

# 사용자 가이드

# **Amazon Verified Permissions**



Copyright © 2024 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

# Amazon Verified Permissions: 사용자 가이드

Copyright © 2024 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon의 상표 및 브랜드 디자인은 Amazon 외 제품 또는 서비스와 관련하여 고객에게 혼동을 일으킬수 있는 방식이나 Amazon 브랜드 이미지를 떨어뜨리는 방식으로 사용할 수 없습니다. Amazon이 소유하지 않은 기타 모든 상표는 Amazon과 제휴 관계이거나 관련이 있거나 후원 관계와 관계없이 해당 소유자의 자산입니다.

# **Table of Contents**

Amazon Verified Permissions란 무엇인가요?	1
Amazon Verified Permissions의 권한 부여	1
Cedar 정책 언어	1
Verified Permissions의 이점	2
애플리케이션 개발 가속화	
애플리케이션 보안 강화	2
최종 사용자 기능	
관련 서비스	2
Verified Permissions에 액세스	3
Verified Permissions 요금	4
시작하기	
가입하기 AWS 계정	6
관리자 액세스 권한이 있는 사용자 생성	6
	7
첫 번째 정책 스토어 생성	8
샘플 정책 스토어 생성	8
샘플 정책 스토어용 템플릿 연결 정책 생성	9
샘플 정책 스토어 테스트	10
API연결된 정책 저장소 생성	13
권한 부여 모델 설계	14
하나의 올바른 모델이란 없습니다	15
리소스에 집중하세요	15
복합 권한 부여	17
멀티테넌시 고려	18
공유 정책 저장소와 테넌트별 정책 저장소 비교	19
선택 방법	20
정책 범위를 채우는 것이 좋습니다	20
모든 리소스는 컨테이너 안에 있어야 합니다	21
보안 주체를 리소스와 분리하세요	23
관계 표현하기	25
속성 기반 관계	25
템플릿 기반 관계	27
세분화된 권한	29
권한 부여 쿼리가 필요한 다른 이유	30

정책 스토어	. 31
정책 스토어 생성	31
API-연결된 정책 저장소	38
작동 방식	40
고려 사항	41
추가 ABAC	42
프로덕션으로 이동	43
문제 해결	46
정책 스토어 삭제	48
정책 스토어 스키마	50
스키마 편집 - 비주얼	52
스키마 편집 - JSON	54
정책 검증 모드	55
정책	57
엔터티 형식	58
IAM 정책 생성	62
정적 정책 편집	64
테스트 정책	
정책 예제	69
개별 엔터티에 대해 액세스 허용하기	69
엔터티 그룹에 대해 액세스 허용하기	
모든 엔터티에 대해 액세스 허용하기	71
엔티티의 속성에 대한 액세스를 허용합니다 (ABAC)	71
액세스 거부	74
대괄호 표기법을 사용하여 토큰 속성을 참조합니다	
점 표기법을 사용하여 속성을 참조합니다	
아마존 Cognito ID 토큰 속성을 반영합니다	
OIDCID 토큰 속성 반영	
Amazon Cognito 액세스 토큰 속성을 반영합니다	
OIDC액세스 토큰 속성을 반영합니다	
정책 템플릿 및 템플릿 연결 정책	
정책 템플릿 생성	
템플릿 연결 정책 생성	
정책 템플릿 편집	
템플릿 연결 정책의 예	
PhotoFlash예제	83

DigitalPetStore 예:	85
TinyToDo예:	85
자격 증명 공급자	86
Amazon Cognito 자격 증명 소스를 사용한 작업	86
자격 증명 소스 사용 OIDC	88
클라이언트 및 잠재고객 검증	89
클라이언트 측 권한 부여: JWTs	91
자격 증명 소스 생성	93
아마존 Cognito 자격 증명 소스	94
OIDCID 소스	96
자격 증명 소스 편집	98
Amazon Cognito 사용자 풀 자격 증명 소스	99
OpenID Connect (OIDC) 아이덴티티 소스	101
토큰을 스키마에 매핑	102
스키마 매핑에 대해 알아야 할 사항	103
ID 토큰 매핑	106
액세스 토큰 매핑	110
Amazon Cognito 콜론으로 구분된 클레임의 대체 표기법	114
요청 승인	117
API오퍼레이션	118
테스트 모델	119
애플리케이션과의 통합	120
	123
예제 컨텍스트 평가	125
보안	131
데이터 보호	131
데이터 암호화	
Identity and Access Management	133
고객	133
ID를 통한 인증	
정책을 사용한 액세스 관리	137
Amazon 검증 권한은 다음과 함께 작동하는 방식 IAM	139
IAM 검증된 권한에 대한 정책	145
자격 증명 기반 정책 예시	
문제 해결	
규정 준수 확인	151

복원력	152
모니터링	153
CloudTrail 로그	153
검증된 권한 정보: CloudTrail	153
Verified Permissions 로그 파일 항목 이해	154
함께 일하기 AWS CloudFormation	172
검증된 권한 및 템플릿 AWS CloudFormation	172
AWS CDK 구성	173
에 대해 자세히 알아보십시오. AWS CloudFormation	173
사용 AWS PrivateLink	174
고려 사항	174
인터페이스 엔드포인트 생성	174
할당량	176
리소스 할당량	176
계층 구조 할당량	178
초당 작업 할당량	179
용어 및 개념	182
권한 부여 모델	183
권한 부여 요청	183
권한 부여 응답	183
고려 대상 정책	183
컨텍스트 데이터	183
결정적 정책	184
엔터티 데이터	184
권한, 권한 부여 및 보안 주체	184
정책 적용	184
정책 스토어	184
충족 대상 정책	185
Cedar와의 차이점	185
네임스페이스 정의	185
정책 템플릿 지원	185
스키마 지원	186
익스텐션 유형 지원	186
엔티티의 시더 JSON 형식	186
작업 그룹 정의	186
길이 및 크기 제한	187

<del>-</del> 188	설명서 기록	사용
clxxxix		

# Amazon Verified Permissions란 무엇인가요?

Amazon Verified Permissions는 사용자가 빌드한 사용자 지정 애플리케이션을 위한 확장 가능하고 세분화된 권한 관리 및 권한 부여 서비스입니다. Verified Permissions를 사용하면 외부에서 권한을 부여하고 중앙에서 정책을 관리하고 운영하여 개발자가 안전한 애플리케이션을 더 빠르게 빌드할 수 있습니다. Verified Permissions는 Cedar 정책 언어를 사용하여 애플리케이션 사용자에 대한 세분화된 권한을 정의합니다.

#### 주제

- Amazon Verified Permissions의 권한 부여
- Cedar 정책 언어
- Verified Permissions의 이점
- 관련 서비스
- Verified Permissions에 액세스
- Verified Permissions 요금

### Amazon Verified Permissions의 권한 부여

Verified Permissions는 사용자 지정 애플리케이션의 지정된 컨텍스트에서 보안 주체가 리소스에 대한 작업을 수행할 수 있는지 여부를 확인하여 권한을 부여합니다. Verified Permissions는 OpenID Connect와 같은 프로토콜, Amazon Cognito와 같은 호스팅 공급자 또는 다른 인증 솔루션 등 다른 수단을 사용하여 보안 주체가 이전에 식별되고 인증된 것으로 가정합니다. Verified Permissions는 사용자가 관리되는 위치 및 사용자 인증 방법에 구애받지 않습니다.

Verified Permissions는 고객이 AWS Management Console에서 정책을 작성, 유지 관리 및 테스트할수 있도록 하는 서비스입니다. 권한은 Cedar 정책 언어를 사용하여 표현됩니다. 클라이언트 애플리케이션은 권한을 APIs 요청하여 서비스에 저장된 Cedar 정책을 평가하고 조치 허용 여부에 대한 액세스 결정을 내립니다.

## Cedar 정책 언어

Verified Permissions의 권한 부여 정책은 Cedar 정책 언어를 사용하여 작성됩니다. Cedar는 권한 부여 정책을 작성하고 해당 정책을 기반으로 권한 부여 결정을 내리는 데 사용되는 오픈 소스 언어입니다. 애플리케이션을 만들 때는 승인된 사용자만 애플리케이션에 액세스할 수 있고 각 사용자가 권한을 부여받은 작업만 수행할 수 있도록 해야 합니다. Cedar를 사용하면 비즈니스 로직을 권한 부여 로직에

서 분리할 수 있습니다. 애플리케이션 코드에서는 작업에 대한 요청 앞에 Cedar 권한 부여 엔진을 호출하여 "이 요청에 대한 권한이 부여되었습니까?"라고 묻습니다. 그러면 애플리케이션은 해당 결정이 "허용"이면 요청된 작업을 수행하고 "거부"인 경우 오류 메시지를 반환할 수 있습니다.

검증된 권한은 현재 Cedar 버전 2.4를 사용합니다.

Cedar에 대한 자세한 내용은 다음을 참조하세요.

- Cedar 정책 언어 참조 가이드
- 시더 리포지토리 GitHub

## Verified Permissions의 이점

### 애플리케이션 개발 가속화

권한 부여를 비즈니스 로직과 분리하여 애플리케이션 개발을 가속화합니다.

### 애플리케이션 보안 강화

Verified Permissions를 통해 개발자는 더 안전한 애플리케이션을 빌드할 수 있습니다.

### 최종 사용자 기능

Verified Permissions를 사용하면 권한 관리를 위해 보다 풍부한 최종 사용자 기능을 제공할 수 있습니다.

# 관련 서비스

- Amazon Cognito Amazon Cognito는 웹 및 모바일 앱을 위한 자격 증명 플랫폼입니다. 사용자 디렉토리, 인증 서버, OAuth 2.0 액세스 토큰 및 AWS 자격 증명을 위한 권한 부여 서비스입니다. 정책 스토어를 생성할 때 Amazon Cognito 사용자 풀에서 보안 주체와 그룹을 구축할 수 있습니다. 자세한 정보는 Amazon Cognito 개발자 안내서를 참조하세요.
- Amazon API Gateway Amazon API Gateway는 규모에 상관없이 생성, 게시, 유지 관리 RESTHTTP, 모니터링 및 WebSocket APIs 보안을 위한 AWS 서비스입니다. 정책 저장소를 생성할 때 내부 API 게이트웨이에서 작업과 리소스를 구축할 수 있는 API 옵션이 제공됩니다. API게이트웨 이에 대한 자세한 내용은 API게이트웨이 개발자 안내서를 참조하십시오.
- AWS IAM Identity Center— IAM Identity Center를 사용하면 직원 ID (직원 사용자라고도 함) 의 로그 인 보안을 관리할 수 있습니다. IAMIdentity Center는 직원 사용자를 생성하거나 연결하고 모든 직원

Verified Permissions의 이점 2

및 애플리케이션에서 직원 액세스를 중앙에서 관리할 수 있는 한 곳을 제공합니다. AWS 계정 자세한 내용은 AWS IAM Identity Center 사용 설명서를 참조하십시오.

# Verified Permissions에 액세스

다음 방법 중 하나를 사용하여 Amazon Verified Permissions에서 작업할 수 있습니다.

#### **AWS Management Console**

콘솔은 Verified Permissions 및 AWS 리소스를 관리하기 위한 브라우저 기반 인터페이스입니다. 콘솔을 통하여 Verified Permissions에 액세스하는 방법에 대한 자세한 내용은 AWS 로그인 사용 설명서의 AWS로그인 방법을 참조하세요.

• Amazon 검증 권한 콘솔

#### AWS 명령줄 도구

AWS 명령줄 도구를 사용하여 시스템 명령줄에서 명령을 실행하여 검증된 권한 및 AWS 작업을 수행할 수 있습니다. 명령줄을 사용하는 것이 콘솔을 사용하는 것보다 더 빠르고 편리할 수 있습니다. AWS 작업을 수행하는 스크립트를 작성할 때도 명령줄 도구가 유용합니다.

AWS 두 개의 명령줄 도구 세트, 즉 <u>AWS Command Line Interface</u>(AWS CLI) 와 를 제공합니다 <u>AWS Tools for Windows PowerShell</u>. 설치 및 사용에 대한 자세한 내용은 사용 <u>AWS Command Line Interface 설명서를</u> 참조하십시오. AWS CLI PowerShellWindows용 도구 설치 및 사용에 대한 자세한 내용은 사용 AWS Tools for Windows PowerShell 설명서를 참조하십시오.

- 명령 참조의 검증된 권한 AWS CLI
- Amazon에서 검증한 권한 AWS Tools for Windows PowerShell

#### **AWS SDKs**

AWS 다양한 프로그래밍 언어 및 플랫폼 SDKs (Java, Python, Ruby 등) 의 라이브러리와 샘플 코드로 구성된 (소프트웨어 개발 키트) 를 제공합니다. NET, iOS, 안드로이드 등). 검증된 권한 및 AWS에 대한 프로그래밍 방식 액세스를 생성할 수 있는 편리한 방법을 SDKs 제공합니다. 예를 들어 요청에 암호로 서명하고, 오류를 관리하고, 요청을 자동으로 재시도하는 등의 작업을 처리합니다. SDKs

자세히 알아보고 AWS SDKs 다운로드하려면 도구 용을 참조하십시오. Amazon Web Services

다음은 다양한 버전의 검증된 권한 리소스에 대한 설명서로 연결되는 AWS SDKs 링크입니다.

AWS SDK for .NET

Verified Permissions에 액세스

- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java
- AWS SDK for JavaScript
- · AWS SDK for PHP
- AWS SDK for Python (Boto)
- AWS SDK for Ruby

#### AWS CDK 구성

코드로 클라우드 인프라를 정의하고 이를 통해 프로비저닝하기 위한 오픈 소스 소프트웨어 개발 AWS Cloud Development Kit (AWS CDK) 프레임워크입니다. AWS CloudFormation구성 또는 재사용 가능한 클라우드 구성 요소를 사용하여 템플릿을 만들 수 있습니다. AWS CloudFormation 그런다음 이러한 템플릿을 사용하여 클라우드 인프라를 배포할 수 있습니다.

자세히 알아보고 다운로드하려면 AWS Cloud AWS CDKs Development Kit를 참조하십시오.

다음은 구문과 같은 검증된 권한 AWS CDK 리소스에 대한 설명서로 연결되는 링크입니다.

• 아마존 검증 권한 L2 구성 CDK

#### 검증된 권한 API

서비스에 직접 HTTPS 요청을 보낼 수 있는 검증된 권한을 API 사용하여 AWS 프로그래밍 방식으로 검증된 권한에 액세스할 수 있습니다. 를 사용할 때는 자격 API 증명을 사용하여 요청에 디지털 서명하는 코드를 포함해야 합니다.

• Amazon 검증 권한 API 참조 가이드

### Verified Permissions 요금

Verified Permissions는 애플리케이션이 Verified Permissions에 요청한 월별 권한 부여 요청 수를 기준으로 단계별 요금을 제공합니다. 애플리케이션이 검증된 권한에 대해 수행한 월별 c URL (클라이언트 URL) 정책 API 요청량을 기준으로 정책 관리 작업에 대한 요금도 부과됩니다.

Verified Permissions의 요금 및 가격을 안내하는 전체 목록은 <u>Amazon Verified Permissions 요금</u>을 참 조하세요.

청구 요금은 AWS Billing and Cost Management 콘솔의 청구 및 비용 관리 대시보드에서 확인할 수 있습니다. 청구서에는 요금 내역을 자세하게 확인할 수 있는 사용 보고서 링크가 포함됩니다. AWS 계정청구에 대한 자세한 내용은 AWS Billing 사용 설명서를 참조하십시오.

Verified Permissions 요금 4

AWS 청구, 계정, 이벤트에 관한 질문이 있는 경우 <u>문의하세요 AWS Support</u>.

Verified Permissions 요금 5

# Amazon 검증 권한 시작하기

검증된 권한을 시작하려면 다음이 필요합니다. AWS 검증된 권한에서 리소스를 생성할 권한이 있는 계정 및 IAM Identity Center 사용자

다음 섹션은 계정을 만드는 데 도움이 될 것입니다. AWS 계정 및 필요한 사용자:

## 가입하기 AWS 계정

가지고 있지 않은 경우 AWS 계정다음 단계를 완료하여 새로 만드세요.

가입하려면 AWS 계정

- 1. https://portal.aws.amazon.com/billing/등록 열기.
- 2. 온라인 지시 사항을 따릅니다.

등록 절차 중 전화를 받고 전화 키패드로 확인 코드를 입력하는 과정이 있습니다.

가입할 때 AWS 계정, 그리고 AWS 계정 루트 사용자생성됩니다. 루트 사용자는 모두에 액세스할 수 있습니다. AWS 서비스 및 계정 내 리소스. 보안 모범 사례는 사용자에게 관리 액세스 권한을 할 당하고, 루트 사용자만 사용하여 루트 사용자 액세스 권한이 필요한 작업을 수행하는 것입니다.

AWS 가입 절차가 완료된 후 확인 이메일을 보냅니다. 언제라도 <a href="https://aws.amazon.com/로">https://aws.amazon.com/로</a> 이동하여 내 계정을 선택하여 현재 계정 활동을 확인하고 계정을 관리할 수 있습니다.

## 관리자 액세스 권한이 있는 사용자 생성

가입한 후 AWS 계정보안을 유지하세요 AWS 계정 루트 사용자, 활성화 AWS IAM Identity Center일상적인 작업에 루트 사용자를 사용하지 않도록 관리 사용자를 생성하십시오.

보안을 유지하세요. AWS 계정 루트 사용자

1. <u>에 로그인하기AWS Management Console</u>루트 사용자를 선택하고 다음을 입력하여 계정 소유자로 등록하십시오. AWS 계정 이메일 주소. 다음 페이지에서 비밀번호를 입력합니다.

루트 사용자로 로그인하는 데 도움이 <u>필요하면 에서 루트 사용자로 로그인을</u> 참조하십시오. AWS 로그인 사용 설명서.

2. 루트 사용자에 대한 다단계 인증 (MFA) 을 켜십시오.

기입하기 AWS 계정 6

지침은 다음을 위한 가상 MFA 장치 <u>활성화를 참조하십시오. AWS 계정</u>의 루트 사용자 (콘솔) IAM 사용 설명서.

관리자 액세스 권한이 있는 사용자 생성

1. IAMID 센터를 활성화합니다.

지침은 <u>활성화를 참조하십시오</u>. AWS IAM Identity Center의AWS IAM Identity Center 사용 설명서.

2. IAMID 센터에서 사용자에게 관리 액세스 권한을 부여하십시오.

사용에 대한 자습서는 IAM Identity Center 디렉터리 ID 소스로 사용하려면 기본값으로 사용자 액세스 <u>구성을 참조하십시오</u>. IAM Identity Center 디렉터리의AWS IAM Identity Center 사용자 가이드

관리 액세스 권한이 있는 사용자로 로그인

• IAMIdentity Center 사용자로 로그인하려면 IAM Identity Center 사용자를 생성할 때 이메일 주소로 전송된 URL 로그인을 사용하십시오.

IAMIdentity Center 사용자를 사용하여 로그인하는 데 도움이 <u>필요하면 로그인을 참조하십시오.</u> AWS포털에 접속할 수 있습니다. AWS 로그인 사용자 가이드.

추가 사용자에게 액세스 권한 할당

1. IAMIdentity Center에서 최소 권한 권한 적용 모범 사례를 따르는 권한 집합을 생성하십시오.

지침은 에서 권한 집합 만들기를 참조하십시오. AWS IAM Identity Center 사용 설명서.

2. 사용자를 그룹에 할당하고, 그룹에 Single Sign-On 액세스 권한을 할당합니다.

자세한 지침은 <u>그룹 추가를</u> 참조하십시오. AWS IAM Identity Center 사용 설명서.

이제 만들었으니 AWS 계정 및 일부 사용자 여러분, 이제 정책 저장소를 생성할 준비가 되었습니다. 다음 중 하나를 선택하여 검증된 권한을 시작하십시오.

• 첫 번째 Amazon 검증 권한 정책 스토어 생성

• ID 공급자와 함께 API Gateway를 사용하기 위한 정책 저장소 생성

# 첫 번째 Amazon 검증 권한 정책 스토어 생성

Verified Permissions 콘솔에 처음 로그인할 때 첫 번째 <u>정책 스토어</u>와 Cedar 정책을 생성하는 방법을 선택할 수 있습니다. AWS 로그인 사용 설명서의 <u>AWS에 로그인하는 방법</u> 항목에 설명된 대로 사용자유형에 맞는 로그인 절차를 따르세요. 콘솔 홈 페이지에서 Amazon Verified Permissions 서비스를 선택합니다. Get started를 선택합니다.

### 샘플 정책 스토어 생성

Verified Permissions를 처음 사용하는 경우 샘플 정책 스토어 중 하나를 사용하여 Verified Permissions 의 작동 방식을 익히는 것이 좋습니다. 샘플 정책 스토어는 사전 정의된 정책과 스키마를 제공합니다.

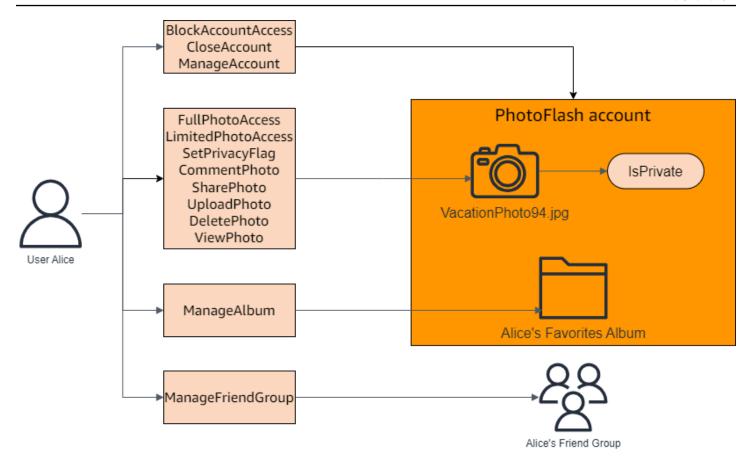
샘플 정책 스토어 구성 방법을 사용하여 정책 스토어를 생성하려면

- 1. 검증된 권한 콘솔에서 새 정책 저장소 생성을 선택합니다.
- 2. 시작 옵션 섹션에서 샘플 정책 저장소를 선택합니다.
- 샘플 프로젝트 섹션에서 사용할 샘플 Verified Permissions 애플리케이션의 유형을 선택합니다. 이 자습서에서는 PhotoFlash정책 저장소를 선택합니다.
- 4. 샘플 정책 스토어의 스키마 네임스페이스는 선택한 샘플 프로젝트에 따라 자동으로 생성됩니다.
- 5. 정책 스토어 생성을 선택합니다.

정책. 정책 템플릿 및 샘플 정책 스토어용 스키마를 사용하여 정책 스토어가 생성됩니다.

아래 다이어그램은 PhotoFlash 샘플 정책 저장소 작업과 해당 작업이 적용되는 리소스 유형 간의 관계를 보여줍니다.

첫 번째 정책 스토어 생성



### 샘플 정책 스토어용 템플릿 연결 정책 생성

PhotoFlash 샘플 정책 저장소에는 정책, 정책 템플릿 및 스키마가 포함됩니다. 샘플 정책 스토어에 포함된 정책 템플릿을 기반으로 템플릿 연결 정책을 생성할 수 있습니다.

샘플 정책 스토어용 템플릿 연결 정책을 생성하려면

- 1. 에서 검증된 권한 콘솔을 엽니다 <u>https://console.aws.amazon.com/verifiedpermissions/</u>. 정책 저장 소를 선택합니다.
- 2. 왼쪽 탐색 창에서 정책을 선택합니다.
- 3. 정책 생성을 선택한 다음 템플릿 연결 정책 생성을 선택합니다.
- 4. 정책 템플릿 옆의 비공개로 공유한 사진에 대한 전체 액세스 권한 부여라는 설명이 있는 라디오 버튼을 선택하고 다음을 선택합니다.
- 5. [교장 주체] 에는 을 입력합니다PhotoFlash::User::"Alice". 리소스에 을 입력합니다PhotoFlash::Album::"Bob-Vacation-Album".
- 6. 템플릿 연결 정책 생성을 선택합니다.

새 템플릿 연결 정책이 정책 아래에 표시됩니다.

7. PhotoFlash 샘플 정책 저장소에 사용할 다른 템플릿 연결 정책을 생성합니다. 정책 생성을 선택한다음 템플릿 연결 정책 생성을 선택합니다.

- 8. 비공개가 아닌 공유 사진에 대한 제한된 액세스 허용이라는 설명이 있는 정책 템플릿 옆의 라디오 버튼을 선택한 후 다음을 선택합니다.
- 9. [교장 주체] 에는 을 입력합니다PhotoFlash::FriendGroup::"MySchoolFriends". 리소스에 을 입력합니다PhotoFlash::Album::"Alice's favorite album".
- 10. 템플릿 연결 정책 생성을 선택합니다.

새 템플릿 연결 정책이 정책 아래에 표시됩니다.

새 템플릿 연결 정책은 튜토리얼의 다음 단원에서 테스트할 수 있습니다. 템플릿 연결 정책을 만드는데 사용할 수 있는 값의 예를 더 보려면 을 참조하십시오. PhotoFlash PhotoFlash예제

### 샘플 정책 스토어 테스트

샘플 정책 스토어와 템플릿 연결 정책을 만든 후에는 Verified Permissions 테스트 벤치를 사용하여 시뮬레이션된 <u>권한 부여 요청</u>을 실행하여 샘플 Verified Permissions 정적 정책과 새 템플릿 연결 정책을 테스트할 수 있습니다.

샘플 정책 저장소를 생성한 시기에 따라 정책 템플릿이 이 절차의 참조와 다를 수 있습니다. 자습서의이 부분을 시작하기 전에 PhotoFlash 예제 정책 저장소에 다음 정책 템플릿이 각각 있는지 확인하십시오. 정책이 이러한 정책에 맞지 않는 경우 샘플 프로젝트 PhotoFlash옵션에서 기존 정책을 편집하거나 새 정책 저장소를 생성하십시오.

비공개로 공유된 사진에 대한 전체 액세스 권한을 부여합니다.

```
permit (
    principal in ?principal,
    action in PhotoFlash::Action::"FullPhotoAccess",
    resource in ?resource
)
when { resource.IsPrivate == false };
```

비공개로 공유한 사진에 제한적 접근 권한 부여

```
permit (
    principal in ?principal,
    action in PhotoFlash::Action::"LimitedPhotoAccess",
```

샘플 정책 스토어 테스트 10

```
resource in ?resource
)
when { resource.IsPrivate == false };
```

#### 샘플 정책 스토어 정책을 테스트하려면

1. 에서 검증된 권한 콘솔을 <a href="https://console.aws.amazon.com/verifiedpermissions/">https://console.aws.amazon.com/verifiedpermissions/</a>여십시오. 정책 저장소를 선택합니다.

- 2. 왼쪽에 있는 탐색 창에서 테스트 벤치를 선택합니다.
- 3. 비주얼 모드를 선택합니다.
- 4. Principal 섹션에서 스키마의 보안 주체 유형 중 PhotoFlash: :User를 선택합니다. 텍스트 상자에 사용자 식별자를 입력합니다. 예: Alice.
- 5. 보안 주체에 대해 상위 추가를 선택하지 마십시오.
- 6. 계정: 엔티티 속성의 경우 PhotoFlash: :Account 엔티티가 선택되어 있는지 확인하십시오. 계정 식별자를 입력합니다. 예: Alice-account.
- 7. 리소스 섹션에서 PhotoFlash: :Photo 리소스 유형을 선택합니다. 텍스트 상자에 사진 식별자를 입력합니다. 예: photo.jpeg.
- 8. 부모 추가를 선택하고 엔티티 PhotoFlash유형으로: :계정을 선택합니다. 사용자의 계정: 엔터티 필드에 지정한 사진의 상위 계정과 동일한 식별자를 입력합니다. 예: Alice-account.
- 9. 작업 섹션의 유효한 작업 목록에서 PhotoFlash: :Action::ViewPhoto" "를 선택합니다.
- 10. 추가 엔터티 섹션에서 이 엔터티 추가를 선택하여 제안된 계정 엔터티를 추가합니다.
- 11. 페이지 상단에서 권한 부여 요청 실행을 선택하여 샘플 정책 스토어의 Cedar 정책에 대한 권한 부여 요청을 시뮬레이션합니다. 테스트 벤치에는 요청을 허용하는 결정 내용이 표시되어야 합니다.

다음 표에는 Verified Permissions 테스트 벤치로 테스트할 수 있는 보안 주체, 리소스 및 작업에 대한추가 값이 나와 있습니다. 이 표에는 PhotoFlash 샘플 정책 저장소에 포함된 정적 정책과 이전 섹션에서 만든 템플릿 연결 정책을 기반으로 한 권한 부여 요청 결정이 포함되어 있습니다.

보안 주체 값	보안 주체 계 정: 엔터티 값	리소스 값	리소스 상위 값	작업	권한 부여 결 정
PhotoFlash: :	PhotoFlash: :	PhotoFlas	PhotoFlash: :	PhotoFlas	거부
사용자   앨리	계정   앨리스	h: :사진	계정   밥 어카	h: :액션::"	
스	계정	photo.jpeg	운트	"ViewPhoto	

샘플 정책 스토어 테스트 11

보안 주체 값	보안 주체 계 정: 엔터티 값	리소스 값	리소스 상위 값	작업	권한 부여 결 정
PhotoFlash: : 사용자   앨리 스	PhotoFlash: : 계정   앨리스 계정	PhotoFlas h: :사진   photo.jpeg	PhotoFlash: : 계정   앨리스 계정	PhotoFlas h: :액션::" "ViewPhoto	허용
PhotoFlash: : 사용자   앨리 스	PhotoFlash: : 계정   앨리스 계정	PhotoFlash: : 사진   Bob- photo.jpeg	PhotoFlash: : 앨범   밥 베케 이션 앨범	PhotoFlas h: :액션::"" ViewPhoto	허용
PhotoFlash: : 사용자   앨리 스	PhotoFlash: : 계정   앨리스 계정	PhotoFlash: : 사진   Bob- photo.jpeg	PhotoFlash: : 앨범   밥 베케 이션 앨범	PhotoFlas h: :액션::"" DeletePhoto	거부
PhotoFlash: : 사용자   앨리 스	PhotoFlash: : 계정   앨리스 계정	PhotoFlash: : 사진   Bob- photo.jpeg,: 불리언   트루 IsPrivate	PhotoFlash: : 앨범   밥 베케 이션 앨범	PhotoFlas h: :액션::"" ViewPhoto	거부
PhotoFlas h: :사용자   제인,PhotoF lash::   FriendGroup MySchoolF riends	PhotoFlash: : 계정   제인 계 정	PhotoFlas h: :사진   photo.jpeg	PhotoFlash: : 앨범   앨리스 가 가장 좋아 하는 앨범	PhotoFlas h: :액션::"" ViewPhoto	허용
PhotoFlas h: :사용자   제인,PhotoF lash::   FriendGroup MySchoolF riends	PhotoFlash: : 계정   제인 계 정	PhotoFlas h: :사진   photo.jpeg	PhotoFlash: : 앨범   앨리스 가 가장 좋아 하는 앨범	PhotoFlas h: :액션::"" DeletePhoto	거부

샘플 정책 스토어 테스트 12

# ID 공급자와 함께 API Gateway를 사용하기 위한 정책 저장소 생성

일반적인 사용 사례는 Amazon 검증 권한을 사용하여 Amazon API Gateway에서 APIs 호스팅되는 사용자 액세스를 승인하는 것입니다. AWS 콘솔의 마법사를 사용하여 Amazon Cognito에서 관리되는 사용자 또는 OIDC 모든 ID 공급자 (IdP) 를 위한 역할 기반 액세스 정책을 생성하고, 검증된 권한을 호출하여 이러한 정책을 평가하는 권한 부여자를 배포할 AWS Lambda 수 있습니다.

마법사를 완료하려면 <u>새 정책 저장소를 생성할</u> 때 API게이트웨이 및 ID 공급자로 설정을 선택하고 단계를 따르십시오.

API연결된 정책 저장소가 생성되고 권한 부여 요청을 위한 권한 부여 모델 및 리소스를 프로비저닝합니다. 정책 스토어에는 자격 증명 소스와 게이트웨이를 검증된 권한에 API 연결하는 Lambda 권한 부여자가 있습니다. 정책 스토어가 생성되면 사용자의 그룹 멤버십을 기반으로 API 요청을 승인할 수있습니다. 예를 들어 검증된 권한은 그룹 구성원인 사용자에게만 액세스 권한을 부여할 수 있습니다. Directors

애플리케이션이 확장되면 Cedar 정책 언어를 사용하여 사용자 속성 및 OAuth 2.0 범위로 세분화된 권한 부여를 구현할 수 있습니다. 예를 들어 확인된 권한은 도메인에 속성이 있는 사용자에게만 액세스권한을 부여할 수 있습니다. email mycompany.co.uk

에 대한 권한 부여 모델을 설정한 후 남은 책임은 애플리케이션에서 사용자를 인증하고 API 요청을 생성하고 정책 저장소를 유지 관리하는 것입니다. API

자세한 내용은 API-연결된 정책 저장소을 참조하십시오.

API연결된 정책 저장소 생성 13

# 권한 부여 모델 설계 모범 사례

소프트웨어 애플리케이션 내에서 Amazon Verified Permissions 서비스를 사용할 준비를 할 때 첫 단계로 바로 정책 설명을 작성하는 것이 어려울 수 있습니다. 이는 애플리케이션이 수행해야 할 작업을 완전히 결정하기 전에 SQL 명령문이나 API 사양을 작성하여 애플리케이션의 다른 부분을 개발하기 시작하는 것과 비슷합니다. 이 경우 먼저 사용자 경험부터 시작하여 최종 사용자가 애플리케이션 UI에서 권한을 관리할 때 확인해야 하는 내용을 명확하게 이해해야 합니다. 그런 다음, 그 경험을 바탕으로 거꾸로 되돌아가 구현 접근 방식을 결정하십시오.

