. **Network Isolation** - VPC endpoints for private access
3. **Content Filtering** - Prevents harmful content generation
4. **Cost Controls** - Budget alerts and usage monitoring
5. **Human Oversight** - Workflows for reviewing edge cases
6. **Approved Prompts** - Storage and management of vetted prompts

Components

1. IAM Controls

The template creates a role with least privilege access to Bedrock:

- **BedrockAccessRole**: Role with restricted permissions
- **BedrockRestrictedPolicy**: Policy limiting access to specific models and requiring environment tags

These controls ensure that only authorized services can access specific foundation models and prevents uncontrolled usage.

2. Network Isolation

The template configures private network access to Bedrock:

- **BedrockSecurityGroup**: Restricts network traffic to Bedrock
- **BedrockVPCEndpoint**: Enables private communication with Bedrock (never traverses the public internet)

This configuration uses your existing VPC and subnet infrastructure to establish secure communication channels.

3. Content Filtering

A custom resource implements Bedrock guardrails for content safety:

- **Hate Speech Filtering**: Blocks content that may contain hate speech
- **Sexual Content Filtering**: Prevents generation of explicit content
- **Violence Filtering**: Blocks violent content generation
- **PII Protection**: Masks or blocks personally identifiable information
- **Custom Word Lists**: Blocks organization-specific restricted terms

These controls help ensure that the generated content adheres to organizational policies and ethical standards.

4. Monitoring and Logging

Comprehensive monitoring enables visibility and auditability:

- **BedrockLogGroup**: Centralized logging for model invocations
- **BedrockTrail**: CloudTrail configuration for API auditing
- **BedrockCostAlarm**: Alerts for unusual usage patterns

These components provide a full audit trail and early warning system for potential issues.

5. Cost Management

Budget controls prevent unexpected expenses:

- **BedrockBudget**: Monthly budget with alerts at 80% threshold

These controls help maintain predictable spending on Bedrock services.

6. Human Oversight

A mechanism for human review of edge cases:

- **EdgeCaseReviewFunction**: Lambda function for detecting and routing edge cases
- **BedrockAlertsTopic**: SNS topic for human notification

This workflow ensures that low-confidence or potentially problematic model responses can be reviewed by humans.

7. Prompt Management

Infrastructure for managing approved prompts:

- **ApprovedPromptsTable**: DynamoDB table for storing vetted prompts
- **SafePromptTemplates**: SSM Parameter Store for secure template storage

These components help standardize interactions with foundation models and enforce consistent, safe usage patterns.

Deployment

Prerequisites

- AWS CLI configured with appropriate permissions
- Existing VPC and subnet IDs

Deployment Steps

1. Clone this repository
2. Update parameter values in the template if needed
3. Deploy using AWS CloudFormation:

bash

```bash
aws cloudformation create-stack \
  --stack-name bedrock-guardrails-securitytoolssandbox \
  --template-body file://bedrock-guardrails.yaml \
  --parameters \
    ParameterKey=AdminEmail,ParameterValue=your-email@example.com \
    ParameterKey=MonthlyBudgetAmount,ParameterValue=100 \
    ParameterKey=ExistingVpcId,ParameterValue=vpc-xxxxxxxx \
    ParameterKey=ExistingSubnetId,ParameterValue=subnet-xxxxxxxx \
  --capabilities CAPABILITY_NAMED_IAM
```

Usage

After deployment, your Bedrock usage will be secured by these guardrails. To use Bedrock:

1. Applications should assume the `BedrockAccessRole`
2. Ensure your application is running in the specified subnet
3. Use the approved prompt templates from SSM Parameter Store
4. Check CloudWatch logs and metrics for usage patterns

Security Considerations

- The IAM role adheres to least privilege principles

- All communication with Bedrock occurs through private VPC endpoints
- Content filtering prevents generation of harmful content
- PII handling prevents leakage of sensitive information
- Cost controls prevent unexpected expenses
- Human oversight enables review of edge cases

Customization

You can customize this solution by:

1. Modifying the content filtering thresholds in the GuardrailsLambda code
2. Adding additional custom word lists for your specific use case
3. Adjusting the budget amount and notification thresholds
4. Adding additional CloudWatch alarms for specific monitoring needs

Troubleshooting

Common issues and their solutions:

1. **Deployment Failure**: Ensure your IAM user has sufficient permissions
2. **VPC Endpoint Issues**: Verify subnet routings and security group settings
3. **GuardrailsLambda Failures**: Check CloudWatch logs for the Lambda function
4. **Model Access Denied**: Verify the IAM policy includes the required model ARNs

Maintenance

Regular maintenance tasks:

1. Review CloudWatch logs for unusual patterns
2. Verify CloudTrail is capturing all relevant events
3. Update approved prompts as needed
4. Adjust content filtering thresholds based on experience
5. Review and adjust budget as usage patterns evolve

Security Best Practices

This implementation follows AWS security best practices:

1. **Principle of Least Privilege**: IAM roles with minimal required permissions
2. **Defense in Depth**: Multiple layers of security controls
3. **Encryption in Transit**: All communication over encrypted channels
4. **Auditing**: Comprehensive logging and monitoring
5. **Cost Control**: Budget limits and alerts

References

1. AWS Bedrock Documentation
2. AWS CloudFormation User Guide
3. AWS Security Best Practices