이 작업을 하다 보면 다음과 같은 질문을 하게 될 것입니다.

- 어떤 리소스가 있나요? 리소스는 서로 관계가 있나요? 예를 들어, 파일은 폴더 내에 있나요?
- 보안 주체는 각 리소스에서 어떤 작업을 수행할 수 있나요?
- 보안 주체는 이러한 권한을 어떻게 획득하나요?
- 최종 사용자가 "관리자", "운영자" 또는 "" 같은 사전 정의된 권한 중에서 선택하기를 원합니까, 아니면 임시 정책 설명을 작성해야 합니까? ReadOnly 아니면 둘 다 허용할까요?
- 파일이 상위 폴더의 권한을 상속하는 것과 같이 권한이 리소스 전체에 상속되어야 할까요?
- 사용자 경험을 렌더링하려면 어떤 유형의 쿼리가 필요한가요? 예를 들어, 보안 주체가 해당 사용자의 홈페이지를 렌더링하기 위해 액세스할 수 있는 모든 리소스를 나열해야 하나요?
- 사용자가 실수로 자신의 리소스를 사용할 수 없게 될 수도 있나요? 이런 상황을 방지해야 할까요?

이러한 설계 작업의 최종 결과를 권한 부여 모델이라고 합니다. 권한 부여 모델에서는 보안 주체, 리소스, 작업, 그리고 이들이 상호 연관되는 방식을 정의합니다. 이 모델을 만드는 데에는 Cedar 또는 Verified Permissions 서비스에 대한 특별한 지식이 필요하지 않습니다. 대신 다른 설계 작업에서와 같이 무엇보다도 사용자 경험을 설계하는 연습 과정에 해당하며 인터페이스 목업, 논리적 다이어그램, 권한이 사용자가 제품에서 보는 내용에 미치는 영향에 대한 전반적인 설명과 같은 아티팩트를 통해 구현될 수 있습니다. Cedar는 Cedar의 구현을 준수하기 위해 모델이 부자연스럽게 변형되도록 강요하는 대신, 모델을 통해 고객의 요구 사항을 충족시킬 수 있을 만큼 충분히 유연하도록 설계되었습니다. 따라서 최적의 모델을 찾는 가장 좋은 방법은 원하는 사용자 경험을 명확하게 이해하는 것입니다.

이 섹션에서는 설계 연습 작업에 접근하는 방식에 대한 일반적인 지침, 주의해야 할 사항, Verified Permissions를 성공적으로 사용하기 위한 모범 사례 모음을 제공합니다.

여기에 제시된 지침 외에도 Cedar 정책 언어 참조 가이드의 모범 사례도 고려해야 합니다.

주제

- 표준적인 "올바른" 모델은 없습니다.
- 운영 이외의 리소스에만 집중하세요 API
- 복합 권한 부여는 정상입니다.
- 멀티 테넌시 고려 사항
- 가능하면 정책 범위를 채우는 것이 좋습니다.
- 모든 리소스는 컨테이너 안에 있어야 합니다.
- 보안 주체를 리소스 컨테이너와 분리하세요.
- 속성 또는 템플릿을 사용하여 관계를 나타냅니다.
- <u>모델에서는 세분화된 권한을 적용하고 사용자 인터페이스에서는 권한을 포괄적으로 표시하는 것이</u> 좋습니다.
- 권한 부여 쿼리가 필요한 다른 이유도 고려하세요.

## 표준적인 "올바른" 모델은 없습니다.

권한 부여 모델을 설계할 때 단 하나의 고유한 정답은 없습니다. 애플리케이션마다 유사한 개념에 대해서로 다른 권한 부여 모델을 효과적으로 사용할 수 있으며, 그렇게 해도 괜찮습니다. 예를 들어 컴퓨터의 파일 시스템을 생각해 보세요. UNIX와 유사한 운영 체제에서 파일을 만들면 상위 폴더의 권한이 자동으로 상속되지 않습니다. 반면 다른 많은 운영 체제와 대부분의 온라인 파일 공유 서비스에서는 파일이 상위 폴더로부터 권한을 상속합니다. 두 가지 선택 모두 애플리케이션이 최적화되는 상황에 따라 유효합니다.

권한 부여 솔루션의 정확성은 절대적인 것이 아니며 고객이 원하는 경험을 제공하는 방법과 고객이 기대하는 방식으로 리소스를 보호하는지 여부 측면에서 살펴봐야 합니다. 권한 부여 모델이 이를 뒷받침한다면 성공이라고 할 수 있습니다.

따라서 효과적인 권한 부여 모델을 만들기 위해서는 원하는 사용자 경험을 바탕으로 설계를 시작하는 것이 가장 유용한 전제 조건입니다.

### 운영 이외의 리소스에만 집중하세요 API

대부분의 소비자 대상 애플리케이션에서 권한은 애플리케이션이 지원하는 리소스를 중심으로 모델링됩니다. 예를 들어 파일 공유 애플리케이션은 권한을 파일 또는 폴더에서 수행할 수 있는 작업으로 나타낼 수 있습니다. 기본 구현과 백엔드 API 작업을 추상화하는 훌륭하고 간단한 모델입니다.

반면 다른 유형의 애플리케이션, 특히 웹 서비스는 API 운영 자체를 중심으로 권한을 설계하는 경우 가 많습니다. 예를 들어 웹 서비스가 API 이름이 지정된 이름을 createThing() 제공하는 경우 권

한 부여 모델은 해당 권한을 정의하거나 action Cedar에서 이름이 지정된 createThing 권한을 정의할 수 있습니다. 이는 다양한 상황에서 사용할 수 있으며 권한을 쉽게 이해할 수 있게 해줍니다. createThing 작업을 호출하려면 createThing 작업 권한이 필요합니다. 간단해 보이죠?

Verified Permissions 콘솔의 <u>시작</u> 프로세스에는 에서 직접 리소스와 작업을 빌드할 수 있는 API 옵션 이 포함되어 있습니다. 이는 유용한 기준입니다. API 즉, 정책 저장소와 해당 정책 저장소가 권한을 부여하는 대상 간의 직접 매핑입니다.

그러나 이 API 초점을 맞춘 접근 방식은 고객이 진정으로 보호하고자 하는 대상, 즉 기본 데이터 및 리소스에 대한 프록시에 불과하기 때문에 APIs 최적이 아닐 수 있습니다. 동일한 리소스에 대한 액세스를 여러 개 APIs 제어할 경우 관리자가 해당 리소스에 대한 경로를 추론하고 그에 따라 액세스를 관리하기가 어려울 수 있습니다.

예를 들어, 조직의 구성원이 포함된 사용자 디렉터리를 생각해 보세요. 사용자는 그룹으로 구성할 수 있으며 보안 목표 중 하나는 승인되지 않은 당사자가 그룹 구성원을 발견하지 못하도록 하는 것입니다. 이 사용자 디렉토리를 관리하는 서비스는 다음 두 가지 API 작업을 제공합니다.

- listMembersOfGroup
- listGroupMembershipsForUser

고객은 이 두 작업 중 하나를 사용하여 그룹 멤버십을 발견할 수 있습니다. 따라서 권한 관리자는 두 작업 모두에 대해 액세스를 조정해야 한다는 점을 기억해야 합니다. 나중에 다음과 같은 추가 사용 사례를 해결하기 위해 새 API 작업을 추가하기로 선택하면 더 복잡해집니다.

• isUserInGroups(사용자가 하나 이상의 그룹에 속하는지 빠르게 API 테스트하기 위한 새로운 기능)

보안 관점에서 보면 그룹 구성원 자격을 발견할 수 있는 세 번째 경로가 API 열려 관리자가 신중하게 만든 권한이 손상됩니다.

API의미 체계를 무시하고 대신 기본 데이터와 리소스, 연결 작업에 집중하는 것이 좋습니다. 이 접근 방식을 그룹 멤버십 예제에 적용하면 추상적 권한 (예: 세 가지 API 작업 각각이 참조해야 함) 이 생성됩 니다. viewGroupMembership

API이름	권한	
listMembersOfGroup	그룹에 대해 viewGroupMembership	권한이 필요합니다.

리소스에 집중하세요. 16

API이름	권한	
listGroupMembershi psForUser	사용자에 대해 viewGroupMembership	권한이 필요합니다.
isUserInGroups	사용자에 대해 viewGroupMembership	권한이 필요합니다.

이 권한 하나를 정의함으로써 관리자는 그룹 멤버십 검색에 대한 액세스를 현재, 그리고 영구적으로 성공적으로 제어할 수 있습니다. 절충안으로, 이제 각 API 작업에 필요한 몇 가지 권한을 문서화해야 하며 관리자는 권한을 만들 때 이 설명서를 참조해야 합니다. 보안 요구 사항을 충족하기 위해 필요한 경우 이 방법은 유효한 절충안이 될 수 있습니다.

### 복합 권한 부여는 정상입니다.

복합 권한 부여는 애플리케이션 인터페이스의 버튼 클릭과 같은 단일 사용자 활동에 대해 해당 활동의 허용 여부를 결정하기 위해 여러 개의 개별 권한 부여 쿼리가 필요할 때 발생합니다. 예를 들어 파일 시 스템의 새 디렉터리로 파일을 이동하려면 세 가지 권한, 즉 소스 디렉터리에서 파일을 삭제하는 권한, 대상 디렉터리에 파일을 추가하는 기능, (애플리케이션에 따라) 파일 자체를 조작할 수 있는 권한이 필 요할 수 있습니다.

권한 부여 모델을 처음 설계하는 경우 모든 권한 부여 결정을 단일 권한 부여 쿼리로 해결할 수 있어야한다고 생각할 수 있습니다. 하지만 그로 인해 모델이 지나치게 복잡해지고 정책 설명이 복잡해질 수있습니다. 실제로 복합 권한 부여를 사용하면 더 간단한 권한 부여 모델을 만드는 데 유용할 수 있습니다. 잘 설계된 권한 부여 모델의 한 가지 척도는 개별 작업으로 충분히 분해한다면 파일 이동과 같은 복잡한 작업을 직관적인 기본 요소 집합으로 표현할 수 있다는 것입니다.

복합 권한 부여가 발생하는 또 다른 상황은 권한 부여 프로세스에 여러 당사자가 관여하는 경우입니다. 사용자가 그룹의 구성원이 될 수 있는 조직 디렉터리를 생각해 보세요. 간단한 접근 방식은 그룹 소유자에게 다른 사람을 추가할 수 있는 권한을 부여하는 것입니다. 하지만 사용자가 먼저 추가에 동의하도록 하려면 어떻게 해야 할까요? 이 경우 사용자와 그룹 모두 멤버십에 동의해야 하는 핸드셰이크가 도입됩니다. 이를 위해 사용자에게 부여되는 또 다른 권한을 도입하여 사용자를 모든 그룹에 추가할 수있는지 또는 특정 그룹에 추가할 수 있는지 지정할 수 있습니다. 이후에 호출자가 그룹에 구성원을 추가하려고 하면 애플리케이션은 양쪽의 권한을 모두 적용해야 합니다. 즉, 호출자에게 지정된 그룹에 구성원을 추가하려고 하면 애플리케이션은 양쪽의 권한을 모두 적용해야 합니다. 즉, 호출자에게 지정된 그룹에 구성원을 추가한 수 있는 권한이 있고 추가되는 개별 사용자에게는 추가될 수 있는 권한이 있어야 합니다. N방향 핸드셰이크가 있는 경우 핸드셰이크의 각 부분을 적용하기 위해 N개의 복합 권한 부여 쿼리를 확인하는 것이 일반적입니다.

여러 리소스가 관련되어 있고 권한을 모델링하는 방법이 명확하지 않다는 설계 문제가 있는 경우 복잡한 권한 부여 시나리오가 있다는 신호일 수 있습니다. 이 경우 작업을 여러 개의 개별 권한 부여 확인 작업으로 분해하여 해결책을 찾을 수 있습니다.

### 멀티 테넌시 고려 사항

애플리케이션을 사용하는 기업이나 테넌트 등 여러 고객이 사용할 애플리케이션을 개발하고 이를 Amazon Verified Permissions와 통합하고 싶을 수 있습니다. 권한 부여 모델을 개발하기 전에 멀티테 넌트 전략을 개발하십시오. 하나의 공유 정책 저장소에서 고객 정책을 관리하거나 각 고객에게 테넌트 별 정책 저장소를 할당할 수 있습니다.

#### 1. 공유 정책 저장소 1개

모든 테넌트는 단일 정책 저장소를 공유합니다. 애플리케이션은 모든 권한 부여 요청을 공유 정책 저장소에 보냅니다.

#### 2. 테넌트별 정책 저장소

각 테넌트에는 전용 정책 저장소가 있습니다. 애플리케이션은 요청하는 테넌트에 따라 여러 정책 저 장소를 쿼리하여 권한 부여 결정을 내립니다.

두 전략 모두 청구서에 영향을 미칠 수 있는 권한 부여 요청의 양을 상대적으로 많이 생성하지 않습니다. AWS 그렇다면 접근 방식을 어떻게 설계해야 할까요? 다음은 검증된 권한 멀티테넌시 권한 부여 전략에 영향을 미칠 수 있는 일반적인 조건입니다.

#### 테너트 정책 격리

테넌트 데이터를 보호하려면 각 테넌트의 정책을 다른 테넌트와 분리하는 것이 중요합니다. 각 테 넌트에 자체 정책 저장소가 있는 경우 각 테넌트에는 고유한 격리된 정책 집합이 있습니다.

#### 권한 부여 흐름

요청의 정책 저장소 ID와 테넌트별 정책 저장소를 사용하여 권한 부여를 요청하는 테넌트를 식별할 수 있습니다. 공유 정책 저장소에서는 모든 요청이 동일한 정책 저장소 ID를 사용합니다.

#### 템플릿 및 스키마 관리

<u>정책 템플릿과</u> <u>정책 저장소 스키마는</u> 각 정책 저장소에 일정 수준의 설계 및 유지 관리 오버헤드를 추가합니다.

멀티테넌시고려 18

#### 글로벌 정책 관리

일부 글로벌 정책을 모든 테넌트에 적용할 수 있습니다. 글로벌 정책 관리에 대한 오버헤드 수준은 공유 정책 저장소 모델과 테넌트별 정책 저장소 모델에 따라 다릅니다.

#### 테넌트 오프보딩

일부 테넌트는 해당 사례에 맞는 스키마 및 정책 요소를 제공합니다. 테넌트가 더 이상 조직에서 활동하지 않는 경우 데이터를 제거하려는 경우 노력의 수준은 다른 테넌트와의 격리 수준에 따라 달라집니다.

#### 서비스 리소스 할당량

검증된 권한에는 멀티테넌시 결정에 영향을 미칠 수 있는 리소스 및 요청률 할당량이 있습니다. 할 당량에 대한 자세한 내용은 리소스 할당량 섹션을 참조하세요.

### 공유 정책 저장소와 테넌트별 정책 저장소 비교

공유 및 테넌트별 정책 저장소 모델에서 각 고려 사항마다 고유한 수준의 시간 및 리소스 투입이 필요합니다.

고려 사항	공유 정책 저장소의 노력 수준	테넌트별 정책 저장소의 노력 수준
테넌트 정책 격리	미디엄. 정책 및 권한 부여 요청 에 테넌트 식별자를 포함해야 합니다.	낮음. 격리는 기본 동작입니다. 테넌트별 정책은 다른 테넌트 가 액세스할 수 없습니다.
승인 흐름	낮음. 모든 쿼리는 하나의 정책 저장소를 대상으로 합니다.	미디엄. 각 테넌트와 해당 정책 저장소 ID 간의 매핑을 유지해 야 합니다.
템플릿 및 스키마 관리	낮음. 모든 테넌트에 대해 하나의 스키마가 작동하도록 만들어야 합니다.	높음. 스키마와 템플릿은 개별 적으로 복잡하지 않을 수 있지 만 변경에는 더 많은 조정과 복 잡성이 필요합니다.
글로벌 정책 관리	낮음. 모든 정책은 전 세계에 적 용되며 중앙에서 업데이트할 수 있습니다.	높음. 온보딩 시 각 정책 저장소 에 글로벌 정책을 추가해야 합 니다. 여러 정책 저장소 간에 글

로벌 정책 업데이트를 복제하 십시오.

테넌트 오프보딩 미디엄. 테넌트별 정책만 식별

하고 삭제해야 합니다.

낮음. 정책 저장소를 삭제합니

낮음. 각 테넌트에는 전용 리소

다.

서비스 리소스 할당량 높음. 테넌트는 스키마 크기, 리

소스당 정책 크기, 정책 저장소 별 ID 소스 등 정책 저장소에 영 향을 미치는 리소스 할당량을

스 할당량이 있습니다.

공유합니다.

### 선택 방법

각 멀티테넌트 애플리케이션은 다릅니다. 아키텍처 결정을 내리기 전에 두 접근 방식과 고려 사항을 주의 깊게 비교하십시오.

애플리케이션에 테넌트별 정책이 필요하지 않고 단일 <u>ID 소스를</u> 사용하는 경우 모든 테넌트를 위한 하나의 공유 정책 저장소가 가장 효과적인 솔루션일 수 있습니다. 따라서 권한 부여 흐름과 글로벌 정책 관리가 단순해집니다. 애플리케이션에서 테넌트별 정책을 삭제할 필요가 없으므로 하나의 공유 정책 저장소를 사용하여 테넌트를 오프보딩하는 데 드는 노력이 줄어듭니다.

그러나 애플리케이션에 많은 테넌트별 정책이 필요하거나 여러 <u>ID 소스를</u> 사용하는 경우에는 테넌트별 정책 저장소가 가장 효과적일 수 있습니다. 각 정책 저장소에 테넌트별 권한을 부여하는 IAM 정책을 사용하여 테넌트 정책에 대한 액세스를 제어할 수 있습니다. 테넌트를 오프보딩하려면 해당 정책 저장소를 삭제해야 합니다. shared-policy-store 환경에서는 테넌트별 정책을 찾아 삭제해야 합니다.

# 가능하면 정책 범위를 채우는 것이 좋습니다.

정책 범위는 Cedar 정책 설명에서 permit or forbid 키워드 뒤와 여는 괄호 사이의 부분입니다.

선택 방법 20

```
Effect permit (

principal == User::"e3527bb8-f74a-48da-818c-f7e6ef79bf7c",

action == Photo::"readFile",

resource in Album::"615e85bc-f03d-4915-b4eb-4c184b8da25d"
)

when {

resource.private == false
};
```

가능하면 principal 및 resource 값을 채우는 것이 좋습니다. 이렇게 하면 Verified Permissions가 정책을 인덱싱하여 검색을 보다 효율적으로 수행할 수 있으므로 성능이 향상됩니다. 여러 보안 주체 또는 리소스에 동일한 권한을 부여해야 하는 경우 정책 템플릿을 사용하고 각 보안 주체 및 리소스 쌍에 정책 템플릿을 연결하는 것이 좋습니다.

when 조항에 보안 주체 및 리소스 목록을 포함하는 대규모 단일 정책을 생성하는 방법은 사용하지 마십시오. 이렇게 하면 확장성 한계에 부딪히거나 운영상의 문제가 발생할 수 있습니다. 예를 들어 정책 내의 대규모 목록에서 단일 사용자를 추가하거나 제거하려면 전체 정책을 읽고, 목록을 편집하고, 새 정책 전체를 작성하고, 한 관리자가 다른 관리자의 변경 내용을 덮어쓸 경우 발생하는 동시성 오류를 처리해야 합니다. 반면, 세분화된 권한을 많이 사용하면 사용자에게 적용되는 단일 정책을 추가하거나 제거하는 것만큼 간단하게 사용자를 추가하거나 제거할 수 있습니다.

## 모든 리소스는 컨테이너 안에 있어야 합니다.

권한 부여 모델을 설계할 때는 모든 작업이 특정 리소스와 연결되어야 합니다. 예를 들어 viewFile 작업을 적용할 수 있는 리소스는 직관적으로 알 수 있습니다. 개별 파일일 수도 있고 폴더 내 여러 개 파일일 수도 있습니다. 그러나 createFile과 같은 작업은 직관적으로 알기 어렵습니다. 파일을 생성하는 기능을 모델링할 때 해당 기능은 어떤 리소스에 적용될까요? 파일이 아직 존재하지 않기 때문에 파일자체일 수는 없습니다.

이는 리소스 생성과 관련된 일반적인 문제의 예입니다. 리소스 생성은 부트스트래핑 문제에 해당합니다. 아직 리소스가 존재하지 않는 경우에도 무언가가 리소스를 만들 수 있는 권한을 가질 수 있는 방법이 있어야 합니다. 이 경우 해결 방법은 모든 리소스가 어떤 컨테이너 내에 존재해야 하며 권한의 기준점 역할을 하는 것은 컨테이너 자체라는 점을 인식하는 것입니다. 예를 들어, 시스템에 폴더가 이미 있는 경우 새 리소스를 인스턴스화하는 데 권한이 필요한 위치가 바로 그 폴더이므로 파일을 만드는 기능을 해당 폴더에 대한 권한으로 모델링할 수 있습니다.

```
permit (
```

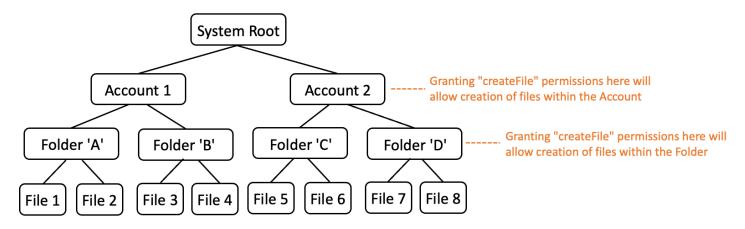
```
principal == User::"6688f676-1aa9-456a-acf4-228340b54e9d",
    action == Action::"createFile",
    resource == Folder::"c863f89b-461f-4fc2-b638-e5fa5f79a48b"
);
```

하지만 폴더가 없다면 어떻게 할까요? 새로 생긴 고객 계정이라 아직 리소스가 없는 애플리케이션인 경우도 있습니다. 이 경우에도 고객이 새 파일을 어디에서 만들 수 있는지 질문하면 직관적으로 이해할 수 있는 컨텍스트가 여전히 존재합니다. 고객이 임의의 고객 계정 내에서 파일을 생성할 수 있게 하고 싶지는 않을 것입니다. 그보다는 고객 자신의 계정 경계라는 암시적인 컨텍스트가 존재합니다. 따라서 계정 자체가 리소스 생성을 위한 컨테이너를 나타내며, 다음 예와 유사한 정책에서 이를 명시적으로 모델링할 수 있습니다.

```
// Grants permission to create files within an account,
// or within any sub-folder inside the account.
permit (
    principal == User::"6688f676-1aa9-456a-acf4-228340b54e9d",
    action == Action::"createFile",
    resource in Account::"c863f89b-461f-4fc2-b638-e5fa5f79a48b"
);
```

하지만 계정이 하나도 없다면 어떻게 할까요? 시스템에 새 계정을 생성하도록 고객 가입 워크플로를 설계할 수도 있습니다. 그렇다면 프로세스가 계정을 생성할 수 있는 가장 바깥쪽 경계를 담을 컨테이너 가 필요할 것입니다. 이 루트 수준 컨테이너는 시스템 전체를 나타내며 이름을 "시스템 루트"와 같이 이 름을 지정할 수 있습니다. 하지만 이것이 필요한지 여부와 이름을 어떻게 지정할지는 애플리케이션 소 유자인 여러분에게 달려 있습니다.

이 샘플 애플리케이션의 경우 결과적인 컨테이너 계층 구조는 다음과 같이 표시됩니다.

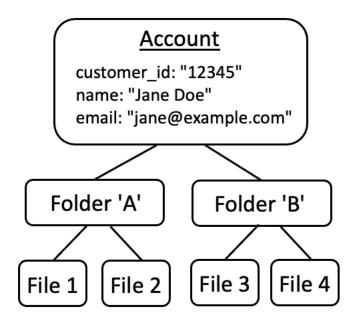


이 예는 샘플 계층 구조입니다. 다른 구조도 유효하게 사용할 수 있습니다. 기억해야 할 점은 리소스 생성은 항상 리소스 컨테이너의 컨텍스트 내에서 이루어진다는 것입니다. 이러한 컨테이너는 계정 경계

와 같이 암시적일 수 있으며 간과하기 쉽습니다. 권한 부여 모델을 설계할 때는 이러한 암시적 가정을 참고하여 공식적으로 문서화하고 권한 부여 모델 내에 표현할 수 있도록 해야 합니다.

### 보안 주체를 리소스 컨테이너와 분리하세요.

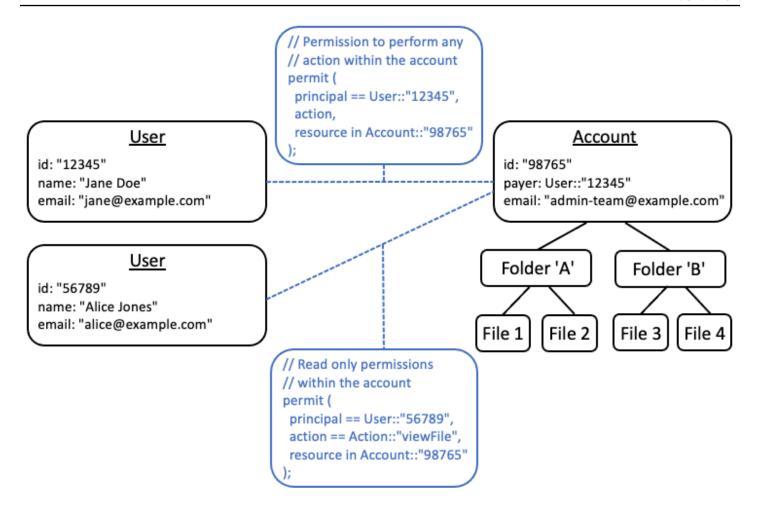
리소스 계층 구조를 설계할 때, 특히 소비자 대상 애플리케이션의 일반적인 성향 중 하나는 고객의 사용자 ID를 고객 계정 내 리소스의 컨테이너로 사용하는 것입니다.



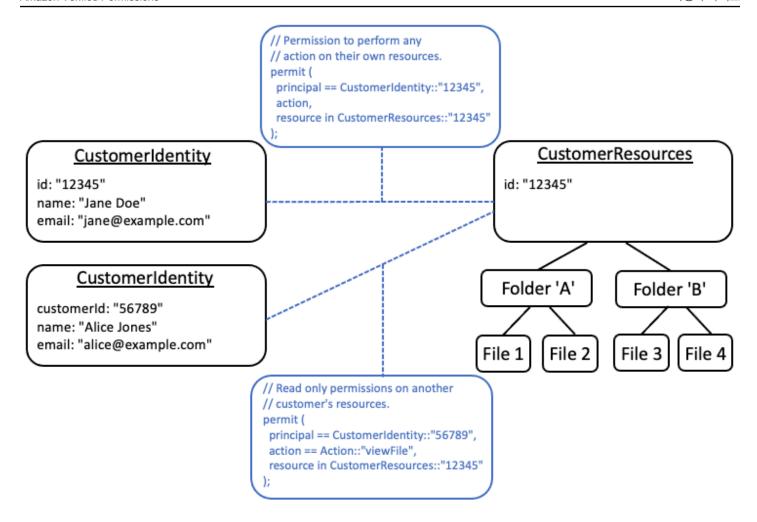
이 전략은 안티패턴으로 간주하는 것이 좋습니다. 애플리케이션이 복잡해질수록 추가 사용자에게 액세스 권한을 위임하려는 자연스러운 경향이 있기 때문입니다. 예를 들어 다른 사용자가 계정 리소스를 공유할 수 있는 "가족" 계정을 도입하는 경우가 있습니다. 마찬가지로 기업 고객도 계정의 일부에 대해여러 명의 인력을 운영자로 지정하고자 하는 경우가 있습니다. 계정의 소유권을 다른 사용자에게 이전하거나 여러 계정의 리소스를 병합해야 할 수도 있습니다.

사용자 ID를 계정의 리소스 컨테이너로 사용하는 경우 앞의 시나리오를 달성하기가 더 어려워집니다. 더욱 놀라운 사실은 이 접근 방식에서 다른 사용자에게 계정 컨테이너에 대한 액세스 권한이 부여될 경 우 Jane의 이메일이나 로그인 보안 인증 정보를 변경하는 등 사용자 ID 자체를 수정할 수 있는 액세스 권한이 의도치 않게 부여될 수 있다는 점입니다.

따라서 가능하면 보안 주체와 리소스 컨테이너를 분리하고 "관리자 권한" 또는 "소유권"과 같은 개념을 사용하여 보안 주체와 리소스 컨테이너 간의 연결을 모델링하는 것이 더 탄력적인 접근 방식입니다.



기존 애플리케이션이 이러한 분리된 모델을 사용할 수 없는 경우 권한 부여 모델을 설계할 때 최대한 모방하는 것을 고려해 보는 것이 좋습니다. 예를 들어 사용자 ID, 로그인 보안 인증 정보 및 소유한 리소스를 캡슐화하는 Customer라는 이름의 단 하나의 개념만 가지고 있는 애플리케이션은 이를 Customer Identity(이름, 이메일 등)에 대한 하나의 논리적 엔터티와 소유한 모든 리소스의 상위 노드 역할을 하는 Customer Resources 또는 Customer Account에 대한 별도의 논리적 엔터티를 포함하는 권한 부여 모델에 매핑할 수 있습니다. 두 엔티티는 동일한 Id를 공유할 수 있지만 Type은 서로 다릅니다.



## 속성 또는 템플릿을 사용하여 관계를 나타냅니다.

리소스 간의 관계를 표현하는 데는 크게 두 가지 방법이 있습니다. 둘 중 하나를 사용하는 시기는 관계가 이미 애플리케이션 데이터베이스에 저장되어 있고 규정 준수와 같은 다른 이유로 사용되었는지 여부에 따라 달라집니다. 그렇다면 <u>속성 기반</u> 접근 방식을 취하십시오. 그렇지 않다면 <u>템플릿</u> 기반 접근 방식을 사용하세요.

### 속성 기반 관계

속성은 권한 부여 결정의 입력으로 사용하여 보안 주체와 하나 이상의 리소스 간의 관계를 나타낼 수 있습니다.

이 패턴은 권한 관리 이외의 목적으로 관계를 추적하고 관리하는 경우에 적합합니다. 예를 들어 Know Your Customer 규칙을 재정적으로 준수하려면 주 계좌 소유자를 기록해야 합니다. 권한은 이러한 관계에서 파생됩니다. 관계 데이터는 권한 부여 시스템 외부에서 관리되며 권한 부여 결정을 내릴 때 입력으로 가져옵니다.

관계 표현하기 25

다음 예는 Alice 사용자와 사용자가 주 계정 소유자인 여러 계정 간의 관계를 나타내는 방법을 보여줍니다.

```
// Using a user attribute to represent the primary account holder relationship
{
    "id": "df82e4ad-949e-44cb-8acf-2d1acda71798",
    "name": "alice",
    "email": "alice@example.com",
    "primaryOnAccounts": [
        "Account::\"c943927f-d803-4f40-9a53-7740272cb969\"",
        "Account::\"b8ee140c-fa09-46c3-992e-099438930894\""
    ]
}
```

그리고 이후에 정책 내에서 해당 속성을 사용하는 경우입니다.

```
// Derived relationship permissions
permit (
   principal,
   action in Action::"primaryAccountHolderActions",
   resource
)when {
   resource in principal.primaryOnAccounts
};
```

반대로, 동일한 관계를 사용자 집합을 포함하는 primaryAccountHolders 이라는 리소스의 속성으로 표시할 수도 있습니다.

보안 주체와 리소스 간에 관계 유형이 여러 개 있는 경우 이러한 유형을 서로 다른 속성으로 모델링해야 합니다. 예를 들어 계정에 승인된 서명자가 있을 수 있고 이러한 개인이 계정에 대해 서로 다른 권한을 갖는 경우 이는 다른 속성으로 표시됩니다.

위의 경우, 세 번째 계정의 승인된 서명자일 Alice 수도 있습니다. 다음 예는 이를 어떻게 표현할 수 있는지를 보여줍니다.

```
// Using user attributes to represent the primary account holder and authorized
signatory relationships
{
   "id": "df82e4ad-949e-44cb-8acf-2d1acda71798",
   "name": "alice",
   "email": "alice@example.com",
   "primaryOnAccounts": [
```

 속성 기반 관계
 26

```
"Account::\"c943927f-d803-4f40-9a53-7740272cb969\"",
    "Account::\"b8ee140c-fa09-46c3-992e-099438930894\""
],
    "authorizedSignatoryOnAccounts": [
    "Account::\"661817a9-d478-4096-943d-4ef1e082d19a\""
]
}
```

해당 정책은 다음과 같습니다.

```
// Derived relationship permissions

permit (
    principal,
    action in Action::"primaryAccountHolderActions",
    resource
)when {
    resource in principal.primaryOnAccounts
};
permit (
    principal,
    action in Action::"authorizedSignatoryActions",
    resource
)when {
    resource in principal.authorizedSignatoryOnAccounts
};
```

### 템플릿 기반 관계

리소스 간의 관계가 권한 관리 목적으로만 존재하는 경우 이 관계를 템플릿 연결 정책 또는 템플릿으로 저장하는 것이 좋습니다. 또한 이러한 템플릿을 특정 리소스에 할당된 역할로 생각할 수도 있습니다.

예를 들어, 문서 관리 시스템에서 문서 소유자는 다른 사용자에게 문서에 기여할 수 있는 권한을 부여하도록 선택할 수 있습니다. Alice Bob 이렇게 하면 Bob과 Alice의 문서 사이에 기여자 관계가 형성됩니다. 이 관계의 유일한 목적은 문서를 편집하고 댓글을 달 수 있는 권한을 부여하는 것이며, 따라서 이 관계를 템플릿으로 표현할 수 있습니다. 이러한 경우에는 각 관계 유형에 맞는 템플릿을 만드는 것이좋습니다. 다음 예에는 두 개의 관계 유형이 Contributor 있으며Reviewer, 따라서 두 개의 템플릿이 있습니다.

다음 템플릿을 사용하여 개별 사용자를 위한 템플릿 연결 정책을 만들 수 있습니다.

```
// Managed relationship permissions - Contributor template
```

템플릿 기반 관계 27

```
permit (
    principal == ?principal,
    action in Action::"DocumentContributorActions",
    resource in ?resource
);

// Managed relationship permissions - Reviewer template
permit (
    principal == ?principal,
    action in Action::"DocumentReviewerActions",
    resource in ?resource
);
```

다음 템플릿을 사용하여 사용자 그룹을 위한 템플릿 연결 정책을 만들 수 있습니다. 개별 사용자를 위한 템플릿과의 유일한 차이점은 대신 in 연산자를 사용한다는 것입니다. ==

```
// Managed relationship permissions - Contributor template
permit (
   principal in ?principal,
   action in Action::"DocumentContributorActions",
   resource in ?resource
);

// Managed relationship permissions - Reviewer template
permit (
   principal in ?principal,
   action in Action::"DocumentReviewerActions",
   resource in ?resource
);
```

그런 다음 이러한 템플릿을 사용하여 문서에 대한 액세스 권한이 부여될 때마다 관리 관계 권한을 나타 내는 다음과 같은 정책을 만들 수 있습니다.

```
//Managed relationship permissions
permit (
   principal in User::"df82e4ad-949e-44cb-8acf-2d1acda71798",
   action in Action::"DocumentContributorActions",
   resource in Document::"c943927f-d803-4f40-9a53-7740272cb969"
);

permit (
   principal in UserGroup::"df82e4ad-949e-44cb-8acf-2d1acda71798",
```

템플릿 기반 관계 28

```
action in Action::"DocumentReviewerActions",
  resource == Document::"661817a9-d478-4096-943d-4ef1e082d19a"
);

permit (
  principal in User::"df82e4ad-949e-44cb-8acf-2d1acda71798",
  action in Action::"DocumentContributorActions",
  resource in Folder::"b8ee140c-fa09-46c3-992e-099438930894"
);
```

Amazon Verified Permissions는 권한 부여 평가 중에 많은 개별적이고 세분화된 정책을 효율적으로 처리할 수 있으며, 이러한 방식으로 모델링하면 Verified Permissions가 모든 권한 부여 결정에 대한 전체감사 로그를 유지 관리합니다. AWS CloudTrail

모델에서는 세분화된 권한을 적용하고 사용자 인터페이스에서는 권한을 포괄적으로 표시하는 것이 좋습니다.

설계자가 나중에 종종 후회하는 전략 중 하나는 Read 및 Write와 같은 매우 광범위한 작업을 포함하는 권한 부여 모델을 설계한 후 나중에 더 세분화된 조치가 필요하다는 것을 깨닫는 것입니다. 세분화에 대한 요구는 보다 세밀한 액세스 제어를 원하는 고객 피드백이나 최소 권한 원칙을 권장하는 규정 준수 및 보안 감사자로 인해 더 강하게 제기될 수도 있습니다.

세분화된 권한을 미리 정의하지 않은 경우 애플리케이션 코드와 정책 설명을 더 세분화된 사용자 권한으로 수정하는 복잡한 변환이 필요할 수 있습니다. 예를 들어, 이전에 포괄적으로 구분된 작업에 대해 권한을 부여한 애플리케이션 코드는 세분화된 작업에 대해 사용하기 위해 수정해야 합니다. 또한 마이그레이션 내용을 반영하도록 정책을 업데이트해야 합니다.

이렇게 비용이 많이 드는 마이그레이션을 피하려면 사전에 세분화된 권한을 정의하는 것이 좋습니다. 그러나 최종 사용자가 이후에 더 많은 수의 세분화된 권한을 이해하도록 강요받게 되는 경우, 특히 대

세분화된 권한 29

부분의 고객이 Read 및 Write와 같이 포괄적으로 구분된 제어 기능에 만족하는 경우 장단점이 발생할 수 있습니다. 두 가지 방식의 장점을 최대한 활용하려면 정책 템플릿이나 작업 그룹과 같은 메커니즘을 사용하여 세분화된 권한을 Read 및 Write와 같은 사전 정의된 집합으로 그룹화할 수 있습니다. 이 접근 방식을 사용하면 고객은 포괄적으로 구분된 권한만 보게 됩니다. 하지만 이면에서는 포괄적으로 구분된 권한을 세분화된 작업의 집합으로 모델링하여 애플리케이션을 향후에도 사용할 수 있도록 대비하게 됩니다. 고객이나 감사자가 요청하는 경우 세분화된 권한이 노출될 수 있습니다.

# 권한 부여 쿼리가 필요한 다른 이유도 고려하세요.

일반적으로 권한 부여 확인은 사용자 요청과 연결됩니다. 이 확인 작업은 사용자에게 해당 요청을 수행할 권한이 있는지 여부를 확인하는 작업입니다. 하지만 권한 부여 데이터는 애플리케이션 인터페이스디자인에 영향을 주는 데 이용할 수도 있습니다. 예를 들어 최종 사용자가 액세스할 수 있는 리소스의목록만 보여주는 홈 화면이 표시되게 할 수 있습니다. 리소스의 세부 정보를 볼 때 사용자가 해당 리소스에서 수행할 수 있는 작업만 인터페이스에 표시되도록 할 수도 있습니다.

이러한 상황에서는 권한 부여 모델에 장단점이 생길 수 있습니다. 예를 들어 속성 기반 액세스 제어 (ABAC) 정책에 지나치게 의존하면 "누가 무엇에 액세스할 수 있습니까?" 라는 질문에 빠르게 답하기가 더 어려워질 수 있습니다. 이 질문에 답하려면 모든 보안 주체와 리소스를 대상으로 각 규칙을 검토하여 일치하는 항목이 있는지 확인해야 하기 때문입니다. 따라서 사용자가 액세스할 수 있는 리소스만 나열하도록 최적화해야 하는 제품은 역할 기반 액세스 제어 () 모델을 사용할 수 있습니다. RBAC 를 사용하면 RBAC 사용자에게 연결된 모든 정책을 반복하여 리소스 액세스를 결정하는 것이 더 쉬워질 수 있습니다.

# Amazon Verified Permissions 정책 스토어

정책 스토어는 정책 및 정책 템플릿의 컨테이너입니다. 각 정책 저장소에서 정책 저장소에 추가된 정책을 검증하는 데 사용되는 스키마를 생성할 수 있습니다. 또한 정책 검증을 활성화할 수 있습니다. 정책검증이 활성화된 상태에서 정책 저장소에 정책을 추가하면 정책에 정의된 엔티티 유형, 공통 유형 및 작업이 스키마에 대해 검증되고 잘못된 정책은 거부됩니다.

애플리케이션당 하나의 정책 스토어를 생성하거나 다중 테넌트 애플리케이션의 경우 테넌트당 하나의 정책 스토어를 생성하는 것이 좋습니다. 권한 부여 요청 시 정책 스토어를 지정해야 합니다.

이 경우 혼동을 방지하기 위해 정책 스토어의 Cedar 엔터티에 네임스페이스를 사용하는 것이 좋습니다. 네임스페이스는 콜론 한 쌍(::)을 구분 기호로 사용하여 구분되는 형식의 문자열 접두사입니다. Verified Permissions는 정책 스토어당 하나의 네임스페이스를 지원합니다. 이러한 네임스페이스는 여러 개의 유사한 애플리케이션으로 작업할 때 상황을 깔끔하게 유지하는 데 도움이 됩니다. 예를 들어 멀티 테넌트 애플리케이션에서 네임스페이스를 사용하여 스키마에 정의된 유형에 테넌트 이름을 추가하면 다른 테넌트가 사용하는 유사한 유형과 구분됩니다. 권한 부여 요청 로그를 보면 권한 부여 요청을 처리한 테넌트를 쉽게 식별할 수 있습니다. 자세한 내용은 Cedar 정책 언어 참조 가이드의 네임스페이스를 참조하세요.

#### 주제

- Verified Permissions 정책 스토어 생성
- API-연결된 정책 저장소
- 정책 스토어 삭제

# Verified Permissions 정책 스토어 생성

다음 방법을 사용하여 정책 스토어를 생성할 수 있습니다.

- 설정 안내 따르기 첫 번째 정책을 생성하기 전에 유효한 작업과 보안 주체 유형을 포함하는 리소스 유형을 정의합니다.
- API게이트웨이 및 자격 증명 소스 설정 ID 공급자 (IdP) 로 로그인하는 사용자로 주 엔티티를 정의하고 Amazon API Gateway에서 작업 및 리소스 엔티티를 정의합니다. API 애플리케이션에서 사용자의 그룹 멤버십으로 API 요청을 승인하도록 하려면 이 옵션을 사용하는 것이 좋습니다.
- 샘플 정책 저장소에서 시작 미리 정의된 샘플 프로젝트 정책 저장소를 선택하십시오. Verified Permissions에 대해 알아보고 예시 정책을 보고 테스트하려는 경우 이 옵션을 사용하는 것이 좋습니다.

• 빈 정책 저장소 생성 - 스키마와 모든 액세스 정책을 직접 정의합니다. 정책 스토어 구성에 이미 익숙한 경우 이 옵션을 사용하는 것이 좋습니다.

#### Guided setup

설정 안내 구성 방법을 사용하여 정책 스토어를 생성하려면

설정 안내 마법사가 정책 스토어의 첫 번째 반복을 생성하는 프로세스를 안내합니다. 이 프로세스에서는 첫 번째 리소스 유형에 대한 스키마를 만들고, 해당 리소스 유형에 적용할 수 있는 작업과 작업 권한을 부여할 보안 주체 유형을 지정합니다. 그런 다음 첫 번째 정책을 생성합니다. 이 마법사를 완료하면 정책 스토어에 정책을 추가하고, 스키마를 확장하여 다른 리소스 및 보안 주체 유형을 설명하고, 추가 정책 및 템플릿을 생성할 수 있습니다.

- 1. 검증된 권한 콘솔에서 새 정책 저장소 생성을 선택합니다.
- 2. 시작 옵션 섹션에서 가이드 설정을 선택합니다.
- 3. 정책 저장소 설명을 입력합니다. 이 텍스트는 현재 정책 저장소의 기능을 쉽게 참조할 수 있도록 조직에 적합한 텍스트 (예: 날씨 업데이트) 를 사용할 수 있습니다.
- 4. 세부 정보 섹션에서 스키마의 네임스페이스를 입력합니다.
- 5. 다음을 선택합니다.
- 6. 리소스 유형 창에서 리소스 유형의 이름을 입력합니다.
- 7. (선택 사항) 속성 추가를 선택하여 리소스 속성을 추가합니다. 속성 이름을 입력하고 리소스의 각 속성에 대해 속성 유형을 선택합니다. 각 속성이 필수인지 여부를 선택합니다. Verified Permissions는 스키마에 대해 정책을 검증할 때 지정된 속성 값을 사용합니다. 리소스 유형에추가된 속성을 제거하려면 해당 속성 옆에 있는 제거를 선택합니다.
- 8. 작업 필드에 지정된 리소스 유형에 대해 승인할 작업을 입력합니다. 리소스 유형에 대해 작업을 더 추가하려면 작업 추가를 선택합니다. 리소스 유형에 대해 추가된 작업을 제거하려면 해당 작업 옆에 있는 제거를 선택합니다.
- 9. 보안 주체 유형 이름 필드에 해당 리소스 유형에 지정된 작업을 사용할 보안 주체 유형의 이름을 입력합니다.
- 10. Next(다음)를 선택합니다.
- 11. 보안 주체 유형 창에서 보안 주체 유형의 자격 증명 소스를 선택합니다.
  - Verified Permissions 애플리케이션에서 보안 주체의 ID 및 속성이 직접 제공되도록 하려면 사용자 지정을 선택합니다. 보안 주체 속성을 추가하려면 속성 추가를 선택합니다. 속성 이름을 입력하고 보안 주체의 각 속성에 대해 속성 유형을 선택합니다. Verified Permissions는

스키마에 대해 정책을 검증할 때 지정된 속성 값을 사용합니다. 보안 주체 유형에 추가된 속성을 제거하려면 해당 속성 옆에 있는 제거를 선택합니다.

- Amazon Cognito에서 생성된 ID 또는 액세스 토큰에서 보안 주체의 ID 및 속성이 제공되도록 하려면 Cognito 사용자 풀을 선택합니다. 사용자 풀 연결을 선택합니다. AWS 리전을 선택하고 연결할 Amazon Cognito 사용자 풀의 사용자 풀 ID를 입력합니다. 연결을 선택합니다. 자세한 내용은 Amazon Cognito 개발자 안내서의 Amazon Verified Permissions를 통한 권한 부여를 참조하세요.
- 12. Next(다음)를 선택합니다.
- 13. 정책 세부 정보 섹션에서 첫 번째 Cedar 정책에 대해 선택적으로 정책 설명을 입력합니다.
- 14. 보안 주체 범위 필드에서 정책으로부터 권한을 부여받을 보안 주체를 선택합니다.
  - 정책을 특정 보안 주체에 적용하려면 특정 보안 주체를 선택합니다. 작업을 수행하도록 허용할 보안 주체 필드에서 보안 주체를 선택하고 보안 주체의 개체 식별자를 입력합니다.
  - 정책 스토어의 모든 보안 주체에 정책을 적용하려면 모든 보안 주체를 선택합니다.
- 15. 리소스 범위 필드에서 지정된 보안 주체에게 작업 권한을 부여할 리소스를 선택합니다.
  - 정책을 특정 리소스에 적용하려면 특정 리소스를 선택합니다. 이 정책을 적용해야 하는 리소 스 필드에서 리소스를 선택하고 리소스의 개체 식별자를 입력합니다.
  - 정책 스토어의 모든 리소스에 정책을 적용하려면 모든 리소스를 선택합니다.
- 16. 작업 범위 필드에서 지정된 보안 주체에게 수행할 권한을 부여할 작업을 선택합니다.
  - 정책을 여러 특정 작업에 적용하려면 특정 작업 세트를 선택합니다. 이 정책을 적용해야 하는 작업 필드에서 해당 작업 옆에 있는 확인란을 선택합니다.
  - 정책 스토어의 모든 작업에 정책을 적용하려면 모든 작업을 선택합니다.
- 17. 정책 미리 보기 섹션에서 정책을 검토하십시오. 정책 스토어 생성을 선택합니다.

#### Set up with API Gateway and an identity source

API게이트웨이 설정 및 ID 소스 구성 방법을 사용하여 정책 저장소를 만들려면

API게이트웨이 옵션은 사용자 그룹 또는 역할을 기준으로 권한 부여 결정을 내리도록 설계된 검증된 권한 정책을 통해 보안을 APIs 유지합니다. 이 옵션은 자격 증명 소스 API 그룹과 Lambda 권한 부여자를 사용하여 권한 부여를 테스트하기 위한 정책 저장소를 구축합니다.

IdP의 사용자 및 그룹은 주체 (ID 토큰) 또는 컨텍스트 (액세스 토큰) 가 됩니다. API게이트웨이의 메서드와 경로는 정책에서 API 승인하는 작업이 됩니다. 애플리케이션이 리소스가 됩니다. 이 워크

플로의 결과로 검증된 권한은 정책 스토어, Lambda 함수 및 Lambda 권한 부여자를 API 생성합니다. 이 워크플로를 완료한 후에는 Lambda 권한 부여자를 할당해야 API 합니다.

- 1. 검증된 권한 콘솔에서 새 정책 스토어 생성을 선택합니다.
- 2. 시작 옵션 섹션에서 API게이트웨이 및 ID 소스로 설정을 선택하고 다음을 선택합니다.
- 3. 리소스 및 작업 가져오기 단계에서 정책 저장소 리소스 및 작업의 모델 역할을 API 할 항목을 선택합니다. API
  - a. 에 구성된 단계에서 배포 단계를 API 선택하고 가져오기를 선택합니다API. API단계에 대한 자세한 내용은 <u>Amazon API Gateway 개발자 안내서의 단계 설정을</u> 참조하십시오. REST API
  - b. 가져온 리소스 및 작업의 맵을 미리 보십시오.
  - c. 리소스 또는 작업을 업데이트하려면 API 경로나 방법을 수정하고 Import를 선택합니다 API.
  - d. 선택에 만족하면 다음을 선택합니다.
- 4. ID 소스에서 ID 제공자 유형을 선택합니다. Amazon Cognito 사용자 풀 또는 OpenID Connect () OIDC IdP 유형을 선택할 수 있습니다.
- 5. Amazon Cognito를 선택한 경우:
  - a. 정책 AWS 리전 스토어와 AWS 계정 동일한 플랫폼에서 사용자 풀을 선택하십시오.
  - b. 승인을 위해 API 제출하려는 토큰에 전달할 토큰 유형을 선택합니다. 두 토큰 유형 모두 사용자 그룹을 포함하며, 이는 이러한 API 연결 권한 부여 모델의 기반입니다.
  - c. 앱 클라이언트 검증에서 정책 스토어의 범위를 멀티 테넌트 사용자 풀의 Amazon Cognito 앱 클라이언트의 하위 집합으로 제한할 수 있습니다. 사용자 풀에 있는 하나 이상의 지정 된 앱 클라이언트에서 사용자를 인증하도록 하려면 예상 앱 클라이언트가 있는 토큰만 수 락을 선택합니다. IDs 사용자 풀로 인증하는 모든 사용자를 수락하려면 앱 클라이언트 검 증 안 함을 선택합니다. IDs
  - d. Next(다음)를 선택합니다.
- 6. OIDC공급자를 선택한 경우:
  - a. 발급자에 URL URL 발급자의 이름을 입력합니다. OIDC 이는 권한 부여 서버, 서명 키 및 제공자에 대한 기타 정보 (예:) 를 제공하는 서비스 엔드포인트입니다. https://auth.example.com 발급자는 에서 OIDC /.well-known/openid-configuration 검색 문서를 URL 호스팅해야 합니다.

b. 토큰 유형에서 신청서가 승인을 위해 제출하려는 유형을 선택합니다. OIDC JWT 자세한 내용은 ID 제공자 토큰을 스키마에 매핑 단원을 참조하십시오.

- c. 토큰 클레임에서 정책 저장소에 사용자 속성을 설정하는 방법을 선택합니다. 이러한 속성은 정책에서 참조할 수 있는 클레임을 정의합니다.
  - i. 클레임 출처를 선택하세요.
    - A. 샘플 토큰을 제공하려면 JWT페이로드에서 추출을 선택하고 선택한 토큰 유형의 페이로드를 붙여넣습니다. JWT JWTs헤더, 페이로드 및 서명을 포함합니다. 샘플은 디코딩되고 페이로드 JWT 전용이어야 합니다. 페이로드를 파싱하려면 추출을 선택합니다.
    - B. 고유한 속성 세트를 입력하려면 클레임 수동 입력을 선택합니다.
  - ii. 스키마의 사용자 주체 또는 작업 컨텍스트의 속성에 추가할 각 토큰 클레임 이름 및 클레임 값 유형을 입력하거나 확인합니다.
- d. 사용자 및 그룹 클레임에서 ID 소스에 대한 사용자 클레임을 선택합니다. 이는 일반적으로 sub 평가 대상 개체의 고유 식별자가 있는 ID 또는 액세스 토큰에서 제기되는 클레임입니다. 연결된 OIDC IdP의 ID는 정책 저장소의 사용자 유형에 매핑됩니다.
- e. 사용자 및 그룹 클레임에서 ID 소스에 대한 그룹 클레임을 선택합니다. 이는 일반적으로 groups 사용자 그룹 목록이 포함된 ID 또는 액세스 토큰에서 제기되는 클레임입니다. 정책 저장소는 그룹 멤버십을 기반으로 요청을 승인합니다.
- f. 대상 유효성 검사 또는 클라이언트에서 IDs 정책 저장소가 승인 요청에서 수락하도록 하려는 클라이언트 IDs 또는 대상 (URLs있는 경우) 을 입력합니다. 액세스 토큰의 경우 오디언스 클레임 값을 와 같이 입력합니다https://myapp.example.com. ID 토큰의 경우다음과 같은 클라이언트 ID를 입력합니다1example23456789.
- g. Next(다음)를 선택합니다.
- 7. Amazon Cognito를 선택한 경우 검증된 권한은 사용자 풀에서 그룹을 쿼리합니다. OIDC공급 자의 경우 그룹 이름을 수동으로 입력합니다. 그룹에 작업 할당 단계는 그룹 구성원이 작업을 수행할 수 있도록 허용하는 정책 저장소용 정책을 생성합니다.
  - a. 정책에 포함하려는 그룹을 선택하거나 추가합니다.
  - b. 선택한 각 그룹에 작업을 할당합니다.
  - c. Next(다음)를 선택합니다.
- 8. 앱 통합 배포에서 검증된 권한이 정책 저장소와 Lambda 권한 부여자를 생성하기 위해 수행하는 단계를 검토하십시오.
- 9. 새 리소스를 생성할 준비가 되면 [Create and deploy] 를 선택합니다.

10. 브라우저에서 정책 저장소 상태 단계를 열어 두고 Verified Permissions에 의한 리소스 생성 진행 상황을 모니터링하세요.

11. 보통 한 시간 정도 시간이 지난 후 또는 Lambda 권한 부여자 배포 단계에서 성공으로 표시되면 권한 부여자를 구성하십시오.

검증된 권한은 Lambda 함수와 Lambda 권한 부여자를 사용자 내에 생성하게 됩니다. API [Open API] 을 선택하여 해당 페이지로 이동하십시오. API

Lambda 권한 부여자를 할당하는 방법을 알아보려면 Amazon Gateway 개발자 안내서의 <u>API</u> 게이트웨이 Lambda 권한 부여자 사용을 참조하십시오. API

- a. 담당 승인자로 이동하여 검증된 권한이 생성한 승인자의 이름을 API 기록해 두십시오.
- b. 리소스로 이동하여 해당 리소스의 최상위 방법을 선택하십시오. API
- c. 메서드 요청 설정에서 편집을 선택합니다.
- d. 권한 부여자를 이전에 기록해 둔 권한 부여자 이름으로 설정합니다.
- e. HTTP요청 헤더를 확장하고 이름 또는 를 입력한 다음 필수를 AUTHORIZATION 선택합니 다.
- f. API스테이지를 배포합니다.
- g. 변경 내용을 저장합니다.
- 12. ID 소스 선택 단계에서 선택한 토큰 유형의 사용자 풀 토큰으로 권한 부여자를 테스트하십시오. 사용자 풀 로그인 및 토큰 검색에 대한 자세한 내용은 Amazon Cognito 개발자 <u>안내서의 사용자 풀 인증 흐름을 참조하십시오.</u>
- 13. 요청 AUTHORIZATION 헤더에 사용자 풀 토큰을 넣어 인증을 다시 테스트하십시오. API
- 14. 새 정책 저장소를 살펴보세요. 정책을 추가하고 수정하십시오.

#### Sample policy store

샘플 정책 스토어 구성 방법을 사용하여 정책 스토어를 생성하려면

- 시작 옵션 섹션에서 샘플 정책 저장소를 선택합니다.
- 2. 샘플 프로젝트 섹션에서 사용할 샘플 Verified Permissions 애플리케이션의 유형을 선택합니다.
  - PhotoFlash는 사용자가 개별 사진 및 앨범을 친구와 공유할 수 있는 샘플 고객 대상 웹 애플리케이션입니다. 사용자는 사진을 보거나 댓글을 달고 재공유할 수 있는 사람의 권한을 세분

> 화하여 설정할 수 있습니다. 계정 소유자는 친구 그룹을 만들고 사진을 앨범으로 정리할 수 도 있습니다.

• DigitalPetStore누구나 등록하고 고객이 될 수 있는 샘플 애플리케이션입니다. 고객은 판매할 반려동물을 추가하거나 반려동물을 검색하고 주문할 수 있습니다. 반려동물을 추가한 고객 은 반려동물 소유자로 기록됩니다. 반려동물 소유자는 반려동물의 세부 정보를 업데이트하 거나 반려동물 이미지를 업로드하고 반려동물 등록 내용을 삭제할 수 있습니다. 반려동물을 주문한 고객은 주문 소유자로 기록됩니다. 주문 소유자는 주문에 대한 세부 정보를 확인하거 나 주문을 취소할 수 있습니다. 반려동물 스토어 매니저에게는 관리자 액세스 권한이 부여됩 니다.

#### Note

DigitalPetStore샘플 정책 저장소에는 정책 템플릿이 포함되어 있지 않습니다. PhotoFlash 및 TinyTodo샘플 정책 저장소에는 정책 템플릿이 포함되어 있습니다.

- TinyTodo사용자가 작업 및 작업 목록을 만들 수 있는 샘플 애플리케이션입니다. 목록 소유자 는 목록을 관리 및 공유하고 목록을 보거나 편집할 수 있는 사람을 지정할 수 있습니다.
- 3. 샘플 정책 스토어의 스키마 네임스페이스는 선택한 샘플 프로젝트에 따라 자동으로 생성됩니 다.
- 정책 스토어 생성을 선택합니다. 4.

선택한 샘플 정책 스토어에 대한 정책 및 스키마를 사용하여 정책 스토어가 생성됩니다. 샘플 정책 스토어에 대해 생성할 수 있는 템플릿이 연결된 정책에 대한 자세한 내용은 Amazon 검증 권한 예제 템플릿 연결 정책를 참조하세요.

#### Empty policy store

빈 정책 스토어 구성 방법을 사용하여 정책 스토어를 생성하려면

- 1. 시작 옵션 섹션에서 정책 저장소 비우기를 선택합니다.
- 2. 정책 스토어 생성을 선택합니다.

빈 정책 스토어는 스키마 없이 생성되므로 정책이 검증되지 않습니다. 정책 스토어의 스키마 업데 이트에 대한 자세한 내용은 Amazon Verified Permissions 정책 스토어 스키마를 참조하세요.

정책 스토어의 정책 생성에 대한 자세한 내용은 Amazon Verified Permissions 정적 정책 생성 및 Amazon 검증 권한 템플릿 연결 정책 생성을 참조하세요.

#### **AWS CLI**

AWS CLI를 사용하여 빈 정책 스토어를 생성하려면

create-policy-store 작업을 사용하여 정책 스토어를 생성할 수 있습니다.

### Note

를 사용하여 생성한 정책 AWS CLI 저장소는 비어 있습니다.

- 스키마를 추가하려면 Amazon Verified Permissions 정책 스토어 스키마를 참조하세요.
- 정책을 추가하려면 Amazon Verified Permissions 정적 정책 생성을 참조하세요.
- 정책 템플릿을 추가하려면 Amazon 검증 권한 정책 템플릿 생성을 참조하세요.

```
$ aws verifiedpermissions create-policy-store \
        --validation-settings "mode=STRICT"
{
        "arn": "arn:aws:verifiedpermissions::123456789012:policy-store/
PSEXAMPLEabcdefg111111",
        "createdDate": "2023-05-16T17:41:29.103459+00:00",
        "lastUpdatedDate": "2023-05-16T17:41:29.103459+00:00",
        "policyStoreId": "PSEXAMPLEabcdefg111111"
}
```

#### **AWS SDKs**

를 사용하여 정책 저장소를 생성할 수 CreatePolicyStore API 있습니다. 자세한 내용은 CreatePolicyStoreAmazon 검증 권한 API 참조 안내서를 참조하십시오.

# API-연결된 정책 저장소

Amazon Verified Permissions 콘솔에서 새 정책 스토어를 생성할 때 API게이트웨이 및 ID 소스로 설정 옵션을 선택할 수 있습니다. 이 옵션을 사용하면 Amazon Cognito 사용자 풀 또는 ID 공급자 (IdP) 로 인 증하고 Amazon Gateway에서 데이터를 가져오는 애플리케이션을 위한 권한 부여 OIDC 모델인 API-linked 정책 스토어를 구축할 수 있습니다. API APIs 시작하려면 ID 공급자와 함께 API Gateway를 사용하기 위한 정책 저장소 생성 섹션을 참조하십시오.

주제

사용자 가이드 **Amazon Verified Permissions** 

- 검증된 권한이 요청을 승인하는 방법 API
- API-연결된 정책 저장소에 대한 고려 사항
- 속성 기반 액세스 제어 추가 () ABAC
- 를 통해 프로덕션으로 이동 AWS CloudFormation
- 연결된 정책 저장소의 문제 해결 API

#### Important

검증된 권한 콘솔의 API게이트웨이 설정 및 자격 증명 소스 옵션을 사용하여 생성한 정책 스 토어는 프로덕션에 즉시 배포하기 위한 것이 아닙니다. 초기 정책 저장소를 사용하여 권한 부여 모델을 마무리하고 정책 저장소 리소스를 로 CloudFormation 내보냅니다. AWSCloud Development Kit (CDK) 를 사용하여 프로그래밍 방식으로 검증된 권한을 프로덕션에 배포할 수 있습니다. 자세한 내용은 를 통해 프로덕션으로 이동 AWS CloudFormation 단원을 참조하 십시오.

ID 소스에 연결된 정책 저장소에서 애플리케이션은 요청 시 권한 부여 헤더에 사용자 풀 토큰을 제 공합니다. API API 정책 저장소의 ID 소스는 검증된 권한에 대한 토큰 검증을 제공합니다. 토큰은 와 함께 principal IN 권한 부여 요청을 구성합니다 IsAuthorizedWithTokenAPI. 검증된 권한은 ID (ID) 및 액세스 토큰에 대한 그룹 클레임 (예: 사용자 풀) 에 명시된 바와 같이 사용자의 그룹 멤버십에 cognito:groups 대한 정책을 구축합니다. Lambda 권한 부여자에서 애플리케이션의 토큰을 API 처 리하고 인증 결정을 위해 검증된 권한에 제출합니다. Lambda 권한 부여자로부터 승인 결정을 API 받 으면 해당 요청을 데이터 소스로 전달하거나 요청을 거부합니다.

검증 권한이 있는 ID 소스 및 API 게이트웨이 권한 부여의 구성 요소

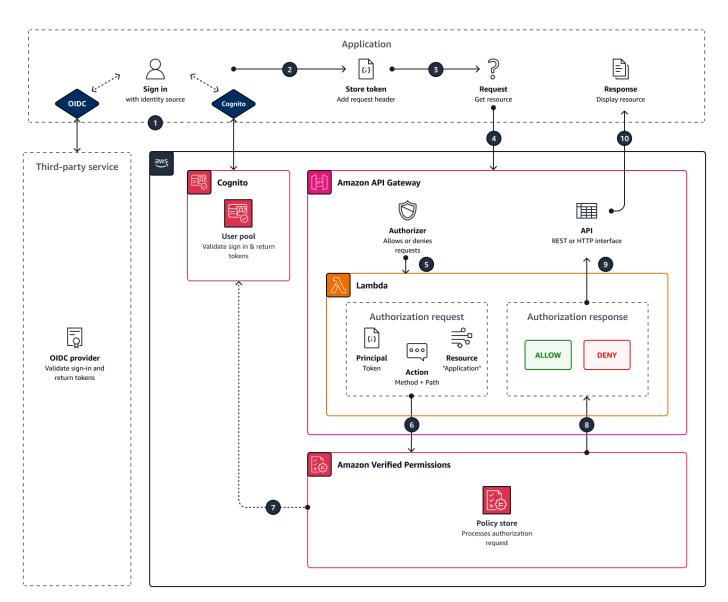
- 사용자를 인증하고 그룹화하는 Amazon Cognito 사용자 풀 또는 OIDC IdP. 사용자 토큰은 그룹 멤버 십과 검증 권한이 정책 저장소에서 평가하는 보안 주체 또는 컨텍스트를 채웁니다.
- 게이트웨이. API REST API 검증된 권한은 예를 들어 API 경로와 API 메서드의 작업을 정의합니 다MyAPI::Action::get /photo.
- 사용자를 위한 Lambda 함수 및 Lambda 권한 부여자 API Lambda 함수는 사용자 풀에서 베어러 토 큰을 가져와 검증된 권한으로부터 승인을 요청하고 Gateway에 결정을 반환합니다. API Cognito 및 API Gateway로 설정 워크플로우는 이 Lambda 권한 부여자를 자동으로 생성합니다.
- 검증된 권한 정책 스토어, 정책 저장소 ID 소스는 사용자 풀입니다. 정책 저장소 스키마는 사용자 구 성을 API 반영하며 정책은 사용자 그룹을 허용된 API 작업에 연결합니다.

API-연결된 정책 저장소

• IdP로 사용자를 인증하고 요청에 토큰을 추가하는 애플리케이션입니다. API

## 검증된 권한이 요청을 승인하는 방법 API

새 정책 저장소를 생성하고 Cognito 및 API Gateway로 설정 옵션을 선택하면 검증된 권한이 정책 저장소 스키마와 정책을 생성합니다. 스키마와 정책은 API 작업을 수행할 수 있도록 권한을 부여하려는 작업과 사용자 풀 그룹을 반영합니다. <u>또한 검증된 권한은 Lambda 함수 및 권한 부여자를 생성합니다.</u>의 메서드에 새 권한 부여자를 구성해야 합니다. API



1. 사용자는 Amazon Cognito 또는 다른 OIDC IdP를 통해 애플리케이션에 로그인합니다. IdP는 사용 자 정보와 함께 ID 및 액세스 토큰을 발급합니다.

· 작동 방식 40

2. 애플리케이션은 를 저장합니다. JWTs 자세한 내용은 Amazon Cognito 개발자 <u>안내서의 사용자 풀</u>과 함께 토큰 사용을 참조하십시오.

- 3. 사용자는 애플리케이션이 외부에서 API 검색해야 하는 데이터를 요청합니다.
- 4. 애플리케이션은 핀 REST API API 게이트웨이에서 데이터를 요청합니다. ID 또는 액세스 토큰을 요 청 헤더로 추가합니다.
- 5. 승인 결정을 위한 API 캐시가 있는 경우 이전 응답을 반환합니다. 캐싱이 비활성화되거나 에 현재 캐 시가 없는 경우 API API Gateway는 요청 파라미터를 <u>토큰 기반 Lambda</u> 권한 부여자에게 전달합니다.
- 6. Lambda 함수는 와 함께 검증된 권한 정책 스토어에 권한 부여 요청을 보냅니다. IsAuthorizedWithTokenAPI Lambda 함수는 권한 부여 결정의 요소를 전달합니다.
  - a. 사용자 토큰이 보안 주체입니다.
  - b. API경로와 결합된 API 메서드 (GetPhoto예: 액션)
  - c. Application리소스로서의 용어.
- 7. 검증된 권한은 토큰을 검증합니다. Amazon Cognito 토큰을 검증하는 방법에 대한 자세한 내용은 Amazon Cognito 개발자 안내서의 Amazon 검증 권한을 통한 권한 부여를 참조하십시오.
- 8. 검증된 권한은 정책 스토어의 정책을 기준으로 권한 부여 요청을 평가하여 승인 결정을 반환합니다.
- 9. Lambda 권한 부여자는 또는 응답을 게이트웨이에 Allow 반환합니다Deny. API
- 10.는 API 애플리케이션에 대한 데이터 또는 ACCESS\_DENIED 응답을 반환합니다. 애플리케이션은 API 요청을 처리하고 결과를 표시합니다.

## API-연결된 정책 저장소에 대한 고려 사항

Verified Permissions 콘솔에서 API 연결된 정책 저장소를 구축하면 최종 프로덕션 배포를 위한 테스트가 생성됩니다. 프로덕션으로 전환하기 전에 사용자 API 풀과 사용자 풀에 대한 고정 구성을 설정하세요. 다음 요소를 고려하십시오.

API게이트웨이는 응답을 캐시합니다.

API연결된 정책 저장소에서 검증된 권한은 120초의 권한 부여 캐시를 사용하여 Lambda 권한 부여 자를 생성합니다. TTL 권한 부여자에서 이 값을 조정하거나 캐싱을 해제할 수 있습니다. 캐싱이 활성화된 권한 부여자의 경우 승인자는 만료될 때까지 매번 동일한 응답을 반환합니다. TTL 이렇게하면 요청된 단계의 TTL 캐싱과 동일한 기간만큼 사용자 풀 토큰의 유효 수명을 연장할 수 있습니다.

고려 사항 41

Amazon Cognito 그룹은 재사용할 수 있습니다.

Amazon Verified Permissions는 사용자 ID 또는 액세스 토큰에 대한 cognito:groups 클레임을 기반으로 사용자 풀 사용자의 그룹 구성원 자격을 결정합니다. 이 클레임의 가치는 사용자가 속한 사용자 풀 그룹의 친숙한 이름의 배열입니다. 사용자 풀 그룹을 고유 식별자와 연결할 수 없습니다.

동일한 이름으로 삭제하고 다시 생성한 사용자 풀 그룹은 정책 저장소에 동일한 그룹으로 표시됩니다. 사용자 풀에서 그룹을 삭제하는 경우 해당 그룹에 대한 모든 참조를 정책 저장소에서 삭제하십시오.

API-파생된 네임스페이스와 스키마는 다음과 같습니다. point-in-time

Verified API Permissions는 특정 시점의 사용자를 캡처합니다. 즉, 정책 저장소를 생성할 API 때만 사용자를 쿼리합니다. 스키마나 이름이 API 변경되면 정책 스토어와 Lambda 권한 부여자를 업데 이트하거나 -linked 정책 저장소를 새로 생성해야 합니다. API 검증된 권한은 사용자의 이름에서 정책 스토어 네임스페이스를 가져옵니다. API

Lambda 함수에는 구성이 없습니다. VPC

인증된 권한이 권한 부여자를 위해 생성하는 Lambda 함수는 a에 API 연결되어 있지 않습니다. VPC 기본값입니다. APIs네트워크 액세스가 비공개로 제한된 경우 검증된 권한으로 액세스 요청을 승인하는 Lambda 함수와 VPCs 통신할 수 없습니다.

검증된 권한은 권한 부여자 리소스를 다음 위치에 배포합니다. CloudFormation

API-linked 정책 저장소를 생성하려면 권한이 높은 AWS 보안 주체를 Verified Permissions 콘솔에로그인해야 합니다. 이 사용자는 여러 곳에 걸쳐 리소스를 생성하는 AWS CloudFormation 스택을 배포합니다. AWS 서비스이 보안 주체는 검증된 권한 IAM, Lambda 및 Gateway에서 리소스를 추가하고 수정할 권한이 있어야 합니다. API 가장 좋은 방법은 이러한 자격 증명을 조직의 다른 관리자와 공유하지 않는 것입니다.

검증된 권한이 생성하는 리소스의 개요는 를 <u>를 통해 프로덕션으로 이동 AWS CloudFormation</u> 참 조하십시오.

# 속성 기반 액세스 제어 추가 () ABAC

IdP를 사용한 일반적인 인증 세션은 ID 및 액세스 토큰을 반환합니다. 애플리케이션 요청 시 이러한 토 큰 유형 중 하나를 전달자 토큰으로 전달할 수 있습니다. API 정책 저장소를 생성할 때 선택한 사항에 따라 Verified Permissions에는 두 가지 유형의 토큰 중 하나가 필요합니다. 두 유형 모두 사용자의 그룹 구성원에 대한 정보를 담고 있습니다. Amazon Cognito의 토큰 유형에 대한 자세한 내용은 Amazon Cognito 개발자 <u>안내서의 사용자 풀과 함께 토큰 사용을</u> 참조하십시오.

추가 ABAC 42

정책 저장소를 생성한 후 정책을 추가하고 확장할 수 있습니다. 예를 들어 사용자 풀에 새 그룹을 추가할 때 정책에 새 그룹을 추가할 수 있습니다. 정책 저장소는 사용자 풀이 그룹을 토큰으로 표시하는 방식을 이미 알고 있으므로 새 정책을 사용하여 모든 새 그룹에 대해 일련의 작업을 허용할 수 있습니다.

그룹 기반 정책 평가 모델을 사용자 속성을 기반으로 하는 보다 정확한 모델로 확장할 수도 있습니다. 사용자 풀 토큰에는 권한 부여 결정에 기여할 수 있는 추가 사용자 정보가 포함되어 있습니다.

#### ID 토큰

ID 토큰은 사용자의 속성을 나타내며 최고 수준의 세분화된 액세스 제어를 제공합니다. 이메일 주소, 전화번호 또는 사용자 지정 특성 (예: 부서 및 관리자) 을 평가하려면 ID 토큰을 평가하세요.

#### 액세스 토큰

액세스 토큰은 범위가 OAuth 2.0인 사용자의 권한을 나타냅니다. 권한 부여 계층을 추가하거나 추가 리소스에 대한 요청을 설정하려면 액세스 토큰을 평가하십시오. 예를 들어, 사용자가 적절한 그룹에 속해 있고 일반적으로 액세스를 PetStore.read 승인하는 것과 같은 범위를 가지고 있는지확인할 수 있습니다. API 사용자 풀은 런타임 시토큰 사용자 지정과 함께 리소스 서버가 있는 토큰에 사용자 지정 범위를 추가할 수 있습니다.

ID 및 액세스 토큰으로 클레임을 처리하는 정책의 ID 제공자 토큰을 스키마에 매핑 예를 참조하십시오.

## 를 통해 프로덕션으로 이동 AWS CloudFormation

API-연결된 정책 저장소는 게이트웨이의 권한 부여 모델을 빠르게 구축할 수 있는 API 방법입니다. API 애플리케이션의 권한 부여 구성 요소에 대한 테스트 환경으로 사용할 수 있도록 설계되었습니다. 테스트 정책 스토어를 생성한 후에는 정책, 스키마 및 Lambda 권한 부여자를 개선하는 데 시간을 할애 하십시오.

아키텍처를 조정하여 정책 스토어 스키마 API 및 정책을 동등하게 조정하도록 요구할 수 있습니다. API-연결된 정책 저장소는 API 아키텍처에서 스키마를 자동으로 업데이트하지 않습니다. 검증된 권한은 정책 저장소를 생성할 때만 스키마를 폴링합니다. API 충분히 API 변경한 경우 새 정책 저장소를 사용하여 프로세스를 반복해야 할 수도 있습니다.

애플리케이션 및 권한 부여 모델을 프로덕션에 배포할 준비가 되면 개발한 API -link 정책 저장소를 자동화 프로세스와 통합하십시오. 가장 좋은 방법은 정책 저장소 스키마와 정책을 다른 AWS 계정 사이트에 배포할 수 있는 AWS CloudFormation 템플릿으로 내보내는 것입니다. AWS 리전

API연결된 정책 저장소 프로세스의 결과는 초기 정책 저장소와 Lambda 권한 부여자입니다. Lambda 권한 부여자에는 여러 종속 리소스가 있습니다. 검증된 권한은 이러한 리소스를 자동으로 생성된 스택

프로덕션으로 이동 43

에 배포합니다. CloudFormation 프로덕션에 배포하려면 정책 스토어와 Lambda 권한 부여자 리소스를 템플릿으로 수집해야 합니다. API연결된 정책 스토어는 다음 리소스로 구성됩니다.

- 1. <u>AWS::VerifiedPermissions:: PolicyStore</u>: 스키마를 SchemaDefinition 객체에 복사합니다. 이스 케이프 " 문자는 다음과 같습니다\".
- 2. <u>AWS::VerifiedPermissions:: IdentitySource</u>: 테스트 정책 저장소의 <u>GetIdentitySource</u> 출력에서 값을 복사하고 필요에 따라 수정합니다.
- 3. <u>AWS:: VerifiedPermissions :Policy 중 하나 이상: 정책</u> 설명을 Definition 객체에 복사합니다. 이 스케이프 " 문자는 다음과 같습니다. \"
- 4. AWS: :Lambda: :함수, AWS::IAM: :역할, AWS::: 정책,IAM::AWS: Authorizer,ApiGateway: :Lambda AWS: :권한: 정책 저장소를 생성할 때 검증된 권한이 배포된 스택의 템플릿 탭에서 템플릿을 복사합니다.

다음 템플릿은 정책 저장소의 예시입니다. 기존 스택의 Lambda 권한 부여자 리소스를 이 템플릿에 추가할 수 있습니다.

```
{
    "AWSTemplateFormatVersion": "2010-09-09",
    "Resources": {
        "MyExamplePolicyStore": {
            "Type": "AWS::VerifiedPermissions::PolicyStore",
            "Properties": {
                "ValidationSettings": {
                    "Mode": "STRICT"
                "Description": "ApiGateway: PetStore/test",
                "Schema": {
                    "CedarJson": "{\"PetStore\":{\"actions\":{\"get /pets\":
{\"appliesTo\":{\"principalTypes\":[\"User\"],\"resourceTypes\":[\"Application\"],
\"context\":{\"type\":\"Record\",\"attributes\":{}}}},\"get /\":{\"appliesTo\":
{\"principalTypes\":[\"User\"],\"resourceTypes\":[\"Application\"],\"context\":{\"type
\":\"Record\",\"attributes\":{}}}},\"get /pets/{petId}\":{\"appliesTo\":{\"context
\":{\"type\":\"Record\",\"attributes\":{}},\"resourceTypes\":[\"Application\"],
\"principalTypes\":[\"User\"]}},\"post /pets\":{\"appliesTo\":{\"principalTypes\":
[\"User\"],\"resourceTypes\":[\"Application\"],\"context\":{\"type\":\"Record\",
\"attributes\":{}}}}},\"entityTypes\":{\"Application\":{\"shape\":{\"type\":\"Record\",
\"attributes\":{}}},\"User\":{\"memberOfTypes\":[\"UserGroup\"],\"shape\":{\"attributes
\":{},\"type\":\"Record\"}},\"UserGroup\":{\"shape\":{\"type\":\"Record\",\"attributes
\":{}}}}}"
                }
```

프로덕션으로 이동 44

```
}
        },
        "MyExamplePolicy": {
            "Type": "AWS::VerifiedPermissions::Policy",
            "Properties": {
                "Definition": {
                    "Static": {
                        "Description": "Policy defining permissions for testgroup
 cognito group",
                        "Statement": "permit(\nprincipal in PetStore::UserGroup::
\"us-east-1_EXAMPLE|testgroup\",\naction in [\n PetStore::Action::\"get /\",
\n PetStore::Action::\"post /pets\",\n PetStore::Action::\"get /pets\",\n
 PetStore::Action::\"get /pets/{petId}\"\n],\nresource);"
                },
                "PolicyStoreId": {
                    "Ref": "MyExamplePolicyStore"
                }
            },
            "DependsOn": [
                "MyExamplePolicyStore"
            ]
        },
        "MyExampleIdentitySource": {
            "Type": "AWS::VerifiedPermissions::IdentitySource",
            "Properties": {
                "Configuration": {
                    "CognitoUserPoolConfiguration": {
                        "ClientIds": [
                             "1example23456789"
                        ],
                        "GroupConfiguration": {
                             "GroupEntityType": "PetStore::UserGroup"
                        },
                        "UserPoolArn": "arn:aws:cognito-idp:us-
east-1:123456789012:userpool/us-east-1_EXAMPLE"
                    }
                },
                "PolicyStoreId": {
                    "Ref": "MyExamplePolicyStore"
                },
                "PrincipalEntityType": "PetStore::User"
            },
            "DependsOn": [
```

프로덕션으로 이동 45

## 연결된 정책 저장소의 문제 해결 API

여기의 정보를 사용하면 Amazon Verified Permissions API 연결 정책 스토어를 구축할 때 흔히 발생하는 문제를 진단하고 해결하는 데 도움이 됩니다.

#### 주제

- 정책을 업데이트했지만 승인 결정은 바뀌지 않았습니다.
- Lambda 권한 부여자를 API 내 인증자에 연결했지만 권한 부여 요청이 생성되지 않습니다.
- 예상치 못한 승인 결정을 받았는데 권한 부여 논리를 검토해 보고 싶습니다.
- Lambda 권한 부여자로부터 로그를 찾고 싶습니다.
- 내 Lambda 권한 부여자가 존재하지 않습니다.
- 내 계정이 API 비공개로 설정되어 권한 VPC 부여자를 호출할 수 없습니다.
- 권한 부여 모델에서 추가 사용자 속성을 처리하고 싶습니다.
- 새 작업, 작업 컨텍스트 속성 또는 리소스 속성을 추가하고 싶습니다.

정책을 업데이트했지만 승인 결정은 바뀌지 않았습니다.

기본적으로 검증된 권한은 권한 부여 결정을 120초 동안 캐시하도록 Lambda 권한 부여자를 구성합니다. 2분 후에 다시 시도하거나 권한 부여자의 캐시를 비활성화하십시오. 자세한 내용은 Amazon API Gateway 개발자 안내서의 응답성 향상을 위한 API 캐싱 활성화를 참조하십시오.

Lambda 권한 부여자를 API 내 인증자에 연결했지만 권한 부여 요청이 생성되지 않습니다.

요청 처리를 시작하려면 권한 부여자를 연결한 API 스테이지를 배포해야 합니다. 자세한 내용은 Amazon API Gateway 개발자 안내서의 <u>배포를</u> 참조하십시오. REST API

예상치 못한 승인 결정을 받았는데 권한 부여 논리를 검토해 보고 싶습니다.

API-link 정책 스토어 프로세스는 권한 부여자를 위한 Lambda 함수를 생성합니다. 검증된 권한은 권한 부여 결정의 로직을 권한 부여 함수에 자동으로 빌드합니다. 정책 저장소를 생성한 후 다시 돌아가서 함수의 로직을 검토하고 업데이트할 수 있습니다.

콘솔에서 AWS CloudFormation Lambda 함수를 찾으려면 새 정책 스토어의 개요 페이지에서 배포 확인 버튼을 선택하십시오.

콘솔에서 함수를 찾을 수도 있습니다. AWS Lambda 정책 AWS 리전 저장소의 콘솔로 이동하여 접두어가 인 함수 이름을 검색합니다. AVPAuthorizerLambda API연결된 정책 저장소를 두 개 이상 생성한 경우 함수의 마지막 수정 시간을 사용하여 정책 저장소 생성과 상호 연관시키십시오.

Lambda 권한 부여자로부터 로그를 찾고 싶습니다.

Lambda 함수는 지표를 수집하고 Amazon에 호출 결과를 기록합니다. CloudWatch 로그를 검토하려면 Lambda 콘솔에서 <u>함수를 찾아</u> Monitor 탭을 선택합니다. CloudWatch 로그 보기를 선택하고 로그 그룹의 항목을 검토하십시오.

Lambda 함수 로그에 대한 자세한 내용은 개발자 안내서의 <u>CloudWatch Amazon Logs AWS Lambda</u> 사용을 참조하십시오.AWS Lambda

내 Lambda 권한 부여자가 존재하지 않습니다.

API연결된 정책 저장소의 설정을 완료한 후에는 Lambda 권한 부여자를 정책 저장소에 연결해야 합니다. API API게이트웨이 콘솔에서 권한 부여자를 찾을 수 없는 경우 정책 스토어의 추가 리소스가 실패했거나 아직 배포되지 않았을 수 있습니다. API-연결된 정책 저장소는 이러한 리소스를 스택에 배포합니다. AWS CloudFormation

검증된 권한은 생성 프로세스 종료 시 Check deploy라는 레이블이 있는 링크를 표시합니다. 이미 이 화면에서 벗어났다면 CloudFormation 콘솔로 가서 최근 스택에서 접두사가 붙은 이름을 검색해 보세요. AVPAuthorizer-<policy store ID> CloudFormation 스택 배포 결과에서 중요한 문제 해결 정보를 제공합니다.

CloudFormation 스택 문제 해결에 도움이 필요하면 AWS CloudFormation 사용 설명서의 <u>문제 해결을</u> CloudFormation 참조하십시오.

내 계정이 API 비공개로 설정되어 권한 VPC 부여자를 호출할 수 없습니다.

검증된 권한은 엔드포인트를 통한 Lambda 권한 부여자에 대한 액세스를 지원하지 않습니다. VPC 권한 부여자 역할을 하는 Lambda API 함수와 Lambda 함수 사이에 네트워크 경로를 열어야 합니다.

권한 부여 모델에서 추가 사용자 속성을 처리하고 싶습니다.

API-linked 정책 저장소 프로세스는 사용자 토큰의 그룹 클레임에서 검증된 권한 정책을 도출합니다. 추가 사용자 속성을 고려하도록 권한 부여 모델을 업데이트하려면 해당 속성을 정책에 통합하세요.

Amazon Cognito 사용자 풀의 ID 및 액세스 토큰에 있는 많은 클레임을 검증된 권한 정책 설명에 매핑할 수 있습니다. 예를 들어, 대부분의 사용자는 ID 토큰에 email 클레임을 가지고 있습니다. ID 소스의클레임을 정책에 추가하는 방법에 대한 자세한 내용은 을 참조하십시오ID 제공자 토큰을 스키마에 매핑.

새 작업, 작업 컨텍스트 속성 또는 리소스 속성을 추가하고 싶습니다.

API연결된 정책 스토어와 이 스토어가 생성하는 Lambda 권한 부여자는 리소스입니다. point-in-time 생성 API 당시의 사용자 상태를 반영합니다. 정책 저장소 스키마는 작업에 컨텍스트 속성을 할당하거나 기본 Application 리소스에 속성이나 상위를 할당하지 않습니다.

작업 (경로 및 메서드) 을 추가할 때는 새 작업을 인식하도록 정책 저장소를 업데이트해야 합니다. API 또한 Lambda 권한 부여자를 업데이트하여 새 작업에 대한 권한 부여 요청을 처리해야 합니다. <u>새 정책</u> 저장소로 다시 시작하거나 기존 정책 저장소를 업데이트할 수 있습니다.

기존 정책 저장소를 <u>업데이트하려면 함수를 찾으세요</u>. 자동으로 생성된 함수의 로직을 검사하고 새 작업, 속성 또는 컨텍스트를 처리하도록 업데이트하십시오. 그런 다음 새 작업과 속성을 <u>포함하도록 스키</u>마를 편집합니다.

## 정책 스토어 삭제

AWS Management Console 또는 를 사용하여 Amazon 검증 권한 정책 스토어를 삭제할 수 AWS CLI 있습니다. 정책 저장소를 삭제하면 정책 저장소의 스키마와 모든 정책이 영구적으로 삭제됩니다.

다음과 같은 이유로 정책 저장소를 삭제해야 할 수 있습니다.

- 특정 지역에서 사용 가능한 정책 저장소 할당량에 도달했습니다. 자세한 내용은 <u>리소스 할당량</u> 단원을 참조하십시오.
- 다중 테넌트 응용 프로그램에서는 더 이상 테넌트를 지원하지 않으므로 해당 정책 저장소가 더 이상 필요하지 않습니다.

### AWS Management Console

정책 스토어를 삭제하려면

- 1. 에서 검증된 권한 콘솔을 <a href="https://console.aws.amazon.com/verifiedpermissions/">https://console.aws.amazon.com/verifiedpermissions/</a>여십시오. 정책 저장소를 선택합니다.
- 2. 왼쪽의 탐색 창에서 설정을 선택합니다.

- 3. 이 정책 스토어 삭제를 선택합니다.
- 4. 텍스트 상자에 delete를 입력하고 삭제를 선택합니다.

#### **AWS CLI**

정책 스토어를 삭제하려면

delete-policy-store 작업을 사용하여 정책 스토어를 삭제할 수 있습니다.

\$ aws verifiedpermissions delete-policy-store \
 --policy-store-id PSEXAMPLEabcdefg111111

이 명령은 성공 시 출력을 생성하지 않습니다.

정책 스토어 삭제 49

# Amazon Verified Permissions 정책 스토어 스키마

<u>스키마</u>는 애플리케이션이 지원하는 엔터티 유형의 구조와 애플리케이션이 권한 부여 요청에서 제공할수 있는 작업을 선언한 것입니다.

자세한 내용은 Cedar 정책 언어 참조 가이드의 Cedar 스키마 형식을 참조하세요.

### Note

Verified Permissions에서 스키마를 사용하는 것은 선택 사항이지만 프로덕션 소프트웨어에는 스키마를 사용하는 것이 좋습니다. 새 정책을 생성할 때 Verified Permissions는 스키마를 사용하여 범위 및 조건에서 참조되는 엔터티와 속성을 검증하여 시스템 동작에 혼란을 줄 수 있는 정책 내 오타와 실수를 방지할 수 있습니다. <u>정책 검증</u>을 활성화하면 모든 새 정책이 스키마를 준수해야 합니다.

#### **AWS Management Console**

스키마를 생성하려면

- 1. 에서 검증된 권한 콘솔을 <a href="https://console.aws.amazon.com/verifiedpermissions/">https://console.aws.amazon.com/verifiedpermissions/</a>여십시오. 정책 저장소를 선택합니다.
- 2. 왼쪽에 있는 탐색 창에서 스키마를 선택합니다.
- 3. 스키마 생성을 선택합니다.

#### **AWS CLI**

- 를 사용하여 새 스키마를 제출하거나 기존 스키마를 덮어쓰려면 AWS CLI.
- 를 실행하여 정책 저장소를 생성할 수 있습니다. AWS CLI 다음 예제와 비슷한 명령입니다.

다음과 같은 Cedar 콘텐츠가 포함된 스키마를 생각해 보십시오.

```
}
            }
        },
        "entityTypes": {
            "Employee": {
                 "shape": {
                     "type": "Record",
                     "attributes": {
                         "jobLevel": {"type": "Long"},
                         "name": {"type": "String"}
                     }
                }
            }
        }
    }
}
```

먼저 를 한 줄 문자열로 이스케이프 처리한 JSON 다음 앞에 해당 데이터 유형을 선언해야 합니다. cedarJson 다음 예제에서는 스키마의 이스케이프된 버전을 포함하는 schema.json 파일의 다음 내용을 사용합니다. JSON

#### Note

이 예제는 가독성을 위해 줄바꿈 처리되었습니다. 명령이 파일을 받아들이려면 전체 파일을 한 줄에 입력해야 합니다.

```
{"cedarJson": "{\"MySampleNamespace\": {\"actions\": {\"remoteAccess\": {\"appliesTo
\":
{\"principalTypes\": [\"Employee\"]}}},\"entityTypes\": {\"Employee\": {\"shape\":
{\"attributes\": {\"jobLevel\": {\"type\": \"Long\"},\"name\": {\"type\": \"String
\"}},
\"type\": \"Record\"}}}}"}
```

```
$ aws verifiedpermissions put-schema \
    --definition file://schema.json \
    --policy-store PSEXAMPLEabcdefg11111
{
    "policyStoreId": "PSEXAMPLEabcdefg11111",
    "namespaces": [
        "MySampleNamespace"
```

```
],
"createdDate": "2023-07-17T21:07:43.659196+00:00",
"lastUpdatedDate": "2023-08-16T17:03:53.081839+00:00"
}
```

#### **AWS SDKs**

를 사용하여 정책 저장소를 생성할 수 있습니다. PutSchema API 자세한 내용은 PutSchemaAmazon 검증 권한 API 참조 안내서를 참조하십시오.

# 비주얼 모드에서 정책 스토어 스키마를 편집합니다.

Amazon 검증 권한 콘솔에서 스키마를 선택하면 비주얼 모드에 스키마를 구성하는 엔티티 유형과 작업이 표시됩니다. 이 최상위 수준 보기에서 또는 엔티티의 세부 정보 내에서 스키마 편집을 선택하여 스키마 업데이트를 시작할 수 있습니다. 중첩 레코드와 같은 일부 스키마 형식에서는 시각적 모드를 사용할 수 없습니다.

시각적 스키마 편집기는 스키마의 항목 간 관계를 보여주는 일련의 다이어그램으로 시작합니다. 다이어그램을 최대한 보려면 확장을 선택하십시오. 다음과 같은 두 가지 다이어그램을 사용할 수 있습니다.

- 작업 다이어그램 작업 다이어그램 뷰에는 정책 저장소에 구성한 주체 유형, 수행할 수 있는 작업, 작업을 수행할 수 있는 리소스가 나열됩니다. 엔티티 사이의 선은 보안 주체가 리소스에 대해 조치를 취하도록 허용하는 정책을 만들 수 있는 능력을 나타냅니다. 작업 다이어그램에 두 엔티티 간의 관계 가 나타나지 않는 경우 정책에서 두 엔티티 간의 관계를 먼저 생성해야 허용하거나 거부할 수 있습니 다. 엔티티를 선택하여 속성 개요를 확인하고 드릴다운하여 전체 세부 정보를 확인하세요. 뷰에서 자 체 연결만 있는 엔티티를 보려면 [작업 | 리소스 유형 | 보안 주체 유형] 을 기준으로 필터를 선택하면 뷰에서 엔티티를 볼 수 있습니다.
- 엔티티 유형 다이어그램 엔티티 유형 다이어그램은 보안 주체와 리소스 간의 관계에 초점을 맞춥니다. 스키마의 복잡한 중첩된 상위 관계를 이해하려면 이 다이어그램을 검토하세요. 엔티티 위에 커서를 올리면 해당 엔티티가 가지고 있는 상위 관계를 자세히 살펴볼 수 있습니다.

다이어그램 아래에는 스키마의 엔티티 유형 및 작업에 대한 목록 보기가 있습니다. 목록 보기는 특정 작업 또는 엔티티 유형의 세부 정보를 즉시 보려는 경우에 유용합니다. 원하는 엔티티를 선택하면 세 부 정보를 볼 수 있습니다.

스키마 편집 - 비주얼 52

#### 비주얼 모드에서 Verified Permissions 스키마를 편집하려면

1. 에서 검증된 권한 콘솔을 엽니다 <u>https://console.aws.amazon.com/verifiedpermissions/</u>. 정책 저장 소를 선택합니다.

- 2. 왼쪽 탐색 창에서 스키마를 선택합니다.
- 3. 비주얼 모드를 선택합니다. 엔티티 관계 다이어그램을 검토하고 스키마에 적용할 변경 사항을 계획하십시오. 선택적으로 한 엔티티별로 필터링하여 다른 엔티티와의 개별 연결을 검사할 수 있습니다.
- 4. Edit schema(스키마 편집)를 선택합니다.
- 5. 세부 정보 섹션에서 스키마의 네임스페이스를 입력합니다.
- 6. 엔터티 유형 섹션에서 새 엔터티 유형 추가를 선택합니다.
- 7. 엔터티 이름을 입력합니다.
- 8. (선택 사항) 상위 추가를 선택하여 새 엔터티가 속한 상위 엔터티를 추가합니다. 엔터티에 추가된 상위 엔터티를 제거하려면 상위 엔터티 이름 옆에 있는 제거를 선택합니다.
- 9. 엔터티에 속성을 추가하려면 속성 추가를 선택합니다. 속성 이름을 입력하고 엔터티의 각 속성에 대해 속성 유형을 선택합니다. Verified Permissions는 스키마에 대해 정책을 검증할 때 지정된 속성 값을 사용합니다. 각 속성이 필수인지 여부를 선택합니다. 엔터티에 추가된 속성을 제거하려면 해당 속성 옆에 있는 제거를 선택합니다.
- 10. 엔터티 유형 추가를 선택하여 스키마에 엔터티를 추가합니다.
- 11. 작업 섹션에서 새 작업 추가를 선택합니다.
- 12. 작업의 이름을 입력합니다.
- 13. (선택 사항) 리소스 추가를 선택하여 작업이 적용되는 리소스 유형을 추가합니다. 작업에 추가된 리소스 유형을 제거하려면 해당 리소스 유형 이름 옆에 있는 제거를 선택합니다.
- 14. (선택 사항) 보안 주체 추가를 선택하여 작업이 적용되는 보안 주체 유형을 추가합니다. 작업에 추가된 보안 주체 유형을 제거하려면 해당 보안 주체 유형 이름 옆에 있는 제거를 선택합니다.
- 15. 속성 추가를 선택하여 권한 부여 요청의 작업 컨텍스트에 추가할 수 있는 속성을 추가합니다. 속성이름을 입력하고 각 속성의 속성 유형을 선택합니다. Verified Permissions는 스키마에 대해 정책을 검증할 때 지정된 속성 값을 사용합니다. 각 속성이 필수인지 여부를 선택합니다. 작업에 추가된 속성을 제거하려면 해당 속성 옆에 있는 제거를 선택합니다.
- 16. 작업 추가를 선택합니다.
- 17. 모든 엔터티 유형과 작업을 스키마에 추가한 후 변경 사항 저장을 선택합니다.

스키마 편집 - 비주얼 53

# 모드에서 정책 저장소 스키마를 편집합니다JSON.

Amazon 검증 권한 콘솔에서 스키마를 선택하면 스키마를 구성하는 엔티티 유형과 작업이 JSON 모드에 표시됩니다. 스키마 편집을 선택하면 편집기에서 직접 스키마 JSON 코드 업데이트를 시작할 수 있습니다. JSON 업데이트하는 동안 JSON 편집기가 JSON 구문을 기준으로 코드의 유효성을 검사하고 편집하면서 오류와 경고를 식별하므로 문제를 더 쉽고 빠르게 찾을 수 있습니다. 또한 의 형식에 대해 걱정할 필요가 없습니다. JSON 업데이트한 후 형식을 선택하기만 하면 형식이 예상 JSON 형식에 맞게 업데이트됩니다. JSON

#### JSON모드에서 검증된 권한 스키마를 편집하려면

- 1. 에서 검증된 권한 콘솔을 엽니다 <u>https://console.aws.amazon.com/verifiedpermissions/</u>. 정책 저장 소를 선택합니다.
- 2. 왼쪽 탐색 창에서 스키마를 선택합니다.
- 3. JSON모드를 선택한 다음 스키마 편집을 선택합니다.
- 4. 콘텐츠 필드에 JSON 스키마의 콘텐츠를 입력합니다. 모든 구문 오류를 해결할 때까지는 스키마 업데이트 사항을 저장할 수 없습니다. 형식을 JSON 선택하여 권장 간격과 들여쓰기로 스키마 JSON 구문 형식을 지정할 수 있습니다.
- 5. Save changes(변경 사항 저장)를 선택합니다.

스키마 편집 - JSON 54

사용자 가이드 Amazon Verified Permissions

# Amazon 검증 권한 정책 검증 모드 활성화

Verified Permissions에서 정책 검증 모드를 설정하여 정책 스토어의 스키마에 대해 정책 변경 내용을 검증할지 여부를 지정할 수 있습니다.

#### Important

정책 검증 기능을 켜면 정책 또는 정책 템플릿을 생성하거나 업데이트하려는 모든 시도에 대해 정책 스토어의 스키마를 기준으로 검증합니다. Verified Permissions는 검증에 실패할 경우 요 청을 거부합니다.

#### **AWS Management Console**

정책 스토어의 정책 검증 모드를 설정하려면

- 1. 에서 검증된 권한 콘솔을 엽니다 https://console.aws.amazon.com/verifiedpermissions/. 정책 저장소를 선택합니다.
- 설정을 선택합니다. 2.
- 정책 검증 모드 섹션에서 수정을 선택합니다. 3.
- 4. 다음 중 하나를 수행합니다.
  - 정책 검증을 활성화하고 모든 정책 변경 사항을 스키마에 대해 검증하도록 하려면 엄격(권 장) 라디오 버튼을 선택합니다.
  - 정책 변경에 대한 정책 검증을 해제하려면 끄기 라디오 버튼을 선택합니다. confirm을 입 력하여 정책 업데이트가 더 이상 스키마에 대해 검증되지 않는지 확인합니다.
- 5. 변경 사항 저장을 선택합니다.

#### **AWS CLI**

정책 스토어의 검증 모드를 설정하려면

UpdatePolicyStore작업을 사용하고 ValidationSettings매개 변수에 다른 값을 지정하여 정책 저장소 의 검증 모드를 변경할 수 있습니다.

\$ aws verifiedpermissions update-policy-store \ --validation-settings "mode=OFF",

```
--policy-store-id PSEXAMPLEabcdefg111111

{
    "createdDate": "2023-05-17T18:36:10.134448+00:00",
    "lastUpdatedDate": "2023-05-17T18:36:10.134448+00:00",
    "policyStoreId": "PSEXAMPLEabcdefg111111",
    "validationSettings": {
        "Mode": "OFF"
    }
}
```

자세한 내용은 Cedar 정책 언어 참조 가이드의 <u>정책 템플릿</u>을 참조하세요.

사용자 가이드 Amazon Verified Permissions

# Amazon Verified Permissions 정책

정책이란 보안 주체가 리소스에 대해 하나 이상의 작업을 수행하는 것을 허용하거나 금지하는 구문입 니다. 각 정책은 다른 정책과 독립적으로 평가됩니다. Cedar 정책의 구조 및 평가 방법에 대한 자세한 내용은 Cedar 정책 언어 참조 가이드의 스키마에 대한 Cedar 정책 검증을 참조하세요.

#### ↑ Important

보안 주체. 리소스 및 작업을 참조하는 Cedar 정책을 작성할 때 사용자는 각 요소에 사용되는 고유 식별자를 정의할 수 있습니다. 이 경우 다음 모범 사례를 따르는 것이 좋습니다.

모든 주체 및 리소스 식별자에는 범용 고유 식별자 (UUIDs) 와 같은 값을 사용하십시오.

예를 들어, 사용자 jane이 퇴사할 때 나중에 다른 사람이 이 jane이라는 이름을 사용하도록 허용하면 새 사용자는 여전히 User::"jane"을 참조하는 정책이 부여하는 모든 항목에 자 동으로 액세스할 수 있게 됩니다. Cedar는 새 사용자와 기존 사용자를 구분할 수 없습니다. 이 사례는 보안 주체 및 리소스 식별자 모두에 적용됩니다. 정책에 이전 식별자가 남아 있어. 의도치 않게 액세스 권한을 부여하는 일이 없도록 항상 고유성이 보장되고 재사용되지 않는 식별자를 사용하십시오.

UUID엔티티에 a를 사용할 때는 뒤에//코멘트 지정자와 엔티티의 '친숙한' 이름을 사용 하는 것이 좋습니다. 이렇게 하면 정책을 더 쉽게 이해할 수 있습니다. 예: 주체 == 사용 자: :"a1b2c3d4-e5f6-a1b2-c3d4- ",//alice EXAMPLE11111

• 개인 식별 정보. 기밀 정보 또는 민감한 정보를 보안 주체 또는 리소스에 대한 고유 식별 자의 일부로 포함하지 마십시오. 이러한 식별자는 공유된 로그 항목에 포함됩니다. AWS CloudTrail 트레일.

#### 주제

- Amazon 검증 권한의 정책 내 엔티티 형식 지정
- Amazon Verified Permissions 정적 정책 생성
- Amazon Verified Permissions 정적 정책 편집
- Amazon 검증 권한 테스트 벤치 사용
- Amazon Verified Permissions 정책 예제

# Amazon 검증 권한의 정책 내 엔티티 형식 지정

Amazon Verified Permissions는 Cedar 정책 언어를 사용하여 정책을 생성합니다. 정책 구문 및 지원되는 데이터 유형은 Cedar의 <u>기본 정책 구성에 설명된 구문 및 데이터 유형과 Cedar 정책 언어 참조 가이으의 Cedar</u> <u>주제에서 지원하는 데이터 유형에 설명된 구문 및 데이터 유형과</u> 일치합니다. 그러나 인증을 요청하는 경우 Verified Permissions와 Cedar의 엔터티 형식에는 차이가 있습니다.

검증된 권한의 엔티티 JSON 형식은 다음과 같은 점에서 Cedar와 다릅니다.

- 검증된 권한에서는 JSON 개체의 모든 키-값 쌍이 이름이 인 JSON 개체로 둘러싸여 있어야 합니다.
   Record
- 검증된 권한의 JSON 목록은 키 이름과 값이 Cedar의 원래 JSON 목록인 JSON 키-값 쌍으로 래핑되어야 합니다. Set
- StringLong, 및 Boolean 유형 이름의 경우 Cedar의 각 키-값 쌍은 검증된 권한의 개체로 대체됩니다. JSON 객체 이름은 원래 키 이름입니다. JSON객체 내에는 키-값 쌍이 하나 있는데, 여기서 키이름은 스칼라 값 (StringLong, 또는Boolean) 의 유형 이름이고 값은 Cedar 엔티티의 값입니다.
- Cedar 엔터티와 Verified Permissions 엔터티의 구문 형식은 다음과 같은 측면에서 다릅니다.

Cedar 형식	Verified Permissions 형식
uid	Identifier
type	EntityType
id	EntityId
attrs	Attributes
parents	Parents

#### Example - 목록

다음 예는 엔티티 목록이 Cedar 및 검증된 권한으로 각각 어떻게 표현되는지 보여줍니다.

#### Cedar

```
[
{
```

엔터티 형식 58

```
"number": 1
},
{
    "sentence": "Here is an example sentence"
},
{
    "Question": false
}
```

#### Verified Permissions

```
{
  "Set": [
    {
      "Record": {
        "number": {
          "Long": 1
        }
      }
    },
      "Record": {
        "sentence": {
          "String": "Here is an example sentence"
        }
      }
    },
      "Record": {
        "question": {
          "Boolean": false
        }
      }
  ]
}
```

### Example - 정책 평가

다음 예는 Cedar와 Verified Permissions의 권한 부여 요청에서 정책을 평가하기 위해 엔티티의 형식을 각각 지정하는 방법을 보여줍니다.

엔터티 형식 59

#### Cedar

```
Г
    {
        "uid": {
            "type": "PhotoApp::User",
            "id": "alice"
        },
        "attrs": {
            "age": 25,
            "name": "alice",
            "userId": "123456789012"
        },
        "parents": [
            {
                "type": "PhotoApp::UserGroup",
                "id": "alice_friends"
            },
            {
                "type": "PhotoApp::UserGroup",
                "id": "AVTeam"
            }
        ]
    },
    {
        "uid": {
            "type": "PhotoApp::Photo",
            "id": "vacationPhoto.jpg"
        },
        "attrs": {
            "private": false,
            "account": {
                "__entity": {
                     "type": "PhotoApp::Account",
                     "id": "ahmad"
                }
            }
        },
        "parents": []
    },
    {
        "uid": {
            "type": "PhotoApp::UserGroup",
            "id": "alice_friends"
```

엔터티 형식 60

```
},
    "attrs": {},
    "parents": []
},
{
    "uid": {
        "type": "PhotoApp::UserGroup",
        "id": "AVTeam"
},
    "attrs": {},
    "parents": []
}
```

#### Verified Permissions

```
Γ
    {
        "Identifier": {
            "EntityType": "PhotoApp::User",
            "EntityId": "alice"
        },
        "Attributes": {
            "age": {
                "Long": 25
            },
            "name": {
                "String": "alice"
            },
            "userId": {
                "String": "123456789012"
            }
        },
        "Parents": [
            {
                "EntityType": "PhotoApp::UserGroup",
                "EntityId": "alice_friends"
            },
            {
                "EntityType": "PhotoApp::UserGroup",
                "EntityId": "AVTeam"
            }
```

에터티 형식 61

```
},
    {
        "Identifier": {
            "EntityType": "PhotoApp::Photo",
            "EntityId": "vacationPhoto.jpg"
        },
        "Attributes": {
            "private": {
                "Boolean": false
            },
            "account": {
                "EntityIdentifier": {
                     "EntityType": "PhotoApp::Account",
                    "EntityId": "ahmad"
                }
            }
        },
        "Parents": []
    },
    {
        "Identifier": {
            "EntityType": "PhotoApp::UserGroup",
            "EntityId": "alice_friends"
        },
        "Parents": []
    },
    {
        "Identifier": {
            "EntityType": "PhotoApp::UserGroup",
            "EntityId": "AVTeam"
        },
        "Parents": []
    }
]
```

# Amazon Verified Permissions 정적 정책 생성

보안 주체가 애플리케이션의 지정된 리소스에서 지정된 작업을 수행하도록 허용하거나 금지하는 정적 정책을 만들 수 있습니다.

IAM 정책 생성 62

#### **AWS Management Console**

#### 정적 정책을 생성하려면

1. 에서 검증된 권한 콘솔을 여십시오. <a href="https://console.aws.amazon.com/verifiedpermissions/">https://console.aws.amazon.com/verifiedpermissions/</a> 정 책 저장소를 선택합니다.

- 2. 왼쪽의 탐색 창에서 정책을 선택합니다.
- 3. 정책 생성과 정적 정책 생성을 차례로 선택합니다.
- 4. 정책 효과 섹션에서 요청이 정책과 일치하는 경우 정책을 허용하거나 금지하도록 선택합니다.
- 5. 보안 주체 범위 필드에서 정책을 적용할 보안 주체의 범위를 선택합니다.
  - 정책을 특정 보안 주체에 적용하려면 특정 보안 주체를 선택합니다. 정책에 명시된 작업을 수행하는 것이 허용되거나 금지될 보안 주체의 엔터티 유형과 식별자를 지정합니다.
  - 정책을 보안 주체 그룹에 적용하려면 보안 주체 그룹을 선택합니다. 보안 주체 그룹 필드에 보안 주체 그룹 이름을 입력합니다.
  - 정책 스토어의 모든 보안 주체에 정책을 적용하려면 모든 보안 주체를 선택합니다.
- 6. 리소스 범위 필드에서 정책을 적용할 리소스의 범위를 선택합니다.
  - 정책을 특정 리소스에 적용하려면 특정 리소스를 선택합니다. 정책이 적용될 리소스의 엔터 티 유형과 식별자를 지정합니다.
  - 리소스 그룹에 정책을 적용하려면 리소스 그룹을 선택합니다. 리소스 그룹 필드에 리소스 그룹 룹 이름을 입력합니다.
  - 정책 스토어의 모든 리소스에 정책을 적용하려면 모든 리소스를 선택합니다.
- 7. 작업 범위 섹션에서 정책을 적용할 리소스의 범위를 선택합니다.
  - 정책을 특정 작업 세트에 적용하려면 특정 작업 세트를 선택합니다. 정책을 적용할 작업 옆에 있는 확인란을 선택합니다.
  - 정책 스토어의 모든 작업에 정책을 적용하려면 모든 작업을 선택합니다.
- 8. 다음을 선택합니다.
- 9. 정책 섹션에서 Cedar 정책을 검토합니다. 형식을 선택하여 정책 구문의 형식을 권장 간격과 들여쓰기로 지정할 수 있습니다. 자세한 내용은 Cedar 정책 언어 참조 가이드의 <u>Cedar 기본 정책</u> 구성을 참조하세요.
- 10. 세부 정보 섹션에 정책에 대한 선택적 설명을 입력합니다.
- 11. 정책 생성을 선택합니다.

IAM 정책 생성 63

#### **AWS CLI**

정적 정책을 생성하려면

<u>CreatePolicy</u>작업을 사용하여 정적 정책을 생성할 수 있습니다. 다음 예제에서는 간단한 정적 정책을 생성합니다.

```
$ aws verifiedpermissions create-policy \
     --definition "{ \"static\": { \"Description\": \"MyTestPolicy\", \"Statement\": \"permit(principal,action,resource) when {principal.owner == resource.owner};\"}}"
     --policy-store-id PSEXAMPLEabcdefg11111
{
     "Arn": "arn:aws:verifiedpermissions::123456789012:policy/PSEXAMPLEabcdefg111111/
SPEXAMPLEabcdefg11111",
     "createdDate": "2023-05-16T20:33:01.730817+00:00",
     "lastUpdatedDate": "2023-05-16T20:33:01.730817+00:00",
     "policyId": "SPEXAMPLEabcdefg111111",
     "policyStoreId": "PSEXAMPLEabcdefg111111",
     "policyType": "STATIC"
}
```

# Amazon Verified Permissions 정적 정책 편집

정책 저장소에서 기존 정적 정책을 편집할 수 있습니다. 정적 정책은 직접 업데이트만 할 수 있습니다. 템플릿 연결 정책을 변경하려면 정책 템플릿을 업데이트해야 합니다. 자세한 내용은 <u>Amazon 검증 권</u>한 정책 템플릿 편집 단원을 참조하십시오.

정적 정책의 다음 요소를 변경할 수 있습니다.

- 정책에서 참조하는 action.
- 조건 조항(예: when, unless).

정적 정책의 다음 요소는 변경할 수 없습니다.

- 정책을 정적 정책에서 템플릿 연결 정책으로 변경.
- 정적 정책의 효과를 변경(permit 또는 forbid).
- 정적 정책에서 참조하는 principal.

정적 정책 편집 64

• 정적 정책에서 참조하는 resource.

#### **AWS Management Console**

#### 정적 정책을 편집하려면

1. 에서 검증된 권한 콘솔을 <a href="https://console.aws.amazon.com/verifiedpermissions/">https://console.aws.amazon.com/verifiedpermissions/</a>여십시오. 정책 저장소를 선택합니다.

- 2. 왼쪽의 탐색 창에서 정책을 선택합니다.
- 3. 편집할 정적 정책 옆의 라디오 버튼을 선택한 다음 편집을 선택합니다.
- 4. 정책 본문 섹션에서 정적 정책의 action 또는 조건 조항을 업데이트합니다. 정책 효과, 정책의 principal 또는 resource는 업데이트할 수 없습니다.
- 5. 정책 업데이트를 선택합니다.

#### Note

정책 스토어에서 <u>정책 검증</u>을 사용하도록 설정한 경우 정적 정책을 업데이트하면 Verified Permissions가 정책 스토어의 스키마를 기준으로 정책을 검증합니다. 업데이트된 정적 정책이 검증을 통과하지 못하면 작업이 실패하고 업데이트가 저장되지 않습니다.

#### **AWS CLI**

정적 정책을 편집하려면

<u>UpdatePolicy</u>작업을 사용하여 정적 정책을 편집할 수 있습니다. 다음 예제에서는 간단한 정적 정책을 편집합니다.

이 예제에서는 definition.txt 파일을 사용하여 정책 정의를 포함합니다.

```
{
    "static": {
        "description": "Grant everyone of janeFriends UserGroup access to the
    vacationFolder Album",
        "statement": "permit(principal in UserGroup::\"janeFriends\", action,
    resource in Album::\"vacationFolder\" );"
    }
}
```

정적 정책 편집 65 65

다음 명령은 해당 파일을 참조합니다.

```
$ aws verifiedpermissions create-policy \
    --definition file://definition.txt \
    --policy-store-id PSEXAMPLEabcdefq111111
{
    "createdDate": "2023-06-12T20:33:37.382907+00:00",
    "lastUpdatedDate": "2023-06-12T20:33:37.382907+00:00",
    "policyId": "SPEXAMPLEabcdefg111111",
    "policyStoreId": "PSEXAMPLEabcdefg111111",
    "policyType": "STATIC",
    "principal": {
        "entityId": "janeFriends",
        "entityType": "UserGroup"
    },
    "resource": {
        "entityId": "vacationFolder",
        "entityType": "Album"
}
```

## Amazon 검증 권한 테스트 벤치 사용

검증된 권한 테스트 벤치를 사용하여 검증된 권한 정책에 대해 권한 <u>부여 요청을</u> 실행하여 검증된 권한 정책을 테스트하고 문제를 해결할 수 있습니다. 테스트 벤치는 지정한 매개 변수를 사용하여 정책 스토어의 Cedar 정책이 요청을 승인할지 여부를 결정합니다. 권한 부여 요청을 테스트하는 동안 비주얼모드와 JSON모드 사이를 전환할 수 있습니다. Cedar 정책의 구조 및 평가 방법에 대한 자세한 내용은 Cedar 정책 언어 참조 가이드의 <u>Cedar의 기본 정책 구성</u>을 참조하세요.

## Note

Verified Permissions를 사용하여 권한 부여를 요청하는 경우 추가 엔터티 섹션에 요청의 일부로 보안 주체 및 리소스 목록을 제공할 수 있습니다. 하지만 작업에 대한 세부 정보는 포함할 수 없습니다. 해당 정보는 스키마에 지정하거나 요청으로부터 유추해야 합니다. 추가 엔터티 섹션에는 작업을 추가할 수 없습니다.

테스트 벤치의 시각적 개요 및 데모를 보려면 다음 사이트의 <u>Amazon 검증 권한 - 정책 생성 및 테스트</u> (프라이머 시리즈 #3) 를 참조하십시오. AWS YouTube 채널.

테스트 정책 66

사용자 가이드 Amazon Verified Permissions

#### Visual mode



#### Note

테스트 벤치의 비주얼 모드를 사용하려면 정책 스토어에 스키마가 정의되어 있어야 합니 다.

#### 비주얼 모드에서 정책을 테스트하려면

- 1. 에서 검증된 권한 콘솔을 엽니다 https://console.aws.amazon.com/verifiedpermissions/. 정책 저장소를 선택합니다.
- 2. 왼쪽에 있는 탐색 창에서 테스트 벤치를 선택합니다.
- 비주얼 모드를 선택합니다. 3.
- 4. 보안 주체 섹션에서 스키마의 보안 주체 유형 중에서 작업을 수행하는 보안 주체를 선택합니 다. 텍스트 상자에 보안 주체 식별자를 입력합니다.
- 5. (선택 사항) 상위 추가를 선택하여 지정된 보안 주체에 대해 상위 엔터티를 추가합니다. 보안 주체에 추가된 상위 엔터티를 제거하려면 상위 엔터티 이름 옆에 있는 제거를 선택합니다.
- 지정된 보안 주체의 각 속성에 대해 속성 값을 지정합니다. 테스트 벤치는 시뮬레이션된 권한 6. 부여 요청에서 지정된 속성 값을 사용합니다.
- 리소스 섹션에서 보안 주체가 작업을 수행하는 리소스를 선택합니다. 텍스트 상자에 리소스 식 별자를 입력합니다.
- 8. (선택 사항) 상위 추가를 선택하여 지정된 리소스에 대해 상위 엔터티를 추가합니다. 리소스에 추가된 상위 엔터티를 제거하려면 상위 엔터티 이름 옆에 있는 제거를 선택합니다.
- 9. 지정된 리소스의 각 속성에 대해 속성 값을 지정합니다. 테스트 벤치는 시뮬레이션된 권한 부 여 요청에서 지정된 속성 값을 사용합니다.
- 10. 작업 섹션에서 지정된 보안 주체 및 리소스에 대한 유효한 작업 목록에서 보안 주체가 수행하 는 작업을 선택합니다.
- 11. 지정된 작업의 각 속성에 대해 속성 값을 지정합니다. 테스트 벤치는 시뮬레이션된 권한 부여 요청에서 지정된 속성 값을 사용합니다.
- 12. (선택 사항) 추가 엔터티 섹션에서 엔터티 추가를 선택하여 권한 부여 결정을 위해 평가할 엔터 티를 추가합니다.
- 13. 드롭다운 목록에서 엔터티 식별자를 선택하고 엔터티 식별자를 입력합니다.
- 14. (선택 사항) 상위 추가를 선택하여 지정된 엔터티에 대해 상위 엔터티를 추가합니다. 엔터티에 추가된 상위 엔터티를 제거하려면 상위 엔터티 이름 옆에 있는 제거를 선택합니다.

테스트 정책 67

15. 지정된 엔터티의 각 속성에 대해 속성 값을 지정합니다. 테스트 벤치는 시뮬레이션된 권한 부여 요청에서 지정된 속성 값을 사용합니다.

- 16. 확인을 선택하여 테스트 벤치에 엔터티를 추가합니다.
- 17. 권한 부여 요청 실행을 선택하여 정책 스토어의 Cedar 정책에 대한 권한 부여 요청을 시뮬레이션합니다. 테스트 벤치에는 평가 중에 충족된 정책이나 발생한 오류에 대한 정보와 함께 요청 허용 또는 거부 결정이 표시됩니다.

#### JSON mode

#### JSON모드에서 정책을 테스트하려면

- 1. 에서 검증된 권한 콘솔을 <a href="https://console.aws.amazon.com/verifiedpermissions/">https://console.aws.amazon.com/verifiedpermissions/</a>여십시오. 정책 저장소를 선택합니다.
- 2. 왼쪽에 있는 탐색 창에서 테스트 벤치를 선택합니다.
- 3. JSON모드를 선택하세요.
- 요청 세부 정보 섹션에서 스키마가 정의되어 있는 경우 스키마의 보안 주체 유형 중에서 작업을 수행하는 보안 주체를 선택합니다. 텍스트 상자에 보안 주체 식별자를 입력합니다.
  - 스키마가 정의되지 않은 경우 작업을 수행하는 보안 주체 텍스트 상자에 보안 주체를 입력합니다.
- 5. 스키마가 정의되어 있는 경우 스키마의 리소스 유형에서 리소스를 선택합니다. 텍스트 상자에 리소스 식별자를 입력합니다.
  - 스키마가 정의되지 않은 경우 리소스 텍스트 상자에 리소스를 입력합니다.
- 6. 스키마가 정의되어 있는 경우 지정된 보안 주체 및 리소스에 대한 유효한 작업 목록에서 작업을 선택합니다.
  - 스키마가 정의되지 않은 경우 작업 텍스트 상자에 작업을 입력합니다.
- 7. 컨텍스트 필드에 시뮬레이션 요청의 컨텍스트를 입력합니다. 요청 컨텍스트는 권한 부여 결정에 사용할 수 있는 추가 정보입니다.
- 8. 엔터티 필드에 엔터티의 계층 구조 및 권한 부여 결정을 위해 평가할 속성을 입력합니다.
- 9. 권한 부여 요청 실행을 선택하여 정책 스토어의 Cedar 정책에 대한 권한 부여 요청을 시뮬레이 션합니다. 테스트 벤치에는 평가 중에 충족된 정책이나 발생한 오류에 대한 정보와 함께 요청 허용 또는 거부 결정이 표시됩니다.

테스트 정책 68

## Amazon Verified Permissions 정책 예제

다음 검증된 권한 정책 예제는 Cedar 정책 언어 참조 가이드의 예제 <u>스키마 섹션에 PhotoFlash 설명된 가상 애플리케이션에 대해 정의된 스키마를</u> 기반으로 합니다. Cedar 정책 구문에 대한 자세한 내용은 Cedar 정책 언어 참조 가이드의 Cedar 기본 정책 구성을 참조하세요.

#### 정책 예제

- 개별 엔터티에 대해 액세스 허용하기
- 엔터티 그룹에 대해 액세스 허용하기
- 모든 엔터티에 대해 액세스 허용하기
- 엔티티의 속성에 대한 액세스를 허용합니다 (ABAC).
- 액세스 거부
- 대괄호 표기법을 사용하여 토큰 속성을 참조합니다.
- 점 표기법을 사용하여 속성을 참조합니다.
- 아마존 Cognito ID 토큰 속성을 반영합니다.
- OIDCID 토큰 속성 반영
- Amazon Cognito 액세스 토큰 속성을 반영합니다.
- OIDC액세스 토큰 속성을 반영합니다.

## 개별 엔터티에 대해 액세스 허용하기

다음 예는 사용자가 alice 사진을 볼 수 있도록 허용하는 정책을 만드는 방법을 보여줍니다. VacationPhoto94.jpg

```
permit(
  principal == User::"alice",
  action == Action::"view",
  resource == Photo::"VacationPhoto94.jpg"
);
```

## 엔터티 그룹에 대해 액세스 허용하기

다음 예는 그룹의 alice\_friends 모든 사용자가 사진을 볼 수 있도록 허용하는 정책을 만드는 방법을 보여줍니다VacationPhoto94.jpg.

정책 예제 69

```
permit(
  principal in Group::"alice_friends",
  action == Action::"view",
  resource == Photo::"VacationPhoto94.jpg"
);
```

다음 예는 사용자가 alice 앨범에 있는 모든 사진을 볼 수 있도록 허용하는 정책을 만드는 방법을 보여줍니다alice vacation.

```
permit(
  principal == User::"alice",
  action == Action::"view",
  resource in Album::"alice_vacation"
);
```

다음 예는 사용자가 alice 앨범에 있는 사진을 보거나, 편집하거나, 삭제할 수 있도록 허용하는 정책을 만드는 방법을 보여줍니다alice\_vacation.

```
permit(
  principal == User::"alice",
  action in [Action::"view", Action::"edit", Action::"delete"],
  resource in Album::"alice_vacation"
);
```

다음 예제는 alice alice\_vacation 앨범에서 사용자에게 권한을 허용하는 정책을 만드는 방법을 보여줍니다. 여기서 admin 그룹은 스키마 계층 구조에 정의되어 있으며 사진을 보고, 편집하고, 삭제 할 수 있는 권한이 포함되어 있습니다.

```
permit(
  principal == User::"alice",
  action in PhotoflashRole::"admin",
  resource in Album::"alice_vacation"
);
```

다음 예제는 alice 앨범에서 사용자에게 권한을 허용하는 정책을 만드는 방법을 보여줍니다. alice\_vacation 앨범에서는 사진을 보고 댓글을 달 수 있는 viewer 권한을 포함하는 스키마 계층 구조에 정의된 그룹입니다. 정책에 나열된 두 번째 작업을 통해서도 사용자 edit에게 alice 권한이 부여됩니다.

```
permit(
```

```
principal == User::"alice",
  action in [PhotoflashRole::"viewer", Action::"edit"],
  resource in Album::"alice_vacation"
)
```

## 모든 엔터티에 대해 액세스 허용하기

다음 예는 인증된 모든 보안 주체가 alice\_vacation 앨범을 볼 수 있도록 허용하는 정책을 만드는 방법을 보여줍니다.

```
permit(
  principal,
  action == Action::"view",
  resource in Album::"alice_vacation"
);
```

다음 예는 사용자가 계정의 모든 앨범을 나열하고, 각 앨범의 사진을 alice 나열하고, jane 계정의 사진을 볼 수 있도록 허용하는 정책을 만드는 방법을 보여줍니다.

```
permit(
  principal == User::"alice",
  action in [Action::"listAlbums", Action::"listPhotos", Action::"view"],
  resource in Account::"jane"
);
```

다음 예제는 사용자가 alice jane\_vaction 앨범의 리소스에 대해 모든 작업을 수행할 수 있도록 허용하는 정책을 만드는 방법을 보여줍니다.

```
permit(
  principal == User::"alice",
  action,
  resource in Album::"jane_vacation"
);
```

## 엔티티의 속성에 대한 액세스를 허용합니다 (ABAC).

속성 기반 액세스 제어 (ABAC) 는 속성을 기반으로 권한을 정의하는 권한 부여 전략입니다. Verified Permissions를 사용하면 속성을 보안 주체, 작업 및 리소스에 추가할 수 있습니다. 그러면 요청의 컨텍스트를 구성하는 보안 주체, 작업 및 리소스의 속성을 평가하는 정책의 when 및 unless 조항 내에서 이러한 속성을 참조할 수 있습니다.

다음 예에서는 Cedar 정책 언어 참조 가이드의 <u>예제 스키마</u> 섹션에 PhotoFlash 설명된 가상 응용 프로 그램에 정의된 속성을 사용합니다.

다음 예는 직무 등급이 5등급 이상인  $Hardware Engineering 부서의 모든 교장이 앨범에 있는 사진을 보고 나열할 수 있도록 하는 정책을 만드는 방법을 보여줍니다. <math>device\_prototypes$ 

```
permit(
  principal,
  action in [Action::"listPhotos", Action::"view"],
  resource in Album::"device_prototypes"
)
when {
  principal.department == "HardwareEngineering" &&
  principal.jobLevel >= 5
};
```

다음 예는 사용자가 alice 파일 유형의 모든 리소스를 볼 수 있도록 허용하는 정책을 만드는 방법을 보여줍니다JPEG.

```
permit(
  principal == User::"alice",
  action == Action::"view",
  resource
)
when {
  resource.fileType == "JPEG"
};
```

작업에는 컨텍스트 속성이 있습니다. 권한 부여 요청 시 이러한 속성을 전달해야 합니다. context 다음 예제는 사용자가 alice 모든 readOnly 작업을 수행할 수 있도록 허용하는 정책을 만드는 방법을 보여줍니다. 스키마의 작업 appliesTo 속성을 설정할 수도 있습니다. 이는 예를 들어 사용자가 특정유형의 PhotoFlash::Photo 리소스에 대해서만 권한 부여를 시도할 수 있도록 하려는 경우 해당 리소스에 ViewPhoto 대한 유효한 작업을 지정합니다.

```
permit(
  principal == PhotoFlash::User::"alice",
  action,
  resource
) when {
    context has readOnly &&
    context.readOnly == true
```

```
};
```

그러나 스키마에서 작업의 속성을 설정하는 더 좋은 방법은 작업을 기능적 작업 그룹으로 정렬하는 것입니다. 예를 들어 이름이 ReadOnlyPhotoAccess 지정되고 작업 그룹의 ReadOnlyPhotoAccess 구성원이 PhotoFlash::Action::"ViewPhoto" 되도록 설정된 작업을 만들 수 있습니다. 다음 예제는 Alice에게 해당 그룹의 읽기 전용 작업에 대한 액세스 권한을 부여하는 정책을 만드는 방법을 보여줍니다.

```
permit(
  principal == PhotoFlash::User::"alice",
  action,
  resource
) when {
    action in PhotoFlash::Action::"ReadOnlyPhotoAccess"
};
```

다음 예제는 모든 보안 주체가 자신에게 속성이 있는 리소스에 대해 모든 작업을 수행할 수 있도록 하는 정책을 만드는 방법을 보여줍니다. owner

```
permit(
  principal,
  action,
  resource
)
when {
  principal == resource.owner
};
```

다음 예는 보안 주체의 department 속성이 리소스의 속성과 일치하는 경우 모든 보안 주체가 모든 리소스를 볼 수 있도록 허용하는 정책을 만드는 방법을 보여줍니다. department

## Note

엔터티에 정책 조건에 언급된 속성이 없는 경우 권한 부여 결정을 내릴 때 해당 정책이 무시되고 해당 엔터티에 대한 해당 정책의 평가가 실패합니다. 예를 들어 department 속성이 없는 보안 주체는 이 정책에 따라 리소스에 대한 액세스 권한을 부여받을 수 없습니다.

```
permit(
```

```
principal,
  action == Action::"view",
  resource
)
when {
  principal.department == resource.owner.department
};
```

다음 예제는 보안 주체가 리소스에 속해 있거나 보안 주체가 리소스 admins 그룹에 속해 있는 경우 모든 보안 주체가 리소스에 대해 모든 작업을 수행하도록 허용하는 정책을 만드는 방법을 보여줍니다. owner

```
permit(
  principal,
  action,
  resource,
)
when {
  principal == resource.owner ||
  resource.admins.contains(principal)
};
```

## 액세스 거부

정책의 정책 효과에 forbid가 포함되는 경우 권한을 부여하는 대신 권한을 제한합니다.

## ▲ Important

권한을 부여하는 중에 permit 및 forbid 정책이 모두 적용되는 경우 forbid가 우선합니다.

다음 예에서는 Cedar 정책 언어 참조 가이드의 <u>예제 스키마</u> 섹션에 PhotoFlash 설명된 가상 응용 프로 그램에 정의된 속성을 사용합니다.

다음 예제는 사용자가 리소스를 제외한 readOnly 모든 작업을 수행하지 alice 못하도록 하는 정책을 만드는 방법을 보여줍니다.

```
forbid (
 principal == User::"alice",
 action,
 resource
```

액세스 거부 74

```
)
unless {
  action.readOnly
};
```

다음 예제는 보안 주체에 해당 리소스에 대한 private 속성이 없는 한 속성이 있는 모든 리소스에 대한 액세스를 거부하는 정책을 만드는 방법을 보여줍니다. owner

```
forbid (
  principal,
  action,
  resource
)
when {
  resource.private
}
unless {
  principal == resource.owner
};
```

## 대괄호 표기법을 사용하여 토큰 속성을 참조합니다.

다음 예제는 대괄호 표기법을 사용하여 토큰 속성을 참조하는 정책을 생성하는 방법을 보여줍니다.

검증된 권한의 정책에서 토큰 속성을 사용하는 방법에 대한 자세한 내용은 을 참조하십시오. <u>ID 제공자</u> <u>토큰을 스키마에 매핑</u>

```
permit (
    principal in MyCorp::UserGroup::"us-west-2_EXAMPLE|MyUserGroup",
    action,
    resource
) when {
    principal["cognito:username"] == "alice" &&
    principal["custom:employmentStoreCode"] == "petstore-dallas" &&
    principal has email && principal.email == "alice@example.com" &&
    context["ip-address"] like "192.0.2.*"
};
```

## 점 표기법을 사용하여 속성을 참조합니다.

다음 예제는 점 표기법을 사용하여 속성을 참조하는 정책을 생성하는 방법을 보여줍니다.

검증된 권한의 정책에서 토큰 속성을 사용하는 방법에 대한 자세한 내용은 다음을 참조하십시오. <u>ID 제</u> 공자 토큰을 스키마에 매핑

```
permit(principal, action, resource)
when {
    principal.cognito.username == "alice" &&
    principal.custom.employmentStoreCode == "petstore-dallas" &&
    principal.tenant == "x11app-tenant-1" &&
    principal has email && principal.email == "alice@example.com"
};
```

## 아마존 Cognito ID 토큰 속성을 반영합니다.

다음 예제는 Amazon Cognito에서 정책 참조 ID 토큰 속성을 생성하는 방법을 보여줍니다.

검증된 권한의 정책에서 토큰 속성을 사용하는 방법에 대한 자세한 내용은 다음을 참조하십시오. <u>ID 제</u>공자 토큰을 스키마에 매핑

```
permit (
    principal in MyCorp::UserGroup::"us-west-2_EXAMPLE|MyUserGroup",
    action,
    resource
) when {
    principal["cognito:username"] == "alice" &&
    principal["custom:employmentStoreCode"] == "petstore-dallas" &&
    principal.tenant == "x11app-tenant-1" &&
    principal has email && principal.email == "alice@example.com"
};
```

## OIDCID 토큰 속성 반영

다음 예는 OIDC 공급자의 ID 토큰 속성을 참조하는 정책을 만드는 방법을 보여줍니다.

검증된 권한의 정책에서 토큰 속성을 사용하는 방법에 대한 자세한 내용은 다음을 참조하십시오. <u>ID 제</u> 공자 토큰을 스키마에 매핑

```
permit (
    principal in MyCorp::UserGroup::"MyOIDCProvider|MyUserGroup",
    action,
    resource
) when {
```

```
principal.email_verified == true && principal.email == "alice@example.com" &&
    principal.phone_number_verified == true && principal.phone_number like "+1206*"
};
```

## Amazon Cognito 액세스 토큰 속성을 반영합니다.

다음 예제는 Amazon Cognito의 정책 참조 액세스 토큰 속성을 생성하는 방법을 보여줍니다.

검증된 권한의 정책에서 토큰 속성을 사용하는 방법에 대한 자세한 내용은 다음을 참조하십시오. <u>ID 제</u> 공자 토큰을 스키마에 매핑

```
permit(principal, action in [MyApplication::Action::"Read",
   MyApplication::Action::"GetStoreInventory"], resource)
when {
    context.token.client_id == "52n97d5afhfiu1c4di1k5m8f60" &&
    context.token.scope.contains("MyAPI/mydata.write")
};
```

## OIDC액세스 토큰 속성을 반영합니다.

다음 예제는 OIDC 제공자의 액세스 토큰 속성을 참조하는 정책을 생성하는 방법을 보여줍니다.

검증된 권한의 정책에서 토큰 속성을 사용하는 방법에 대한 자세한 내용은 다음을 참조하십시오. <u>ID 제</u> <u>공자 토큰을 스키마에 매핑</u>

```
permit(
    principal,
    action in [MyApplication::Action::"Read",
    MyApplication::Action::"GetStoreInventory"],
    resource
)
when {
    context.token.client_id == "52n97d5afhfiu1c4di1k5m8f60" &&
    context.token.scope.contains("MyAPI-read")
};
```

## Amazon 검증 권한 정책 템플릿 및 템플릿 연결 정책

검증된 권한에서 정책 템플릿은 principalresource, 또는 둘 다에 대한 자리 표시자가 있는 정책입니다. 정책 템플릿만으로는 권한 부여 요청을 처리하는 데 사용할 수 없습니다. 권한 부여 요청을 처리하려면 정책 템플릿을 기반으로 템플릿 연결 정책을 생성해야 합니다. 정책 템플릿을 사용하면 정책을 한 번 정의한 다음 여러 주체 및 리소스와 함께 사용할 수 있습니다. 정책 템플릿에 대한 업데이트는 해당 템플릿을 사용하는 모든 정책에 반영됩니다. 자세한 내용은 Cedar 정책 언어 참조 가이드의 Cedar 정책 템플릿을 참조하세요.

예를 들어, 다음 정책 템플릿은 정책 템플릿을 사용하는 보안 주체 및 리소스에Edit, 및 Comment 권한을 제공합니다Read.

```
permit(
  principal == ?principal,
  action in [Action::"Read", Action::"Edit", Action::"Comment"],
  resource == ?resource
);
```

이 템플릿을 Editor 기반으로 이름이 지정된 정책을 만들려는 경우 주도자가 특정 리소스의 편집자로 지정되면 응용 프로그램은 주도자에게 해당 리소스에 대한 읽기, 편집 및 의견 수렴 권한을 제공하는 정책을 생성합니다.

정적 정책과 달리 템플릿 연결 정책은 동적입니다. 위의 예를 들어, 정책 템플릿에서 Comment 작업을 제거하면 해당 템플릿에 연결되거나 템플릿을 기반으로 하는 모든 정책이 그에 따라 업데이트되고 정책에 지정된 보안 주체는 더 이상 해당 리소스에 대해 의견을 제시할 수 없게 됩니다.

템플릿 연결 정책 예제에 대한 자세한 내용은 을 참조하십시오. <u>Amazon 검증 권한 예제 템플릿 연결</u> <u>정책</u>

## Amazon 검증 권한 정책 템플릿 생성

검증된 권한에서 AWS Management Console AWS CLI, 또는 를 사용하여 정책 템플릿을 생성할 수 AWS SDKs 있습니다. 정책 템플릿을 사용하면 정책을 한 번 정의한 다음 여러 주체 및 리소스와 함께 사용할 수 있습니다. 정책 템플릿을 만든 후에는 템플릿으로 연결된 정책을 생성하여 특정 주체 및 리소스가 포함된 정책 템플릿을 사용할 수 있습니다. 자세한 내용은 Amazon 검증 권한 템플릿 연결 정책 생성 단원을 참조하십시오.

정책 템플릿 생성 78

#### **AWS Management Console**

#### 정책 템플릿을 생성하려면

1. 에서 검증된 권한 콘솔을 엽니다. <a href="https://console.aws.amazon.com/verifiedpermissions/">https://console.aws.amazon.com/verifiedpermissions/</a> 정책 저장소를 선택합니다.

- 2. 왼쪽의 탐색 창에서 정책 템플릿을 선택합니다.
- 3. 정책 템플릿 생성을 선택합니다.
- 4. 세부 정보 섹션에서 정책 템플릿 설명을 입력합니다.
- 5. 정책 템플릿 본문 섹션에서 ?principal 및 ?resource 자리 표시자를 사용하여 이 템플릿을 기반으로 만든 정책이 부여하는 권한을 사용자 지정할 수 있도록 허용합니다. 형식을 선택하여 정책 템플릿 구문의 형식을 권장 간격과 들여쓰기로 지정할 수 있습니다.
- 6. 정책 템플릿 생성을 선택합니다.

#### **AWS CLI**

정책 템플릿을 생성하려면

<u>CreatePolicyTemplate</u>작업을 사용하여 정책 템플릿을 생성할 수 있습니다. 다음 예에서는 보안 주체에 대한 자리 표시자가 있는 정책 템플릿을 생성합니다.

template1.txt 파일에는 다음 코드가 포함되어 있습니다.

```
"VacationAccess"
permit(
    principal in ?principal,
    action == Action::"view",
    resource == Photo::"VacationPhoto94.jpg"
);
```

```
$ aws verifiedpermissions create-policy-template \
    --description "Template for vacation picture access"
    --statement file://template1.txt
    --policy-store-id PSEXAMPLEabcdefg11111
{
    "createdDate": "2023-05-18T21:17:47.284268+00:00",
    "lastUpdatedDate": "2023-05-18T21:17:47.284268+00:00",
    "policyStoreId": "PSEXAMPLEabcdefg111111",
```

정책 템플릿 생성 79

```
"policyTemplateId": "PTEXAMPLEabcdefg111111"
```

## Amazon 검증 권한 템플릿 연결 정책 생성

, 또는 를 사용하여 템플릿 연결 정책 또는 정책 템플릿을 기반으로 하는 정책을 만들 수 있습니다. AWS Management Console AWS CLI AWS SDKs 템플릿 연결 정책은 해당 정책 템플릿에 연결된 상 태로 유지됩니다. 정책 템플릿에서 정책 설명을 변경하면 해당 템플릿에 연결된 모든 정책이 해당 시점 부터 내려진 모든 권한 부여 결정에 새 설명을 자동으로 사용합니다.

템플릿 연결 정책 예제는 을 참조하십시오. Amazon 검증 권한 예제 템플릿 연결 정책

#### **AWS Management Console**

}

정책 템플릿을 인스턴스화하여 템플릿 연결 정책을 생성하려면

- 1. 에서 검증된 권한 콘솔을 여십시오. <a href="https://console.aws.amazon.com/verifiedpermissions/">https://console.aws.amazon.com/verifiedpermissions/</a> 정 책 저장소를 선택합니다.
- 2. 왼쪽 탐색 창에서 정책을 선택합니다.
- 3. 정책 생성을 선택한 다음 템플릿 연결 정책 생성을 선택합니다.
- 4. 사용할 정책 템플릿 옆의 라디오 버튼을 선택한 후 다음을 선택합니다.
- 5. 템플릿 연결 정책의 이 특정 인스턴스에 사용할 보안 주체와 리소스를 입력합니다. 지정된 값은 정책 설명 미리 보기 필드에 표시됩니다.

#### Note

보안 주체 및 리소스 값은 정적 정책과 형식이 같아야 합니다. 예를 들어, 보안 주체에 대한 AdminUsers 그룹을 지정하려면 Group::"AdminUsers"를 입력합니다. AdminUsers를 입력하면 유효성 검사 오류가 표시됩니다.

6. 템플릿 연결 정책 생성을 선택합니다.

새 템플릿 연결 정책이 정책 아래에 표시됩니다.

#### **AWS CLI**

정책 템플릿을 인스턴스화하여 템플릿 연결 정책을 생성하려면

템플릿 연결 정책 생성 80

기존 정책 템플릿을 참조하고 템플릿에서 사용하는 자리 표시자의 값을 지정하는 템플릿 연결 정책을 만들 수 있습니다.

다음 예에서는 다음 문을 사용하여 템플릿을 사용하는 템플릿 연결 정책을 생성합니다.

```
permit(
    principal in ?principal,
    action == PhotoFlash::Action::"view",
    resource == PhotoFlash::Photo::"VacationPhoto94.jpg"
);
```

또한 다음 definition.txt 파일을 사용하여 definition 매개 변수 값을 제공합니다.

```
{
    "templateLinked": {
        "policyTemplateId": "PTEXAMPLEabcdefg111111",
        "principal": {
            "entityType": "PhotoFlash::User",
            "entityId": "alice"
        }
    }
}
```

출력에는 템플릿에서 가져오는 리소스와 정의 매개변수에서 가져오는 보안 주체 리소스가 모두 표시됩니다.

```
$ aws verifiedpermissions create-policy \
    --definition file://definition.txt
    --policy-store-id PSEXAMPLEabcdefg111111
{
    "createdDate": "2023-05-22T18:57:53.298278+00:00",
    "lastUpdatedDate": "2023-05-22T18:57:53.298278+00:00",
    "policyId": "TPEXAMPLEabcdefg111111",
    "policyStoreId": "PSEXAMPLEabcdefg111111",
    "policyType": "TEMPLATELINKED",
    "principal": {
        "entityId": "alice",
        "entityType": "PhotoFlash::User"
    },
    "resource": {
        "entityId": "VacationPhoto94.jpg",
        "entityType": "PhotoFlash::Photo"
```

템플릿 연결 정책 생성 81

```
}
```

# Amazon 검증 권한 정책 템플릿 편집

, 또는 를 사용하여 검증된 권한에서 정책 템플릿을 편집하거나 업데이트할 수 AWS SDKs 있습니다. AWS Management Console AWS CLI정책 템플릿을 편집하면 템플릿에 연결되거나 템플릿에 기반한 정책이 자동으로 업데이트되므로 정책 템플릿을 편집할 때 주의하여 실수로 응용 프로그램을 중단시키는 변경 사항이 발생하지 않도록 하십시오.

#### **AWS Management Console**

정책 템플릿을 편집하려면

- 1. 에서 검증된 권한 콘솔을 <a href="https://console.aws.amazon.com/verifiedpermissions/">https://console.aws.amazon.com/verifiedpermissions/</a>여십시오. 정책 저장소를 선택합니다.
- 2. 왼쪽의 탐색 창에서 정책 템플릿을 선택합니다. 콘솔에는 현재 정책 스토어에서 생성한 모든 정책 템플릿이 표시됩니다.
- 3. 정책 템플릿 옆에 있는 라디오 버튼을 선택하면 정책 템플릿 생성, 업데이트 시기, 정책 템플릿 내용 등 정책 템플릿에 대한 세부 정보가 표시됩니다.
- 4. 편집을 선택하여 정책 템플릿을 편집합니다. 필요에 따라 정책 설명 및 정책 본문을 업데이트 한 다음 정책 템플릿 업데이트를 선택합니다.
- 정책 템플릿 옆에 있는 라디오 버튼을 선택한 다음 삭제를 선택하면 정책 템플릿을 삭제할 수 있습니다. 확인을 선택하여 정책 템플릿 삭제를 확인합니다.

#### **AWS CLI**

정책 템플릿을 업데이트하려면

<u>UpdatePolicy</u>작업을 사용하여 정적 정책을 생성할 수 있습니다. 다음 예에서는 정책 본문을 파일에 정의된 새 정책으로 교체하여 지정된 정책 템플릿을 업데이트합니다.

template1.txt 파일의 내용:

```
permit(
    principal in ?principal,
    action == Action::"view",
```

정책 템플릿 편집 82

```
resource in ?resource)
when {
   principal has department && principal.department == "research"
};
```

```
$ aws verifiedpermissions update-policy-template \
    --policy-template-id PTEXAMPLEabcdefg111111 \
    --description "My updated template description" \
    --statement file://template1.txt \
    --policy-store-id PSEXAMPLEabcdefg111111

{
    "createdDate": "2023-05-17T18:58:48.795411+00:00",
    "lastUpdatedDate": "2023-05-17T19:18:48.870209+00:00",
    "policyStoreId": "PSEXAMPLEabcdefg111111",
    "policyTemplateId": "PTEXAMPLEabcdefg111111"
}
```

## Amazon 검증 권한 예제 템플릿 연결 정책

샘플 정책 스토어 방법을 사용하여 Verified Permissions에서 정책 스토어를 생성하면 선택한 샘플 프로젝트에 대한 사전 정의된 정책, 정책 템플릿 및 스키마를 사용하여 정책 스토어가 생성됩니다. 다음과 같은 Verified Permissions 템플릿 연결 정책 예제를 샘플 정책 스토어 및 해당 정책, 정책 템플릿 및 스키마와 함께 사용할 수 있습니다.

## PhotoFlash예제

다음 예에서는 개인 사용자와 비공개로 공유한 사진 및 사진에 제한된 액세스 권한 부여 정책 템플릿을 사용하는 템플릿 연결 정책을 만드는 방법을 보여줍니다.

Note

Cedar 정책 언어에서는 엔터티를 in 자체로 간주합니다. 따라서 principal in User::"Alice"는 principal == User::"Alice"와 같습니다.

```
permit (
  principal in PhotoFlash::User::"Alice",
  action in PhotoFlash::Action::"SharePhotoLimitedAccess",
  resource in PhotoFlash::Photo::"VacationPhoto94.jpg"
```

템플릿 연결 정책의 예 83

```
);
```

다음 예제는 개인 사용자 및 앨범과 비공개로 공유된 사진에 대한 제한된 액세스 허용이라는 정책 템플 릿을 사용하는 템플릿 연결 정책을 만드는 방법을 보여줍니다.

```
permit (
  principal in PhotoFlash::User::"Alice",
  action in PhotoFlash::Action::"SharePhotoLimitedAccess",
  resource in PhotoFlash::Album::"Italy2023"
  );
```

다음 예는 친구 그룹과 비공개로 공유한 사진 및 개별 사진에 대한 제한된 액세스 허용이라는 정책 템 플릿을 사용하는 템플릿 연결 정책을 만드는 방법을 보여줍니다.

```
permit (
  principal in PhotoFlash::FriendGroup::"Jane::MySchoolFriends",
  action in PhotoFlash::Action::"SharePhotoLimitedAccess",
  resource in PhotoFlash::Photo::"VacationPhoto94.jpg"
  );
```

다음 예제는 친구 그룹 및 앨범과 비공개로 공유한 사진에 대한 제한된 액세스 허용이라는 정책 템플릿을 사용하는 템플릿 연결 정책을 만드는 방법을 보여줍니다.

```
permit (
  principal in PhotoFlash::FriendGroup::"Jane::MySchoolFriends",
  action in PhotoFlash::Action::"SharePhotoLimitedAccess",
  resource in PhotoFlash::Album::"Italy2023"
);
```

다음 예는 친구 그룹과 비공개로 공유한 사진 및 개별 사진에 대한 전체 액세스 권한 부여 정책 템플릿을 사용하는 템플릿 연결 정책을 만드는 방법을 보여줍니다.

```
permit (
  principal in PhotoFlash::UserGroup::"Jane::MySchoolFriends",
  action in PhotoFlash::Action::"SharePhotoFullAccess",
  resource in PhotoFlash::Photo::"VacationPhoto94.jpg"
  );
```

다음 예제는 계정에서 사용자 차단 정책 템플릿을 사용하는 템플릿 연결 정책을 만드는 방법을 보여줍니다.

PhotoFlash예제 84

```
forbid(
principal == PhotoFlash::User::"Bob",
action,
resource in PhotoFlash::Account::"Alice-account"
);
```

## DigitalPetStore 예:

DigitalPetStore 샘플 정책 저장소에는 정책 템플릿이 포함되어 있지 않습니다. DigitalPetStore샘플 정책 저장소를 생성한 후 왼쪽의 탐색 창에서 정책을 선택하여 정책 저장소에 포함된 정책을 볼 수 있습니다.

## TinyToDo예:

다음 예제는 개별 사용자 및 작업 목록에 대한 뷰어 액세스 권한을 부여하는 정책 템플릿을 사용하는 템플릿 연결 정책을 만드는 방법을 보여줍니다.

```
permit (
    principal == TinyTodo::User::"https://cognito-idp.us-east-1.amazonaws.com/us-
east-1_h2aKCU1ts|5ae0c4b1-6de8-4dff-b52e-158188686f31|bob",
    action in [TinyTodo::Action::"ReadList", TinyTodo::Action::"ListTasks"],
    resource == TinyTodo::List::"1"
);
```

다음 예제는 정책 템플릿을 사용하여 개별 사용자 및 작업 목록에 대한 편집자 액세스 권한을 부여하는 템플릿 연결 정책을 만드는 방법을 보여줍니다.

```
permit (
    principal == TinyTodo::User::"https://cognito-idp.us-east-1.amazonaws.com/us-
east-1_h2aKCU1ts|5ae0c4b1-6de8-4dff-b52e-158188686f31|bob",
    action in [
        TinyTodo::Action::"ReadList",
        TinyTodo::Action::"UpdateList",
        TinyTodo::Action::"ListTasks",
        TinyTodo::Action::"CreateTask",
        TinyTodo::Action::"UpdateTask",
        TinyTodo::Action::"DeleteTask"
],
    resource == TinyTodo::List::"1"
);
```

DigitalPetStore 예: 85

## 자격 증명 공급자와 함께 Amazon Verified Permissions 사용

자격 증명 소스는 Amazon 검증 권한에 있는 외부 ID 공급자 (IdP) 를 나타냅니다. ID 소스는 정책 저장소와 신뢰 관계에 있는 IdP로 인증한 사용자의 정보를 제공합니다. 애플리케이션이 ID 소스의 토큰을 사용하여 권한 부여를 요청하면 정책 저장소가 사용자 속성 및 액세스 권한을 기반으로 권한 부여를 결정할 수 있습니다. Verified Permissions ID 소스는 중앙 ID 저장소 및 인증 서비스에 직접 연결하여 권한 부여를 개선합니다.

검증된 권한이 있는 OpenID Connect (OIDC) ID 공급자 (IdPs) 를 사용할 수 있습니다. 애플리케이션은 ID (OIDCID) 또는 액세스 JSON 웹 토큰 (JWTs) 을 사용하여 권한 부여 요청을 생성할 수 있습니다. ID 토큰을 사용하는 경우 Verified Permissions는 사용자 IDs 및 속성 클레임을 속성 기반 액세스 제어의 주체로 읽습니다 (). ABAC 액세스 토큰을 사용하는 경우 Verified Permission은 사용자를 보안 IDs 주체로, 다른 클레임을 컨텍스트로 읽습니다. 두 가지 토큰 유형을 모두 사용하여 클레임을 주 그룹처럼 groups 매핑하고 역할 기반 액세스 제어를 평가하는 정책을 구축할 수 있습니다 (). RBAC

Amazon Cognito 사용자 풀 또는 사용자 지정 OpenID Connect () OIDC IdP를 자격 증명 소스로 추가할 수 있습니다.

Amazon Cognito 사용자 풀 OIDC 또는 자격 증명 공급자를 REST APIs 사용하여 Amazon API Gateway에 대한 권한 부여 로직을 구축하는 방법을 step-by-step 살펴보려면 Amazon Cognito의 Amazon 검증 권한을 사용하여 APIs 게이트웨이를 API 승인하거나 자체 자격 증명 공급자를 불러오기를 참조하십시오. AWS 보안 블로그.

#### 주제

- Amazon Cognito 자격 증명 소스를 사용한 작업
- 자격 증명 소스 사용 OIDC
- 클라이언트 및 잠재고객 검증
- <u>클라이언트 측 권한 부여: JWTs</u>
- Amazon Verified Permissions 자격 증명 소스 생성
- Amazon Verified Permissions 자격 증명 소스 편집
- ID 제공자 토큰을 스키마에 매핑

## Amazon Cognito 자격 증명 소스를 사용한 작업

검증된 권한은 Amazon Cognito 사용자 풀과 밀접하게 작동합니다. Amazon Cognito는 예측 가능한 JWTs 구조를 가지고 있습니다. 검증된 권한은 이 구조를 인식하고 해당 구조에 포함된 정보를 최대한

활용합니다. 예를 들어 ID 토큰 또는 액세스 토큰을 사용하여 역할 기반 액세스 제어 (RBAC) 권한 부여 모델을 구현할 수 있습니다.

새 Amazon Cognito 사용자 풀 자격 증명 소스에는 다음 정보가 필요합니다.

- The AWS 리전.
- 사용자 풀 ID입니다.
- ID 소스와 연결하려는 사용자 엔티티 유형 (예:)MyCorp::User.
- ID 소스와 연결하려는 그룹 엔티티 유형을 예로 들 수 MyCorp::UserGroup 있습니다.
- (선택 사항) 정책 저장소에 요청을 보낼 수 있도록 권한을 부여하려는 사용자 풀의 클라이언트IDs.

검증된 권한은 동일한 Amazon Cognito 사용자 풀에서만 작동하기 때문입니다. AWS 계정다른 계정 에서는 자격 증명 소스를 지정할 수 없습니다. Verified Permissions는 예를 들어 사용자 풀 보안 주체 에 적용되는 정책에서 참조해야 하는 ID 소스 식별자인 엔티티 접두사를 사용자 풀의 ID로 설정합니다. us-west-2\_EXAMPLE

사용자 풀 토큰 클레임에는 특성, 범위, 그룹, 클라이언트 및 사용자 지정 데이터가 포함될 수 있습니다. IDs Amazon JWTs Cognito는 인증 결정에 기여할 수 있는 다양한 정보를 검증된 권한에 포함할 수 있 습니다. 다음이 포함됩니다.

- 1. 접두사가 cognito: 있는 사용자 이름 및 그룹 클레임
- 2. 사용자 지정 속성 (A) custom: prefix
- 3. 런타임에 추가된 커스텀 클레임
- 4. OIDCsub및 와 같은 표준 클레임 email

에서는 이러한 클레임을 자세히 다루고 검증 권한 정책에서 이러한 클레임을 관리하는 방법을 설명합 니다ID 제공자 토큰을 스키마에 매핑.

#### ♠ Important

Amazon Cognito 토큰은 만료되기 전에 취소할 수 있지만JWTs, 토큰은 서명과 유효 기간이 있 는 독립형 상태 비저장 리소스로 간주됩니다. JSON웹 토큰 RFC 7519를 준수하는 서비스는 원격으로 토큰을 검증해야 하며 발급자와 함께 토큰을 검증할 필요는 없습니다. 즉. Verified Permissions는 취소되었거나 나중에 삭제된 사용자에게 발급된 토큰을 기반으로 액세스 권한 을 부여할 수 있습니다. 이러한 위험을 줄이려면 유효 기간이 가장 짧은 토큰을 생성하고 사용 자 세션을 계속할 수 있는 권한을 제거하려는 경우 새로 고침 토큰을 취소하는 것이 좋습니다.

Verified Permissions의 사용자 풀 ID 소스에 대한 Cedar 정책은 영숫자 및 밑줄 () 이외의 문자를 포함하는 클레임 이름에 특수 구문을 사용합니다. \_ 여기에는 및 와 같은 문자가 포함된 사용자 풀 접두사 클레임이 포함됩니다.: cognito:username custom:department cognito:username또는 custom:department 클레임을 참조하는 정책 조건을 principal["custom:department"] 작성하려면 각각 principal["cognito:username"] 및 로 작성하십시오.

#### Note

토큰에 custom: 접두사가 cognito: or인 클레임과 리터럴 값이 cognito 포함된 클레임이름이 포함된 경우 a를 사용한 승인 <u>IsAuthorizedWithToken</u>요청은 실패합니다. custom ValidationException

다음 예는 보안 주체와 관련된 일부 Amazon Cognito 사용자 풀 클레임을 참조하는 정책을 생성하는 방법을 보여줍니다.

```
permit(
    principal == ExampleCo::User::"us-east-1_example|4fe90f4a-ref8d9-4033-
a750-4c8622d62fb6",
    action,
    resource == ExampleCo::Photo::"VacationPhoto94.jpg"
)
when {
    principal["cognito:username"]) == "alice" &&
    principal["custom:department"]) == "Finance"
};
```

클레임 매핑에 대한 자세한 내용은 을 참조하십시오<u>ID 토큰을 스키마에 매핑하기</u>. Amazon Cognito 사용자의 권한 부여에 대한 자세한 내용은 Amazon Cognito <u>개발자 안내서의 Amazon 검증 권한을 사용</u>한 권한 부여를 참조하십시오.

## 자격 증명 소스 사용 OIDC

또한 모든 호환 OpenID OIDC Connect () IdP를 정책 저장소의 ID 소스로 구성할 수 있습니다. OIDC 공급자는 Amazon Cognito 사용자 풀과 비슷합니다. 즉, 인증 JWTs 제품으로 생성된다는 점입니다. OIDC공급자를 추가하려면 발급자를 제공해야 합니다. URL

새 OIDC ID 소스에는 다음 정보가 필요합니다.

자격 증명 소스 사용 OIDC 88

• 발급자. URL 검증된 권한은 이 URL 시점에서 .well-known/openid-configuration 엔드포인 트를 발견할 수 있어야 합니다.

- CNAME와일드카드가 포함되지 않은 레코드. 예를 들어 a.example.com 매핑할 \*.example.net 수 없습니다. 반대로 \*.example.com 매핑할 수는 없습니다. a.example.net
- 권한 부여 요청에 사용하려는 토큰 유형. 이 경우에는 ID 토큰을 선택했습니다.
- ID 소스와 연결하려는 사용자 엔티티 유형 (예:)MyCorp::User.
- ID 소스와 연결하려는 그룹 엔티티 유형을 예로 들 수 MyCorp::UserGroup 있습니다.
- ID 토큰의 예 또는 ID 토큰의 클레임 정의.
- 사용자 및 그룹 IDs 엔티티에 적용하려는 접두사. CLI및 API 에서 이 접두사를 선택할 수 있습니다. 예를 들어 API게이트웨이로 설정, ID 소스 또는 가이드 설치 옵션을 사용하여 생성한 정책 저장소에서는 Verified Permissions가 발급자 이름에서 https:// 빼기 접두사를 할당합니다 (예:). MyCorp::User::"auth.example.com|a1b2c3d4-5678-90ab-cdef-EXAMPLE11111"

OIDCID 소스를 사용한 권한 부여에는 사용자 풀 ID 소스 (및) 와 동일한 API 작업이 사용됩니다. IsAuthorizedWithTokenBatchIsAuthorizedWithToken

다음 예는 회계 부서에 근무하지 않는 직원의 연말 보고서 액세스를 허용하고 기밀 분류를 적용하는 정책을 만드는 방법을 보여줍니다. 검증된 권한은 이러한 속성을 주도자의 ID 토큰에 있는 클레임에서 파생됩니다.

```
permit(
    principal in MyCorp::UserGroup::"MyOIDCProvider|Accounting",
    action,
    resource in MyCorp::Folder::"YearEnd2024"
) when {
    principal.jobClassification == "Confidential" &&
    !(principal.location like "SatelliteOffice*")
};
```

## 클라이언트 및 잠재고객 검증

정책 저장소에 ID 소스를 추가하면 Verified Permissions에는 ID 및 액세스 토큰이 의도한 대로 사용되고 있는지 확인하는 구성 옵션이 있습니다. 이 검증은 IsAuthorizedWithToken 및 BatchIsAuthorizedWithToken API 요청을 처리할 때 수행됩니다. 동작은 ID와 액세스 토큰, Amazon OIDC Cognito와 자격 증명 소스 간에 다릅니다. Amazon Cognito 사용자 풀 공급자를 사용하

클라이언트 및 잠재고객 검증 89

면 검증된 권한으로 ID 및 액세스 토큰 모두에서 클라이언트 ID를 검증할 수 있습니다. OIDC공급자를 통해 검증된 권한은 ID 토큰으로 클라이언트 ID를. 액세스 토큰으로 대상 그룹을 검증할 수 있습니다.

클라이언트 ID는 예를 들어 공급자와 함께 구성된 OAuth 또는 OIDC 애플리케이션과 관련된 식별자입니다1example23456789. 오디언스는 예를 들어 https://myapplication.example.com 대상애플리케이션의 의도된 신뢰 당사자 또는 목적지와 관련된 URL 경로입니다. aud소유권 주장이 잠재고객과 항상 연관되는 것은 아닙니다.

인증된 권한은 다음과 같이 ID 소스 대상 및 클라이언트 검증을 수행합니다.

#### Amazon Cognito

Amazon Cognito ID 토큰에는 3 = 10000 ID가 포함된 aud 클레임이 있습니다. 액세스 토큰에는 앱 클라이언트 ID도 포함된 client\_id 클레임이 있습니다.

ID 소스에 클라이언트 애플리케이션 유효성 검사를 위해 하나 이상의 값을 입력하면 Verified Permissions는 이 앱 클라이언트 IDs 목록을 ID 토큰 aud 클레임 또는 액세스 토큰 client\_id 클레임과 비교합니다. 검증된 권한은 Amazon URL Cognito ID 소스에 대한 신뢰 대상 사용자의 유효성을 확인하지 않습니다.

#### OIDC

OIDCID 토큰에는 클라이언트 목록이 포함된 aud 클레임이 있습니다. IDs 액세스 토큰에는 토큰의 대상이 URL 포함된 aud 클레임이 있습니다. 액세스 토큰에는 의도한 클라이언트 ID가 포함된 client id 클레임도 있습니다.

OIDC공급자를 통한 Audience 검증을 위해 하나 이상의 값을 입력할 수 있습니다. 토큰 유형의 ID 토큰을 선택하면 Verified Permissions는 aud 클레임에 포함된 클라이언트 구성원 중 한 명 이상이 대상 유효성 검사 값과 일치하는지 확인하여 클라이언트 IDs ID를 검증합니다.

인증된 권한은 aud 청구가 대상 유효성 검사 값과 일치하는지 확인하여 대상 사용자의 액세스 토 큰을 검증합니다. 이 액세스 토큰 값은 주로 클레임에서 가져오지만 aud 클레임이 없는 aud 경우 cid or client\_id 클레임에서 나올 수 있습니다. 올바른 오디언스 클레임 및 형식은 IdP에 문의하세요.

ID 토큰 오디언스 검증 값의 예는 다음과 같습니다1example23456789.

액세스 토큰 오디언스 검증 값의 예는 다음과 같습니다https://myapplication.example.com.

클라이언트 및 잠재고객 검증 90

## 클라이언트 측 권한 부여: JWTs

정책 저장소 ID 소스를 사용하지 않고 애플리케이션에서 JSON 웹 토큰을 처리하고 해당 클레임을 Verified Permissions에 전달하는 것이 좋습니다. JSON웹 토큰 (JWT) 에서 엔티티 속성을 추출하고 이를 검증된 권한으로 파싱할 수 있습니다.

이 예제에서는 OIDC IdP에서 검증된 권한을 호출하는 방법을 보여줍니다. 1

```
async function authorizeUsingJwtToken(jwtToken) {
    const payload = await verifier.verify(jwtToken);
    var principalEntity = {
        entityType: "PhotoFlash::User", // the application needs to fill in the
 relevant user type
        entityId: payload["sub"], // the application need to use the claim that
 represents the user-id
    };
    var resourceEntity = {
        entityType: "PhotoFlash::Photo", //the application needs to fill in the
 relevant resource type
        entityId: "jane_photo_123.jpg", // the application needs to fill in the
 relevant resource id
    };
    var action = {
        actionType: "PhotoFlash::Action", //the application needs to fill in the
 relevant action id
        actionId: "GetPhoto", //the application needs to fill in the relevant action
 type
    };
    var entities = {
        entityList: [],
    };
    entities.entityList.push(...getUserEntitiesFromToken(payload));
    var policyStoreId = "PSEXAMPLEabcdefq111111"; // set your own policy store id
    const authResult = await client
        .isAuthorized({
        policyStoreId: policyStoreId,
        principal: principalEntity,
        resource: resourceEntity,
        action: action,
        entities,
```

클라이언트 측 권한 부여: JWTs 9º

```
})
        .promise();
    return authResult;
}
function getUserEntitiesFromToken(payload) {
  let attributes = {};
  let claimsNotPassedInEntities = ['aud', 'sub', 'exp', 'jti', 'iss'];
  Object.entries(payload).forEach(([key, value]) => {
    if (claimsNotPassedInEntities.includes(key)) {
        return;
    }
    if (Array.isArray(value)) {
      var attibuteItem = [];
      value.forEach((item) => {
        attibuteItem.push({
          string: item,
        });
      });
      attributes[key] = {
        set: attibuteItem,
      };
    } else if (typeof value === 'string') {
      attributes[key] = {
        string: value,
    } else if (typeof value === 'bigint' || typeof value ==='number') {
        attributes[key] = {
            long: value,
    } else if (typeof value === 'boolean') {
        attributes[key] = {
            boolean: value,
       }
    }
  });
  let entityItem = {
    attributes: attributes,
    identifier: {
      entityType: "PhotoFlash::User",
```

클라이언트 측 권한 부여: JWTs

```
entityId: payload["sub"], // the application needs to use the claim that
represents the user-id
    }
};
return [entityItem];
}
```

¹ 이 코드 예제는 <u>aws-jwt-verify</u>라이브러리를 사용하여 signby -compatible 여부를 검증합니다JWTs. OIDC IdPs

## Amazon Verified Permissions 자격 증명 소스 생성

다음 절차는 기존 정책 저장소에 자격 증명 소스를 추가합니다. ID 소스를 추가한 후에는 <u>스키마에 속</u>성을 추가해야 합니다.

Verified Permissions 콘솔에서 <u>새 정책 저장소를 생성할</u> 때 ID 소스를 생성할 수도 있습니다. 이 프로세스에서 ID 소스 토큰의 클레임을 자동으로 엔티티 속성으로 가져올 수 있습니다. 가이드 설정 또는 API 게이트웨이 및 ID 제공자로 설정 옵션을 선택합니다. 또한 이러한 옵션은 초기 정책을 생성합니다.

### Note

정책 스토어를 생성하기 전에는 왼쪽 탐색 창에서 자격 증명 소스를 사용할 수 없습니다. 생성한 자격 증명 소스는 현재 정책 스토어와 연결됩니다.

AWS CLI 또는 <u>CreateIdentitySource</u>유효성이 확인된 권한으로 <u>create-identity-source</u>ID 소스를 만들때는 보안 주체 유형을 생략할 수 API 있습니다. 하지만 엔티티 유형이 비어 있으면 엔티티 유형이 인ID 소스가 생성됩니다AWS::Cognito. 이 엔티티 이름은 정책 저장소 스키마와 호환되지 않습니다. Amazon Cognito ID를 정책 스토어 스키마와 통합하려면 보안 주체 유형을 지원되는 정책 저장소 엔티티로 설정해야 합니다.

#### 주제

- 아마존 Cognito 자격 증명 소스
- OIDCID 소스

지격 증명 소스 생성 93

## 아마존 Cognito 자격 증명 소스

#### **AWS Management Console**

Amazon Cognito 사용자 풀 자격 증명 소스를 생성하려면

1. 에서 검증된 권한 콘솔을 <a href="https://console.aws.amazon.com/verifiedpermissions/">https://console.aws.amazon.com/verifiedpermissions/</a>여십시오. 정책 저장소를 선택합니다.

- 2. 왼쪽 탐색 창에서 자격 증명 소스를 선택합니다.
- 3. 자격 증명 소스 생성을 선택합니다.
- 4. Cognito 사용자 풀 세부 정보에서 자격 증명 소스의 사용자 풀 ID를 AWS 리전 선택하고 입력합니다.
- 5. 기본 구성에서 ID 소스의 보안 주체 유형을 선택합니다. 연결된 Amazon Cognito 사용자 풀의 자격 증명은 선택한 보안 주체 유형에 매핑됩니다.
- 6. 그룹 구성에서 사용자 풀 cognito:groups 클레임을 매핑하려면 Cognito group 사용을 선택합니다. 주도자 유형의 상위 항목인 엔티티 유형을 선택합니다.
- 7. 클라이언트 응용 프로그램 검증에서 클라이언트 응용 프로그램을 검증할지 여부를 선택합니다 IDs.
  - 클라이언트 애플리케이션을 IDs 검증하려면 일치하는 클라이언트 애플리케이션이 있는 토 큰만 수락을 선택합니다IDs. 검증할 각 클라이언트 애플리케이션 ID에 대해 새 클라이언트 애플리케이션 ID 추가를 선택합니다. 추가된 클라이언트 애플리케이션 ID를 제거하려면 클라이언트 애플리케이션 ID 옆의 제거를 선택합니다.
  - 클라이언트 애플리케이션을 검증하지 IDs 않으려면 [클라이언트 애플리케이션 검증 안 함] 을 선택합니다IDs.
- 8. 자격 증명 소스 생성을 선택합니다.
- 9. 자격 증명 또는 액세스 토큰에서 추출한 속성을 Cedar 정책에서 참조하려면 먼저 자격 증명소스가 생성하는 보안 주체 유형을 Cedar가 인식하도록 스키마를 업데이트해야 합니다. 스키마에 추가되는 항목에는 Cedar 정책에서 참조하려는 속성이 포함되어야 합니다. Amazon Cognito 토큰 속성을 Cedar 보안 주체 속성에 매핑하는 방법에 대한 자세한 내용은 ID 제공자토큰을 스키마에 매핑을(를) 참조하세요.

API연결된 정책 저장소를 생성하면 Verified Permissions는 사용자 풀에서 사용자 속성을 쿼리하고 보안 주체 유형이 사용자 풀 속성으로 채워지는 스키마를 생성합니다.

#### **AWS CLI**

Amazon Cognito 사용자 풀 자격 증명 소스를 생성하려면

작업을 사용하여 ID 원본을 만들 수 있습니다. <u>CreateIdentitySource</u> 다음 예제는 Amazon Cognito 사용자 풀의 인증된 자격 증명에 액세스할 수 있는 자격 증명 소스를 생성합니다.

다음 config.txt 파일에는 create-identity-source 명령의 --configuration 파라미터가 사용할 수 있는 Amazon Cognito 사용자 풀의 세부 정보가 들어 있습니다.

#### 명령:

```
$ aws verifiedpermissions create-identity-source \
    --configuration file://config.txt \
    --principal-entity-type "User" \
    --policy-store-id 123456789012
{
    "createdDate": "2023-05-19T20:30:28.214829+00:00",
    "identitySourceId": "ISEXAMPLEabcdefg111111",
    "lastUpdatedDate": "2023-05-19T20:30:28.214829+00:00",
    "policyStoreId": "PSEXAMPLEabcdefg111111"
}
```

자격 증명 또는 액세스 토큰에서 추출한 속성을 Cedar 정책에서 참조하려면 먼저 자격 증명 소스가 생성하는 보안 주체 유형을 Cedar가 인식하도록 스키마를 업데이트해야 합니다. 스키마에 추가되는 항목에는 Cedar 정책에서 참조하려는 속성이 포함되어야 합니다. Amazon Cognito 토큰 속성을 Cedar 보안 주체 속성에 매핑하는 방법에 대한 자세한 내용은 ID 제공자 토큰을 스키마에 매핑을 (를) 참조하세요.

API연결된 정책 저장소를 생성하면 Verified Permissions는 사용자 풀에서 사용자 속성을 쿼리하고 보안 주체 유형이 사용자 풀 속성으로 채워지는 스키마를 생성합니다.

아마존 Cognito 자격 증명 소스 9:

Verified Permissions에서 인증된 사용자를 위한 Amazon Cognito 액세스 및 자격 증명 토큰을 사용하는 방법에 대한 자세한 내용은 Amazon Cognito 개발자 안내서의 <u>Amazon Verified Permissions를 통한</u>권한 부여를 참조하세요.

### OIDCID 소스

#### **AWS Management Console**

OpenID Connect (OIDC) 아이덴티티 소스를 만들려면

- 1. 에서 검증된 권한 콘솔을 <a href="https://console.aws.amazon.com/verifiedpermissions/">https://console.aws.amazon.com/verifiedpermissions/</a>여십시오. 정책 저장소를 선택합니다.
- 2. 왼쪽 탐색 창에서 자격 증명 소스를 선택합니다.
- 3. 자격 증명 소스 생성을 선택합니다.
- 4. 외부 OIDC 공급자를 선택합니다.
- 5. 발급자에 URL URL 발급자의 이름을 입력합니다. OIDC 이는 권한 부여 서버, 서명 키 및 제공자에 대한 기타 정보 (예:) 를 제공하는 서비스 엔드포인트입니다. https:// auth.example.com 발급자는 에서 OIDC /.well-known/openid-configuration 검색 문서를 URL 호스팅해야 합니다.
- 6. 토큰 유형에서 신청서가 승인을 위해 제출하려는 유형을 선택합니다. OIDC JWT 자세한 내용은 ID 제공자 토큰을 스키마에 매핑 단원을 참조하십시오.
- 7. 사용자 및 그룹 클레임에서 ID 소스에 대한 사용자 엔티티 및 사용자 클레임을 선택합니다. 사용자 엔티티는 OIDC 공급자의 사용자에게 참조하려는 정책 저장소의 엔티티입니다. 사용자 클레임은 일반적으로 sub 평가 대상 엔티티의 고유 식별자를 보유한 ID 또는 액세스 토큰에서 제기되는 클레임입니다. 연결된 OIDC IdP의 ID가 선택한 주 유형에 매핑됩니다.
- 8. 사용자 및 그룹 클레임에서 ID 소스에 대한 그룹 엔티티와 그룹 클레임을 선택합니다. 그룹 엔티티는 사용자 엔티티의 상위 엔티티입니다. 그룹 클레임은 이 엔티티에 매핑됩니다. 그룹 클레임은 일반적으로 groups ID 또는 액세스 토큰에서 제기되는 클레임으로JSON, 평가 대상엔티티의 사용자 그룹 이름 문자열 또는 공백으로 구분된 문자열을 포함합니다. 연결된 OIDC IdP의 ID가 선택한 주 유형에 매핑됩니다.
- 9. Audience 유효성 검사에서 정책 저장소가 승인 요청에서 수락하도록 하려는 클라이언트 IDs 또는 대상 (URLs있는 경우) 을 입력합니다.
- 10. 자격 증명 소스 생성을 선택합니다.
- 11. 자격 증명 소스가 생성하는 보안 주체 유형을 Cedar가 인식하도록 스키마를 업데이트하십시오. 스키마에 추가되는 항목에는 Cedar 정책에서 참조하려는 속성이 포함되어야 합니다.

OIDCID 소스 96

Amazon Cognito 토큰 속성을 Cedar 보안 주체 속성에 매핑하는 방법에 대한 자세한 내용은 ID 제공자 토큰을 스키마에 매핑을(를) 참조하세요.

API연결된 정책 저장소를 생성하면 Verified Permissions는 사용자 풀에서 사용자 속성을 쿼리하고 보안 주체 유형이 사용자 풀 속성으로 채워지는 스키마를 생성합니다.

#### **AWS CLI**

#### ID 소스를 만들려면 OIDC

<u>CreateIdentitySource</u>작업을 사용하여 ID 소스를 만들 수 있습니다. 다음 예제는 Amazon Cognito 사용자 풀의 인증된 자격 증명에 액세스할 수 있는 자격 증명 소스를 생성합니다.

다음 config.txt 파일에는 명령의 --configuration 매개 변수에서 사용할 수 있는 OIDC IdP의 세부 정보가 들어 있습니다. create-identity-source 이 예제는 ID 토큰의 OIDC ID 소스를 만듭니다.

다음 config.txt 파일에는 명령의 --configuration 매개 변수에서 사용할 수 있는 OIDC IdP의 세부 정보가 들어 있습니다. create-identity-source 이 예제에서는 액세스 토큰의 OIDC ID 소스를 생성합니다.

```
{
   "openIdConnectConfiguration": {
     "issuer": "https://auth.example.com",
```

OIDCID 소스 97

#### 명령:

```
$ aws verifiedpermissions create-identity-source \
     --configuration file://config.txt \
     --principal-entity-type "User" \
     --policy-store-id 123456789012
{
     "createdDate": "2023-05-19T20:30:28.214829+00:00",
     "identitySourceId": "ISEXAMPLEabcdefg111111",
     "lastUpdatedDate": "2023-05-19T20:30:28.214829+00:00",
     "policyStoreId": "PSEXAMPLEabcdefg111111"
}
```

자격 증명 또는 액세스 토큰에서 추출한 속성을 Cedar 정책에서 참조하려면 먼저 자격 증명 소스가 생성하는 보안 주체 유형을 Cedar가 인식하도록 스키마를 업데이트해야 합니다. 스키마에 추가되는 항목에는 Cedar 정책에서 참조하려는 속성이 포함되어야 합니다. Amazon Cognito 토큰 속성을 Cedar 보안 주체 속성에 매핑하는 방법에 대한 자세한 내용은 ID 제공자 토큰을 스키마에 매핑을 (를) 참조하세요.

API연결된 정책 저장소를 생성하면 Verified Permissions는 사용자 풀에서 사용자 속성을 쿼리하고 보안 주체 유형이 사용자 풀 속성으로 채워지는 스키마를 생성합니다.

# Amazon Verified Permissions 자격 증명 소스 편집

자격 증명 소스를 생성한 후 자격 증명 소스의 일부 파라미터를 편집할 수 있습니다. 정책 저장소 스키마가 ID 소스 속성과 일치하는 경우 ID 소스의 변경 내용을 반영하도록 스키마를 별도로 업데이트해야합니다.

지격 증명 소스 편집 98 **98** 

#### 주제

- Amazon Cognito 사용자 풀 자격 증명 소스
- OpenID Connect (OIDC) 아이덴티티 소스

## Amazon Cognito 사용자 풀 자격 증명 소스

### **AWS Management Console**

Amazon Cognito 사용자 풀 자격 증명 소스를 업데이트하려면

- 1. 에서 검증된 권한 콘솔을 <a href="https://console.aws.amazon.com/verifiedpermissions/">https://console.aws.amazon.com/verifiedpermissions/</a>여십시오. 정책 저장소를 선택합니다.
- 2. 왼쪽 탐색 창에서 자격 증명 소스를 선택합니다.
- 3. 편집할 자격 증명 소스의 ID를 선택합니다.
- 4. 편집을 선택합니다.
- 5. Cognito 사용자 풀 세부 정보에서 자격 증명 소스의 사용자 풀 ID를 AWS 리전 선택하고 입력합니다.
- 6. 주체 세부 정보에서 ID 소스의 주체 유형을 업데이트할 수 있습니다. 연결된 Amazon Cognito 사용자 풀의 자격 증명은 선택한 보안 주체 유형에 매핑됩니다.
- 7. 그룹 구성에서 사용자 풀 cognito:groups 클레임을 매핑하려면 Cognito group 사용을 선택합니다. 주도자 유형의 상위 항목인 엔티티 유형을 선택합니다.
- 8. 클라이언트 응용 프로그램 검증에서 클라이언트 응용 프로그램을 검증할지 여부를 선택합니다 IDs.
  - 클라이언트 애플리케이션을 IDs 검증하려면 일치하는 클라이언트 애플리케이션이 있는 토 큰만 수락을 선택합니다IDs. 검증할 각 클라이언트 애플리케이션 ID에 대해 새 클라이언트 애플리케이션 ID 추가를 선택합니다. 추가된 클라이언트 애플리케이션 ID를 제거하려면 클라이언트 애플리케이션 ID 옆의 제거를 선택합니다.
  - 클라이언트 애플리케이션을 검증하지 IDs 않으려면 [클라이언트 애플리케이션 검증 안 함] 을 선택합니다IDs.
- 9. Save changes(변경 사항 저장)를 선택합니다.
- 10. 자격 증명 소스의 보안 주체 유형을 변경한 경우 업데이트된 보안 주체 유형을 올바르게 반영 하도록 스키마를 업데이트해야 합니다.

자격 증명 소스 옆에 있는 라디오 버튼을 선택한 다음 자격 증명 소스 삭제를 선택하여 자격 증명 소스를 삭제할 수 있습니다. 텍스트 상자에 delete를 입력한 다음 자격 증명 소스 삭제를 선택하여 자격 증명 소스 삭제를 확인합니다.

#### AWS CLI

Amazon Cognito 사용자 풀 자격 증명 소스를 업데이트하려면

<u>UpdateIdentitySource</u>작업을 사용하여 ID 소스를 업데이트할 수 있습니다. 다음 예제는 다른 Amazon Cognito 사용자 풀을 사용하도록 지정된 자격 증명 소스를 업데이트합니다.

다음 config.txt 파일에는 create-identity-source 명령의 --configuration 파라미터가 사용할 수 있는 Amazon Cognito 사용자 풀의 세부 정보가 들어 있습니다.

#### 명령:

```
$ aws verifiedpermissions update-identity-source \
    --update-configuration file://config.txt \
    --policy-store-id 123456789012
{
    "createdDate": "2023-05-19T20:30:28.214829+00:00",
    "identitySourceId": "ISEXAMPLEabcdefg111111",
    "lastUpdatedDate": "2023-05-19T20:30:28.214829+00:00",
    "policyStoreId": "PSEXAMPLEabcdefg111111"
}
```

자격 증명 소스의 보안 주체 유형을 변경하는 경우 업데이트된 보안 주체 유형을 올바르게 반영하 도록 스키마를 업데이트해야 합니다.

## OpenID Connect (OIDC) 아이덴티티 소스

#### **AWS Management Console**

#### OIDCID 소스를 업데이트하려면

1. 에서 검증된 권한 콘솔을 <a href="https://console.aws.amazon.com/verifiedpermissions/">https://console.aws.amazon.com/verifiedpermissions/</a>여십시오. 정책 저장소를 선택합니다.

- 2. 왼쪽 탐색 창에서 자격 증명 소스를 선택합니다.
- 3. 편집할 자격 증명 소스의 ID를 선택합니다.
- 4. 편집을 선택합니다.
- 5. OIDC제공자 세부 정보에서 발급자를 URL 필요에 따라 변경하십시오.
- 6. 토큰 클레임을 스키마 속성에 매핑에서 필요에 따라 사용자와 그룹 클레임 및 정책 저장소 엔 티티 유형 간의 연결을 변경합니다. 엔티티 유형을 변경한 후에는 정책과 스키마 속성을 업데 이트하여 새 엔티티 유형에 적용해야 합니다.
- 7. Audience 유효성 검사에서 적용하려는 잠재고객 값을 추가하거나 제거합니다.
- 8. Save changes(변경 사항 저장)를 선택합니다.

자격 증명 소스 옆에 있는 라디오 버튼을 선택한 다음 자격 증명 소스 삭제를 선택하여 자격 증명 소스를 삭제할 수 있습니다. 텍스트 상자에 delete를 입력한 다음 자격 증명 소스 삭제를 선택하여 자격 증명 소스 삭제를 확인합니다.

#### AWS CLI

#### OIDCID 소스를 업데이트하려면

<u>UpdateIdentitySource</u>작업을 사용하여 ID 소스를 업데이트할 수 있습니다. 다음 예제에서는 다른 OIDC 공급자를 사용하도록 지정된 ID 소스를 업데이트합니다.

다음 config.txt 파일에는 create-identity-source 명령의 --configuration 파라미터가 사용할 수 있는 Amazon Cognito 사용자 풀의 세부 정보가 들어 있습니다.

```
},
},

"entityIdPrefix": "MyOIDCProvider",
    "groupConfiguration": {
        "groupClaim": "groups",
        "groupEntityType": "MyCorp::UserGroup"
}
}
```

명령:

```
$ aws verifiedpermissions update-identity-source \
    --update-configuration file://config.txt \
    --policy-store-id 123456789012
{
    "createdDate": "2023-05-19T20:30:28.214829+00:00",
    "identitySourceId": "ISEXAMPLEabcdefg111111",
    "lastUpdatedDate": "2023-05-19T20:30:28.214829+00:00",
    "policyStoreId": "PSEXAMPLEabcdefg111111"
}
```

자격 증명 소스의 보안 주체 유형을 변경하는 경우 업데이트된 보안 주체 유형을 올바르게 반영하 도록 스키마를 업데이트해야 합니다.

# ID 제공자 토큰을 스키마에 매핑

ID 소스를 정책 저장소에 추가하고 공급자 클레임을 정책 저장소 스키마에 매핑하고 싶을 수도 있습니다. 이 프로세스를 자동화하거나 스키마를 수동으로 업데이트할 수 있습니다. 토큰을 스키마에 매핑한 후에는 토큰을 참조하는 정책을 만들 수 있습니다.

사용 설명서의 이 섹션에는 다음과 같은 정보가 있습니다.

- 정책 저장소 스키마에 속성을 자동으로 채울 수 있는 경우
- 검증된 권한 정책에서 Amazon Cognito 및 OIDC 토큰 클레임을 사용하는 방법
- 자격 증명 소스의 스키마를 수동으로 구축하는 방법

API<u>가이드 설정을</u> 통해 ID 원본과 함께 <u>연결된 정책 저장소와</u> 정책 저장소의 경우 ID (ID) 토큰 속성을 스키마에 수동으로 매핑할 필요가 없습니다. 사용자 풀 또는 OIDC 토큰의 속성과 함께 Verified

토큰을 스키마에 매핑 102

Permissions를 제공하고 사용자 속성으로 채워진 스키마를 생성할 수 있습니다. ID 토큰 인증에서 검증된 권한은 클레임을 보안 주체의 속성에 매핑합니다. 다음과 같은 상황에서는 Amazon Cognito 토큰을 스키마에 수동으로 매핑해야 할 수 있습니다.

- 샘플에서 빈 정책 스토어나 정책 스토어를 생성했습니다.
- 액세스 토큰 사용을 역할 기반 액세스 제어 () RBAC 이상으로 확장하고 싶습니다.
- 검증된 권한 REST API AWS SDK, an 또는 를 사용하여 정책 저장소를 생성합니다. AWS CDK

Amazon Cognito 또는 OIDC ID 공급자 (IdP) 를 검증된 권한 정책 스토어의 자격 증명 소스로 사용하려면 스키마에 공급자 속성이 있어야 합니다. ID 토큰의 공급자 정보로부터 스키마를 자동으로 채우는 방식으로 정책 스토어를 생성한 경우 정책을 작성할 준비가 된 것입니다. ID 소스에 대한 스키마 없이 정책 저장소를 생성하는 경우 스키마에 공급자 속성을 추가해야 합니다. 스키마는 제공자 토큰이 IsAuthorizedWithToken 생성하거나 BatchIsAuthorizedWithToken API요청하는 엔티티와 일치해야 합니다. 그런 다음 제공자 토큰의 속성을 사용하여 정책을 작성할 수 있습니다.

검증된 권한에서 인증된 사용자의 Amazon Cognito ID 및 액세스 토큰을 사용하는 방법에 대한 자세한 내용은 Amazon Cognito 개발자 안내서의 Amazon 검증 권한을 통한 권한 부여를 참조하십시오.

#### 주제

- 스키마 매핑에 대해 알아야 할 사항
- ID 토큰을 스키마에 매핑하기
- 액세스 토큰 매핑
- Amazon Cognito 콜론으로 구분된 클레임의 대체 표기법

# 스키마 매핑에 대해 알아야 할 사항

속성 매핑은 토큰 유형마다 다릅니다.

액세스 토큰 인증에서 검증된 권한은 클레임을 <u>컨텍스트에</u> 매핑합니다. ID 토큰 인증에서 검증된 권한은 클레임을 주요 속성에 매핑합니다. Verified Permissions 콘솔에서 생성하는 정책 저장소의 경우비어 있는 샘플 정책 저장소만 있으면 ID 소스가 없으므로 ID 토큰 인증을 위한 사용자 풀 속성으로 스키마를 채워야 합니다. 액세스 토큰 권한 부여는 그룹 구성원 클레임이 포함된 역할 기반 액세스 제어 (RBAC) 를 기반으로 하며 다른 클레임을 정책 저장소 스키마에 자동으로 매핑하지 않습니다.

ID 소스 속성은 필요하지 않습니다.

검증된 권한 콘솔에서 ID 소스를 만들 때는 어떤 속성도 필수로 표시되지 않습니다. 이렇게 하면 클레임 누락으로 인해 권한 부여 요청에서 유효성 검사 오류가 발생하는 것을 방지할 수 있습니다. 필요에따라 속성을 필수로 설정할 수 있지만 모든 권한 부여 요청에 해당 속성이 있어야 합니다.

RBAC스키마의 속성이 필요하지 않습니다.

ID 소스의 스키마는 ID 소스를 추가할 때 만드는 엔티티 연결에 따라 달라집니다. ID 소스는 클레임 하나를 사용자 엔티티 유형에 매핑하고, 다른 클레임을 그룹 엔티티 유형에 매핑합니다. 이러한 엔티티 매핑은 ID-소스 구성의 핵심입니다. 이 최소한의 정보만 있으면 역할 기반 액세스 제어 () 모델에서 특정 사용자 및 사용자가 구성원으로 속할 수 있는 특정 그룹에 대해 권한 부여 작업을 수행하는 정책을 작성할 수 있습니다. RBAC 스키마에 토큰 클레임을 추가하면 정책 저장소의 권한 부여 범위가 확장됩니다. ID 토큰의 사용자 속성에는 속성 기반 액세스 제어 () ABAC 권한 부여에 기여할 수 있는 사용자에 대한 정보가 있습니다. 액세스 토큰의 컨텍스트 속성에는 OAuth 2.0 범위와 같은 정보가 포함되어 있어 공급자로부터 추가 액세스 제어 정보를 제공할 수 있지만 추가 스키마 수정이 필요합니다.

검증된 권한 콘솔의 API게이트웨이 및 ID 소스로 설정 및 가이드 설정 옵션은 ID 토큰 클레임을 스키마에 할당합니다. 액세스 토큰 클레임의 경우는 그렇지 않습니다. 스키마에 비그룹 액세스 토큰 클레임을 추가하려면 JSON 모드에서 스키마를 편집하고 속성을 추가해야 합니다. commonTypes 자세한 내용은 액세스 토큰 매핑 단원을 참조하십시오.

OIDC그룹 클레임은 여러 형식을 지원합니다.

OIDC공급자를 추가할 때 ID 또는 액세스 토큰에서 그룹 클레임의 이름을 선택하여 정책 저장소의 사용자 그룹 멤버십에 매핑할 수 있습니다. 검증된 권한은 다음 형식의 그룹 클레임을 인식합니다.

- 1. 공백 없는 문자열: "groups": "MyGroup"
- 2. 공백으로 구분된 목록:. "groups": "MyGroup1 MyGroup2 MyGroup3" 각 문자열은 그룹입니다.
- 3. JSON(쉼표로 구분된) 목록: "groups": ["MyGroup1", "MyGroup2", "MyGroup3"]

# Note

검증된 권한은 공백으로 구분된 그룹 클레임의 각 문자열을 별도의 그룹으로 해석합니다. 공백 문자가 있는 그룹 이름을 단일 그룹으로 해석하려면 클레임에서 공백을 바꾸거나 제거하십시 오. 예를 들어, 이름이 인 그룹의 형식을 지정합니다My Group. MyGroup

토큰 유형을 선택합니다.

정책 저장소가 ID 소스와 연동되는 방식은 ID 소스 구성의 주요 결정, 즉 ID 토큰을 처리할지 액세스 토 큰을 처리할지에 따라 달라집니다. Amazon Cognito 자격 증명 공급자를 사용하면 API 연결된 정책 저장소를 생성할 때 토큰 유형을 선택할 수 있습니다. API-linked 정책 스토어를 생성할 때는 ID 또는 액세스 토큰에 대한 권한 부여를 설정할지 여부를 선택해야 합니다. 이 정보는 Verified Permissions가 정책 스토어에 적용하는 스키마 속성과 게이트웨이의 Lambda 권한 부여자 구문에 영향을 줍니다. API API OIDC공급자의 경우 ID 소스를 추가할 때 토큰 유형을 선택해야 합니다. ID 또는 액세스 토큰을 선택할 수 있으며 선택하지 않은 토큰 유형은 정책 저장소에서 처리되지 않도록 선택할 수 있습니다. 특히 Verified Permissions 콘솔에서 ID 토큰 클레임을 속성에 자동으로 매핑하는 기능을 활용하려면 ID 소스를 생성하기 전에 처리할 토큰 유형을 먼저 결정해야 합니다. 토큰 유형을 변경하려면 정책과 스키마를 리팩토링하는 데 상당한 노력이 필요합니다. 다음 항목에서는 정책 저장소에서 ID 및 액세스 토큰을 사용하는 방법을 설명합니다.

Cedar 파서의 일부 문자에는 대괄호가 필요합니다.

정책은 일반적으로 다음과 같은 형식의 스키마 속성을 참조합니다. principal.username:., 또는 같이 영숫자가 아닌 대부분의 문자가 토큰 클레임 이름에 나타나는 경우 Verified Permissions는 또는 / 같은 조건 값을 파싱할 수 없습니다. principal.cognito:username context.ip-address 대신 각각 또는 형식으로 대괄호 표기법을 사용하여 이러한 조건의 형식을 지정해야 합니다. principal["cognito:username"] context["ip-address"] 밑줄 문자는 클레임 이름에 사용할 수 있는 \_ 문자이며 이 요구 사항에서 영숫자가 아닌 유일한 예외입니다.

이 유형의 주요 속성에 대한 일부 예제 스키마는 다음과 같습니다.

```
"User": {
   "shape": {
      "type": "Record",
      "attributes": {
         "cognito:username": {
            "type": "String",
            "required": true
         },
         "custom:employmentStoreCode": {
            "type": "String",
            "required": true,
         },
         "email": {
            "type": "String",
            "required": false
         }
   }
```

}

이 유형의 컨텍스트 속성에 대한 일부 예제 스키마는 다음과 같습니다.

```
"GetOrder": {
   "memberOf": [],
   "appliesTo": {
      "resourceTypes": [
         "Order"
      ],
      "context": {
         "type": "Record",
         "attributes": {
             "ip-address": {
                "required": false,
                "type": "String"
            }
   }
   },
      "principalTypes": [
         "User"
      ]
   }
}
```

이 스키마를 기준으로 검증할 예제 정책은 을 참조하십시오<u>대괄호 표기법을 사용하여 토큰 속성을 참</u>조합니다..

# ID 토큰을 스키마에 매핑하기

검증된 권한은 ID 토큰 클레임을 사용자의 특성 (이름 및 직함, 그룹 구성원, 연락처 정보) 으로 처리합니다. ID 토큰은 속성 기반 액세스 제어 (ABAC) 인증 모델에서 가장 유용합니다. Verified Permissions에서 요청한 사람을 기반으로 리소스에 대한 액세스를 분석하도록 하려면 ID 소스의 ID 토큰을 선택하십시오.

# 아마존 Cognito ID 토큰

Amazon Cognito ID 토큰은 대부분의 OIDC 타사 라이브러리와 호환됩니다. 추가 클레임을 통해 의OIDC 기능을 확장합니다. 애플리케이션은 Amazon Cognito 사용자 풀 API 인증 작업 또는 사용자 풀호스팅 UI를 사용하여 사용자를 인증할 수 있습니다. 자세한 내용은 Amazon Cognito 개발자 <u>안내서의 API 및 엔드포인트 사용을 참조하십시오</u>.

ID 토큰 매핑 100g

#### Amazon Cognito ID 토큰의 유용한 클레임

cognito:username 및 preferred\_username

사용자 사용자 이름의 변형.

sub

사용자의 고유한 사용자 식별자 (UUID)

custom: 접두사가 있는 클레임

와 같은 사용자 지정 사용자 풀 속성의 접두사. custom:employmentStoreCode 표준 클레임

email및 와 같은 표준 OIDC 클레임phone\_number. 자세한 내용은 에라타 세트 2를 포함하는 OpenID Connect Core 1.0의 표준 클레임을 참조하십시오.

cognito:groups

사용자의 그룹 멤버십. 역할 기반 액세스 제어 (RBAC) 에 기반한 권한 부여 모델에서 이 클레임은 정책에서 평가할 수 있는 역할을 제시합니다.

#### 일시적 클레임

사용자의 속성은 아니지만 사용자 풀  $\frac{\text{N전 }}{\text{E-P}}$  생성  $\frac{\text{Lambda}}{\text{Lambda}}$  트리거에 의해 런타임에 추가되는 클레임. 일시적 클레임은 표준 클레임과 비슷하지만 표준을 벗어납니다 (예: 또는). tenant department

:구분자가 있는 Amazon Cognito 속성을 참조하는 정책에서는 형식의 속성을 참조합니다. principal["cognito:username"] roles cognito:groups 클레임은 이 규칙의 예외입니다. 검 증된 권한은 이 클레임의 내용을 사용자 엔티티의 상위 엔티티에 매핑합니다.

Amazon Cognito 사용자 풀의 ID 토큰 구조에 대한 자세한 내용은 Amazon Cognito 개발자 <u>안내서의</u> ID 토큰 사용을 참조하십시오.

다음 예제 ID 토큰에는 각각 네 가지 유형의 속성이 있습니다. 여기에는 Amazon Cognito 관련 클레임 cognito:username, 사용자 지정 클레임 custom:employmentStoreCode, 표준 클레임 email, 및 일시적 클레임 tenant이 포함됩니다.

```
{
    "sub": "91eb4550-XXX",
    "cognito:groups": [
```

ID 토큰 매핑 107

```
"Store-Owner-Role",
        "Customer"
    ],
    "email_verified": true,
    "clearance": "confidential",
    "iss": "https://cognito-idp.us-east-2.amazonaws.com/us-east-2 EXAMPLE",
    "cognito:username": "alice",
    "custom:employmentStoreCode": "petstore-dallas",
    "origin_jti": "5b9f50a3-05da-454a-8b99-b79c2349de77",
    "aud": "1example23456789",
    "event_id": "0ed5ad5c-7182-4ecf-XXX",
    "token_use": "id",
    "auth_time": 1687885407,
    "department": "engineering",
    "exp": 1687889006,
    "iat": 1687885407,
    "tenant": "x11app-tenant-1",
    "jti": "a1b2c3d4-e5f6-a1b2-c3d4-T0KEN1111111",
    "email": "alice@example.com"
}
```

Amazon Cognito 사용자 풀을 사용하여 자격 증명 소스를 생성할 때는 인증 요청에서 확인된 권한이 생성하는 주요 개체의 유형을 지정합니다. IsAuthorizedWithToken 그러면 정책을 통해 해당 요청 평가의 일환으로 해당 보안 주체의 속성을 테스트할 수 있습니다. 스키마는 자격 증명 소스의 주체 유형과 속성을 정의하며, 이를 Cedar 정책에서 참조할 수 있습니다.

또한 ID 토큰 그룹 클레임에서 파생하려는 그룹 엔티티의 유형을 지정합니다. 권한 부여 요청에서 검증된 권한은 그룹 클레임의 각 구성원을 해당 그룹 엔티티 유형에 매핑합니다. 정책에서 해당 그룹 엔티티를 주도자로 참조할 수 있습니다.

다음 예는 Verified Permissions 스키마에 자격 증명 토큰 예제의 속성을 반영하는 방법을 보여줍니다. 스키마 편집에 대한 자세한 내용은 <u>모드에서 정책 저장소 스키마를 편집합니다JSON.</u>을(를) 참조하세요. 자격 증명 소스 구성에서 보안 주체 유형 User를 지정하면 다음 예제와 유사한 내용을 포함하여 Cedar에서 해당 속성을 사용할 수 있도록 할 수 있습니다.

ID 토큰 매핑 108

```
},
    "custom:employmentStoreCode": {
        "type": "String",
        "email": {
            "type": "String"
        },
        "tenant": {
            "type": "String",
            "required": true
        }
    }
}
```

이 스키마를 기준으로 검증할 정책의 예는 을 참조하십시오<u>아마존 Cognito ID 토큰 속성을 반영합니</u>다..

#### OIDCID 토큰

OIDC공급자의 ID 토큰으로 작업하는 것은 Amazon Cognito ID 토큰을 사용하는 것과 거의 같습니다. 차이점은 클레임에 있습니다. IdP는 <u>표준 OIDC 속성을</u> 제공하거나 사용자 지정 스키마를 가질 수 있습니다. Verified Permissions 콘솔에서 새 정책 저장소를 생성할 때 예제 ID 토큰을 사용하여 ID 소스를 추가하거나 토큰 클레임을 사용자 속성에 수동으로 매핑할 수 있습니다. OIDC 검증된 권한은 IdP의 속성 스키마를 인식하지 못하므로 이 정보를 제공해야 합니다.

자세한 내용은 Verified Permissions 정책 스토어 생성 단원을 참조하십시오.

다음은 OIDC ID 소스가 있는 정책 저장소의 예제 스키마입니다.

ID 토큰 매핑 109

이 스키마를 기준으로 검증할 정책의 예는 을 참조하십시오OIDCID 토큰 속성 반영.

### 액세스 토큰 매핑

검증된 권한은 그룹 클레임 이외의 액세스 토큰 클레임을 작업의 속성 또는 컨텍스트 속성으로 처리합니다. 그룹 멤버십과 함께 IdP의 액세스 토큰에는 액세스에 대한 API 정보가 포함될 수 있습니다. 액세스 토큰은 역할 기반 액세스 제어 () 를 사용하는 권한 부여 모델에 유용합니다. RBAC 그룹 구성원 자격 이외의 액세스 토큰 클레임에 의존하는 권한 부여 모델은 스키마 구성에 추가 작업이 필요합니다.

### Amazon Cognito 액세스 토큰 매핑

Amazon Cognito 액세스 토큰에는 권한 부여에 사용할 수 있는 클레임이 있습니다.

Amazon Cognito 액세스 토큰의 유용한 클레임

# client\_id

OIDC신뢰 당사자의 클라이언트 애플리케이션 ID. 검증된 권한은 클라이언트 ID를 사용하여 정책 저장소에 대해 허용된 클라이언트에서 권한 부여 요청이 왔는지 확인할 수 있습니다. machine-to-machine (M2M) 권한 부여에서 요청 시스템은 클라이언트 암호를 사용하여 요청을 승인하고 클라이언트 ID와 범위를 권한 부여의 증거로 제공합니다.

#### scope

OAuth2.0 범위는 토큰 전달자의 액세스 권한을 나타냅니다.

#### cognito:groups

사용자의 그룹 멤버십. 역할 기반 액세스 제어 (RBAC) 에 기반한 권한 부여 모델에서 이 클레임은 정책에서 평가할 수 있는 역할을 제시합니다.

#### 일시적 클레임

액세스 권한은 아니지만 사용자 풀 <u>사전 토큰 생성 Lambda</u> 트리거에 의해 런타임에 추가되는 클레임. 일시적 클레임은 표준 클레임과 비슷하지만 표준을 벗어납니다 (예: 또는). tenant department 액세스 토큰을 사용자 지정하면 청구서에 비용이 추가됩니다. AWS

Amazon Cognito 사용자 풀의 액세스 토큰 구조에 대한 자세한 내용은 Amazon Cognito 개발자  $\underline{\text{안내서}}$ 의 액세스 토큰 사용을 참조하십시오.

Amazon Cognito 액세스 토큰은 Verified Permissions으로 전달될 때 컨텍스트 객체에 매핑됩니다. context.token.attribute\_name을 사용하여 액세스 토큰의 속성을 참조할 수 있습니다. 다음 액세스 토큰 예제에는 client\_id 및 scope 클레임이 모두 포함됩니다.

```
{
    "sub": "91eb4550-9091-708c-a7a6-9758ef8b6b1e",
    "cognito:groups": [
        "Store-Owner-Role",
        "Customer"
    ],
    "iss": "https://cognito-idp.us-east-2.amazonaws.com/us-east-2_EXAMPLE",
    "client_id": "1example23456789",
    "origin_jti": "a1b2c3d4-e5f6-a1b2-c3d4-T0KEN1111111",
    "event_id": "bda909cb-3e29-4bb8-83e3-ce6808f49011",
    "token_use": "access",
    "scope": "MyAPI/mydata.write",
    "auth_time": 1688092966,
    "exp": 1688096566,
    "iat": 1688092966,
    "jti": "a1b2c3d4-e5f6-a1b2-c3d4-TOKEN2222222",
    "username": "alice"
}
```

다음 예는 Verified Permissions 스키마에 액세스 토큰 예제의 속성을 반영하는 방법을 보여줍니다. 스키마 편집에 대한 자세한 내용은 모드에서 정책 저장소 스키마를 편집합니다JSON.을(를) 참조하세요.

```
"type": "ReusedContext"
               },
               "resourceTypes": [
                   "Application"
               ],
               "principalTypes": [
                   "User"
               ]
            }
         }
      },
      "commonTypes": {
         "ReusedContext": {
            "attributes": {
               "token": {
                   "type": "Record",
                   "attributes": {
                      "scope": {
                         "type": "Set",
                         "element": {
                            "type": "String"
                         }
                      },
                      "client_id": {
                         "type": "String"
                   }
               }
            },
            "type": "Record"
      }
   }
}
```

이 스키마를 기준으로 유효성을 검사하는 예제 정책은 을 참조하십시오. <u>Amazon Cognito 액세스 토큰</u> 속성을 반영합니다.

#### OIDC액세스 토큰 매핑

외부 OIDC 공급자가 제공하는 대부분의 액세스 토큰은 Amazon Cognito 액세스 토큰과 밀접하게 일치합니다. 검증된 권한으로 전달되면 OIDC 액세스 토큰이 컨텍스트 객체에 매핑됩니다. context.token.attribute\_name을 사용하여 액세스 토큰의 속성을 참조할 수 있습니다. 다음 예제 OIDC 액세스 토큰에는 예제 기본 클레임이 포함되어 있습니다.

다음 예는 Verified Permissions 스키마에 액세스 토큰 예제의 속성을 반영하는 방법을 보여줍니다. 스키마 편집에 대한 자세한 내용은 모드에서 정책 저장소 스키마를 편집합니다JSON.을(를) 참조하세요.

```
},
      . . .
      "commonTypes": {
          "ReusedContext": {
             "attributes": {
                "token": {
                   "type": "Record",
                   "attributes": {
                       "scope": {
                          "type": "Set",
                          "element": {
                             "type": "String"
                          }
                       },
                       "client_id": {
                          "type": "String"
                   }
                }
             },
             "type": "Record"
         }
      }
   }
}
```

이 스키마를 기준으로 유효성을 검사할 정책의 예는 을 참조하십시오<u>OIDC액세스 토큰 속성을 반영합</u>니다..

# Amazon Cognito 콜론으로 구분된 클레임의 대체 표기법

검증된 권한 출시 당시 Amazon Cognito 토큰 클레임에 대한 권장 스키마는 콜론으로 구분된 문자열을 cognito:groups 반환하고 해당 문자를 계층 구분 기호로 사용하도록 custom:store 변환했습니다. 이 형식을 점 표기법이라고 합니다. 예를 들어, 정책에 대한 cognito:groups principal.cognito.groups 참조가 포함되었습니다. 이 형식을 계속 사용할 수 있지만 대괄호표기법을 사용하여 스키마와 정책을 작성하는 것이 좋습니다. 이 형식에서는 에 대한 cognito:groups 참조가 principal["cognito:groups"] 정책에 포함됩니다. Verified Permissions 콘솔에서 사용자 풀 ID 토큰에 대해 자동으로 생성되는 스키마는 괄호 표기법을 사용합니다.

Amazon Cognito 자격 증명 소스에 대해 수동으로 구축한 스키마 및 정책에서 점 표기법을 계속 사용할수 있습니다. 다른 유형의 IdP에 대해서는 스키마 : 또는 정책에서 점 표기법이나 영숫자가 아닌 다른 문자를 사용할 수 없습니다. OIDC

점 표기법 스키마는 다음 예와 같이 : 문자의 각 인스턴스를 cognito 또는 custom 첫 구문의 하위 문으로 중첩합니다.

```
"CognitoUser": {
   "shape": {
      "type": "Record",
      "attributes": {
         "cognito": {
            "type": "Record",
            "required": true,
            "attributes": {
                "username": {
                   "type": "String",
                   "required": true
               }
            }
         },
         "custom": {
            "type": "Record",
            "required": true,
            "attributes": {
                "employmentStoreCode": {
                   "type": "String",
                   "required": true
               }
            }
         },
         "email": {
            "type": "String"
         },
         "tenant": {
            "type": "String",
            "required": true
         }
      }
   }
}
```

이 스키마를 기준으로 유효성을 검사하고 점 표기법을 사용하는 예제 정책은 을 참조하십시오. <u>점 표기법을 사용하여 속성을 참조합니다.</u>

# Amazon 검증 권한에서 권한 부여 구현

정책 스토어, 정책, 템플릿, 스키마 및 권한 부여 모델을 구축했으면 Amazon Verified Permissions를 사용하여 요청을 승인할 준비가 된 것입니다. 검증된 권한 인증을 구현하려면 정책 구성을 애플리케이션에서의 AWS 통합과 결합해야 합니다. 검증된 권한을 애플리케이션에 통합하려면 검증된 권한을 AWS SDK 호출하고 정책 저장소에 대한 권한 API 부여 결정을 생성하는 메서드를 추가하고 구현하십시오.

검증된 권한을 통한 승인은 애플리케이션의 UX API 권한 및 권한에 유용합니다.

#### UX 권한

애플리케이션 UX에 대한 사용자 액세스를 제어합니다. 사용자가 액세스해야 하는 정확한 양식, 버튼, 그래픽 및 기타 리소스만 보도록 허용할 수 있습니다. 예를 들어 사용자가 로그인할 때 계정에 '자금 이체' 버튼이 표시되는지 확인하고 싶을 수 있습니다. 또한 사용자가 취할 수 있는 조치를 제어할 수 있습니다. 예를 들어 동일한 뱅킹 앱에서 사용자가 거래 범주를 변경할 수 있는지 여부를 확인하고 싶을 수 있습니다.

#### API권한

데이터에 대한 사용자 액세스를 제어합니다. 애플리케이션은 분산 시스템의 일부인 경우가 많으며 외부로부터 정보를 가져옵니다APIs. Verified Permissions에서 "자금 이체" 버튼 표시를 허용한 은행 앱의 예에서는 사용자가 이체를 시작할 때 더 복잡한 승인 결정을 내려야 합니다. 인증된 권한은이체 대상이 되는 대상 계정을 나열하는 API 요청을 승인한 다음 다른 계좌로 이체를 푸시해 달라는요청을 승인할 수 있습니다.

이 콘텐츠를 설명하는 예는 <u>샘플 정책</u> 저장소에서 가져온 것입니다. 따라하려면 테스트 환경에 DigitalPetStore샘플 정책 저장소를 생성하세요.

일괄 인증을 사용하여 UX 권한을 구현하는 엔드 투 엔드 샘플 애플리케이션에 대해서는 보안 <u>블로그에</u> 서 Amazon Verified Permissions 사용을 통해 대규모 세분화된 권한 부여를 참조하십시오.AWS

#### 주제

- 권한 부여에 사용할 수 있는 작업 API
- 권한 부여 모델 테스트
- 인증 모델을 애플리케이션과 통합

# 권한 부여에 사용할 수 있는 작업 API

검증된 API 권한에는 다음과 같은 권한 부여 작업이 있습니다.

#### **IsAuthorized**

IsAuthorizedAPI작업은 검증된 권한이 있는 권한 부여 요청의 진입점입니다. 주도자, 작업, 리소스, 컨텍스트 및 엔티티 요소를 제출해야 합니다. 검증된 권한은 정책 저장소 스키마를 기준으로 요청의 엔티티를 검증합니다. 그런 다음 Verified Permissions는 요청의 엔티티에 적용되는 요청된 정책 저장소의 모든 정책을 기준으로 요청을 평가합니다.

#### **IsAuthorizedWithToken**

이 IsAuthorizedWithToken 작업은 Amazon Cognito JSON 웹 토큰 () JWTs 의 사용자 데이터 로부터 권한 부여 요청을 생성합니다. 검증된 권한은 Amazon Cognito를 정책 저장소의 자격 증명소스로 사용하여 직접 작동합니다. 검증된 권한은 요청의 보안 주체에 대한 모든 속성을 사용자 ID 또는 액세스 토큰의 클레임으로 채웁니다. Amazon Cognito 사용자 풀의 사용자 속성 또는 그룹 멤버십에서 작업 및 리소스를 승인할 수 있습니다.

요청에는 그룹 또는 사용자 보안 주체 유형에 대한 정보를 포함할 수 없습니다.

IsAuthorizedWithToken 제공한 데이터로 모든 보안 주체 데이터를 채워야 합니다JWT.

#### BatchIsAuthorized

이 BatchIsAuthorized 작업에서는 단일 주체 또는 리소스에 대한 여러 권한 부여 결정을 단일 API 요청으로 처리합니다. 이 작업은 요청을 단일 일괄 작업으로 그룹화하여 <u>할당량 사용을</u> 최소화하고 최대 30개의 복잡한 중첩 작업 각각에 대해 권한 부여 결정을 반환합니다. 단일 리소스에 대한 일괄 권한 부여를 통해 사용자가 리소스에서 수행할 수 있는 작업을 필터링할 수 있습니다. 단일 보안 주체에 대한 일괄 권한 부여를 통해 사용자가 조치를 취할 수 있는 리소스를 필터링할 수 있습니다. 다일 보다.

#### BatchIsAuthorizedWithToken

이 BatchIsAuthorizedWithToken 작업은 단일 보안 주체에 대한 여러 권한 부여 결정을 한 번의 API 요청으로 처리합니다. 보안 주체는 정책 저장소 ID 소스에서 ID 또는 액세스 토큰으로 제공합니다. 이 작업은 요청을 단일 일괄 작업으로 그룹화하여 <u>할당량 사용을</u> 최소화하고 최대 30개의 작업 및 리소스 요청 각각에 대해 권한 부여 결정을 반환합니다. 정책에서 Amazon Cognito 사용자풀의 속성 또는 그룹 멤버십을 통한 액세스를 승인할 수 있습니다.

와 IsAuthorizedWithToken 마찬가지로 요청에는 그룹 또는 사용자 보안 주체 유형에 대한 정보를 포함할 수 없습니다. BatchIsAuthorizedWithToken 모든 보안 주체 데이터를 제공한 데이터로 채워야 합니다JWT.

API오퍼레이션 118

# 권한 부여 모델 테스트

애플리케이션을 배포할 때 Amazon 검증 권한 승인 결정이 미치는 영향을 이해하려면 검증된 권한에 대한 Amazon 검증 권한 테스트 벤치 사용 HTTPS REST API 요청과 함께 정책을 개발할 때 정책을 평가해 볼 수 있습니다. 테스트 벤치는 정책 저장소의 권한 부여 요청 및 응답을 AWS Management Console 평가하기 위한 도구입니다.

검증된 REST API 권한은 개념적 이해에서 애플리케이션 설계로 넘어가는 개발의 다음 단계입니다. 검증된 권한은 지역 <u>서비스</u> 엔드포인트에 대한 권한 부여 요청 및 <u>서명된 BatchlsAuthorizedAWS API요</u> <u>청으로 승인 요청을 API</u> 수락합니다. <u>IsAuthorizedIsAuthorizedWithToken</u> 권한 부여 모델을 테스트하려면 모든 API 클라이언트를 대상으로 요청을 생성하고 정책이 예상대로 권한 부여 결정을 반환하는지확인할 수 있습니다.

예를 들어, 다음 절차에 따라 샘플 정책 IsAuthorized 저장소에서 테스트할 수 있습니다.

#### Test bench

- 1. 에서 검증된 권한 콘솔을 엽니다 <a href="https://console.aws.amazon.com/verifiedpermissions/">https://console.aws.amazon.com/verifiedpermissions/</a>. 샘플 정책 저장소에서 이름을 사용하여 정책 저장소를 생성합니다 DigitalPetStore.
- 2. 새 정책 저장소에서 테스트 벤치를 선택합니다.
- 3. 검증된 권한 <u>IsAuthorized</u>API참조에서 테스트 벤치 요청을 채웁니다. 다음 세부 정보는 샘플을 참조하는 예제 4의 조건을 복제합니다 DigitalPetStore.
  - a. Alice를 교장으로 설정합니다. 교장이 조치를 취하도록 하려면 DigitalPetStore::User 선택하고 입력하십시오Alice.
  - b. Alice의 고객 역할을 설정합니다. [부모 추가] 를 선택하고 [DigitalPetStore::Role고객]을 입력합니다.
  - c. 리소스를 주문 "1234"로 설정합니다. 주도자가 활동 중인 리소스에 대해 DigitalPetStore::Order 선택하고 입력합니다1234.
  - d. DigitalPetStore::Order리소스에는 owner 속성이 필요합니다. Alice를 주문 소유자로 설정합니다. 선택 DigitalPetStore::User 후 입력 Alice
  - e. Alice가 주문 보기를 요청했습니다. 해당 교장이 취하는 조치에 대해 선택하세 요DigitalPetStore::Action::"GetOrder".
- 4. 권한 부여 요청 실행을 선택합니다. 수정되지 않은 정책 저장소에서는 이 요청으로 인해 ALLOW 결정이 내려집니다. 결정을 반환한 만족스러운 정책을 기록해 두십시오.
- 5. 왼쪽 탐색 메뉴에서 정책을 선택합니다. 고객 역할 주문 받기에 대한 설명과 함께 정적 정책을 검토하십시오.

테스트 모델 119

6. 보안 주체가 고객 역할을 담당하고 리소스의 소유자이기 때문에 확인된 권한이 요청을 허용했다는 점을 살펴보세요.

#### **REST API**

- 1. 에서 검증된 권한 콘솔을 <a href="https://console.aws.amazon.com/verifiedpermissions/">https://console.aws.amazon.com/verifiedpermissions/</a>여십시오. 샘플 정책 저장소에서 이름을 사용하여 정책 저장소를 생성합니다 DigitalPetStore.
- 2. 새 정책 저장소의 정책 저장소 ID를 기록해 둡니다.
- 3. 검증된 권한 <u>IsAuthorized</u>API참조에서 DigitalPetStore샘플을 참조하는 예제 4의 요청 본문을 복사합니다.
- 4. API클라이언트를 열고 정책 저장소의 지역 서비스 엔드포인트에 대한 요청을 생성합니다. <u>예</u> 제에 표시된 대로 헤더를 채웁니다.
- 5. 샘플 요청 본문을 붙여넣고 의 값을 앞에서 기록해 둔 정책 저장소 policyStoreId ID로 변경합니다.
- 6. 요청을 제출하고 결과를 검토하십시오. 기본 DigitalPetStore정책 저장소에서 이 요청은 ALLOW 결정을 반환합니다.

테스트 환경에서 정책, 스키마 및 요청을 변경하여 결과를 변경하고 더 복잡한 결정을 내릴 수 있습니다.

- 1. 검증된 권한의 결정을 변경하는 방식으로 요청을 변경하십시오. 예를 들어, Alice의 역할을 로 Employee 변경하거나 주문 1234의 owner 속성을 로 변경합니다. Bob
- 2. 권한 부여 결정에 영향을 미치는 방식으로 정책을 변경하십시오. 예를 들어 고객 역할 주문 받기라는 설명으로 정책을 수정하여 의 User 소유자여야 한다는 조건을 제거하고 주문을 Bob 보려는 요청을 수정합니다. Resource
- 3. 정책에서 더 복잡한 결정을 내릴 수 있도록 스키마를 변경하십시오. Alice가 새 요구 사항을 충족할수 있도록 요청 엔티티를 업데이트하십시오. 예를 들어, ActiveUsers InactiveUsers or의 구성원이 될 수 User 있도록 스키마를 편집하십시오. 활성 사용자만 자신의 주문을 볼 수 있도록 정책을업데이트하십시오. Alice가 활성 또는 비활성 사용자가 되도록 요청 엔티티를 업데이트하십시오.

# 인증 모델을 애플리케이션과 통합

애플리케이션에서 Amazon Verified Permissions를 구현하려면 앱에 적용할 정책과 스키마를 정의해야합니다. 권한 부여 모델을 마련하고 테스트한 후, 다음 단계는 적용 시점부터 API 요청 생성을 시작하

애플리케이션과의 통합 120

는 것입니다. 이렇게 하려면 사용자 데이터를 수집하여 권한 부여 요청에 입력하도록 애플리케이션 로 직을 설정해야 합니다.

앱이 검증된 권한으로 요청을 승인하는 방법

- 1. 현재 사용자에 대한 정보를 수집하세요. 일반적으로 사용자 세부 정보는 인증된 세션의 세부 정보 (예: JWT 또는 웹 세션 쿠키) 에 제공됩니다. 이 사용자 데이터는 정책 스토어에 연결된 Amazon Cognito 자격 증명 소스 또는 다른 OpenID Connect OIDC () 공급자로부터 생성될 수 있습니다.
- 2. 사용자가 액세스하려는 리소스에 대한 정보를 수집하십시오. 일반적으로 애플리케이션은 사용자가 새 자산을 로드해야 하는 항목을 선택하면 리소스에 대한 정보를 받게 됩니다.
- 3. 사용자가 취하고자 하는 조치를 결정하세요.
- 4. 사용자가 시도한 작업에 대한 주체, 작업, 리소스 및 엔티티를 포함하여 Verified Permissions에 대한 권한 부여 요청을 생성합니다. 검증된 권한은 정책 저장소의 정책을 기준으로 요청을 평가하여 권한 부여 결정을 반환합니다.
- 5. 애플리케이션은 Verified Permissions의 허용 또는 거부 응답을 읽고 사용자 요청에 대한 결정을 적용합니다.

검증된 권한 API 작업이 내장되어 있습니다. AWS SDKs 앱에 검증된 권한을 포함하려면 선택한 언어의 AWS SDK 를 앱 패키지에 통합하세요.

자세히 알아보고 AWS SDKs 다운로드하려면 <u>도구 용을</u> 참조하십시오 Amazon Web Services.

다음은 다양한 버전의 검증된 권한 리소스에 대한 설명서로 연결되는 AWS SDKs 링크입니다.

- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java
- AWS SDK for JavaScript
- AWS SDK for PHP
- AWS SDK for Python (Boto)
- AWS SDK for Ruby

다음 AWS SDK for JavaScript 예시는 Amazon <u>검증 권한 및 IsAuthorized Amazon Cognito를 사용</u>한 세분화된 권한 부여를 단순화하여 만든 것입니다.

애플리케이션과의 통합 121

```
const authResult = await avp.isAuthorized({
    principal: 'User::"alice"',
    action: 'Action::"view"',
    resource: 'Photo::"VacationPhoto94.jpg"',
    // whenever our policy references attributes of the entity,
    // isAuthorized needs an entity argument that provides
    // those attributes
    entities: {
       entityList: [
         {
            "identifier": {
                "entityType": "User",
                "entityId": "alice"
            },
            "attributes": {
                "location": {
                    "String": "USA"
            }
         }
       ]
    }
});
```

#### 더 많은 개발자 리소스

- Amazon 검증 권한 워크숍
- Amazon 검증 권한 리소스
- 에 <u>대한 사용자 지정 권한 부여 정책 공급자를 ASP 구현하십시오. NETAmazon 검증 권한을 사용하</u>는 핵심 앱
- Amazon 검증 권한을 사용하여 비즈니스 애플리케이션을 위한 권한 부여 서비스를 구축하십시오.
- Amazon 검증 권한 및 Amazon Cognito를 사용하여 세분화된 권한 부여를 간소화합니다.

애플리케이션과의 통합 122

# 컨텍스트 추가

컨텍스트는 정책 결정과 관련된 정보이지만 주체, 행동 또는 리소스의 정체성에는 포함되지 않습니다. 소스 IP 주소 집합에서 또는 사용자가 MFA로 로그인한 경우에만 작업을 허용할 수 있습니다. 애플리케이션은 이 컨텍스트 세션 데이터에 액세스할 수 있으므로 권한 부여 요청에 해당 데이터를 채워야 합니다. 검증된 권한 승인 요청의 컨텍스트 데이터는 요소에서 JSON 형식이어야 합니다. contextMap

<u>이 콘텐츠를 설명하는 예는 샘플 정책 저장소에서 가져온 것입니다.</u> 따라하려면 테스트 환경에 DigitalPetStore샘플 정책 저장소를 생성하세요.

다음 컨텍스트 객체는 샘플 DigitalPetStore정책 저장소를 기반으로 응용 프로그램의 각 Cedar 데이터 유형 중 하나를 선언합니다.

```
"context": {
   "contextMap": {
      "MfaAuthorized": {
         "boolean": true
      },
      "AccountCodes": {
         "set": [
            {
               "long": 111122223333
            },
               "long": 444455556666
            },
               "long": 123456789012
            }
         ]
      },
      "UserAgent": {
         "string": "My UserAgent 1.12"
      },
      "RequestedOrderCount": {
         "long": 4
      },
      "NetworkInfo": {
         "record": {
            "IPAddress": {
               "string": "192.0.2.178"
```

```
},
             "Country": {
                "string": "United States of America"
            },
             "SSL": {
                "boolean": true
            }
         }
      },
      "approvedBy": {
         "entityIdentifier": {
             "entityId": "Bob",
             "entityType": "DigitalPetStore::User"
         }
      }
   }
}
```

#### 권한 부여 컨텍스트의 데이터 유형

#### Boolean

바이너리 true 또는 false 값. 이 예제에서 for의 true 부울 값은 고객이 주문 보기를 요청하기 전에 다단계 인증을 MfaAuthenticated 수행했음을 나타냅니다.

#### 설정

컨텍스트 요소 모음입니다. 집합 멤버는 이 예제에서처럼 모두 같은 유형일 수도 있고, 중첩된 집합을 포함하여 형식이 서로 다를 수도 있습니다. 이 예시에서는 고객이 서로 다른 세 개의 계정과 연결되어 있습니다.

#### String

문자로 묶인 " 일련의 문자, 숫자 또는 기호. 예제에서 UserAgent 문자열은 고객이 주문 보기를 요청하는 데 사용한 브라우저를 나타냅니다.

#### Long

정수. 이 예에서 는 이 요청이 고객이 과거 주문 4건을 보도록 요청하여 발생한 일괄 처리의 RequestedOrderCount 일부임을 나타냅니다.

#### 레코드

속성 컬렉션. 요청 컨텍스트에서 이러한 속성을 선언해야 합니다. 스키마가 있는 정책 저장소에는 이 엔티티와 스키마의 엔티티 속성이 포함되어야 합니다. 이 예제에서 NetworkInfo 레코드에는

사용자의 원본 IP, 클라이언트가 결정한 해당 IP의 지리적 위치, 전송 중 암호화에 대한 정보가 포함됩니다.

#### EntityIdentifier

요청 entities 요소에 선언된 엔티티 및 속성에 대한 참조. 이 예시에서는 직원이 사용자의 주문을 승인했습니다Bob.

예제 DigitalPetStore앱에서 이 예제 컨텍스트를 테스트하려면 Customer Role - Get Order 설명과 함께 요청entities, 정책 저장소 스키마, 정적 정책을 업데이트해야 합니다.

# 권한 부여 DigitalPetStore 컨텍스트를 수락하도록 수정

처음에는 DigitalPetStore매우 복잡한 정책 저장소가 아닙니다. 앞서 제시한 컨텍스트를 지원하는 사전 구성된 정책이나 컨텍스트 속성은 포함되어 있지 않습니다. 이 컨텍스트 정보를 사용하여 권한 부여 요 청 예제를 평가하려면 정책 저장소와 권한 부여 요청을 다음과 같이 수정하십시오.

#### Schema

새 컨텍스트 속성을 지원하려면 다음 업데이트를 정책 저장소 스키마에 적용하십시오. 다음과 Get0rder actions 같이 업데이트하십시오.

```
"GetOrder": {
   "memberOf": [],
   "appliesTo": {
      "resourceTypes": [
         "Order"
      ],
      "context": {
         "type": "Record",
         "attributes": {
            "UserAgent": {
               "required": true,
               "type": "String"
            },
            "approvedBy": {
               "name": "User",
               "required": true,
               "type": "Entity"
            },
            "AccountCodes": {
```

에제 컨텍스트 평가 125

```
"type": "Set",
               "required": true,
               "element": {
                   "type": "Long"
               }
            },
            "RequestedOrderCount": {
               "type": "Long",
               "required": true
            },
            "MfaAuthorized": {
               "type": "Boolean",
               "required": true
            }
         }
      },
      "principalTypes": [
         "User"
      ]
   }
}
```

요청 NetworkInfo 컨텍스트에서 이름이 지정된 record 데이터 유형을 참조하려면 다음과 같이 스키마에  $\frac{CommonType}{CommonType}$  구문을 만드십시오.  $\frac{CommonType}{CommonType}$  구문을 만드십시오.  $\frac{CommonType}{CommonType}$  구문을 만드십시오.  $\frac{CommonType}{CommonType}$  구문을 만드십시오.

### Note

검증된 권한 시각적 스키마 편집기는 현재 commonType 구문을 지원하지 않습니다. 스키마에 추가하면 더 이상 비주얼 모드에서 스키마를 볼 수 없습니다.

```
"commonTypes": {
    "NetworkInfo": {
        "attributes": {
            "IPAddress": {
                "type": "String",
                "required": true
        },
        "SSL": {
                "required": true,
                "type": "Boolean"
```

#### **Policy**

다음 정책은 제공된 각 컨텍스트 요소가 충족해야 하는 조건을 설정합니다. 이 정책은 고객 역할 -주문 받기라는 설명과 함께 기존 정적 정책을 기반으로 합니다. 이 정책에서는 처음에는 요청을 보 내는 보안 주체가 리소스 소유자만 지정할 것을 요구합니다.

```
permit (
    principal in DigitalPetStore::Role::"Customer",
    action in [DigitalPetStore::Action::"GetOrder"],
    resource
) when {
    principal == resource.owner &&
        context.MfaAuthorized == true &&
        context.UserAgent like "*My UserAgent*" &&
        context.RequestedOrderCount <= 4 &&
        context.AccountCodes.contains(111122223333) &&
        context.NetworkInfo.Country like "*United States*" &&
        context.NetworkInfo.SSL == true &&
        context.NetworkInfo.IPAddress like "192.0.2.*" &&
        context.approvedBy in DigitalPetStore::Role::"Employee"
};</pre>
```

이제 주문 검색 요청은 요청에 추가한 추가 컨텍스트 조건을 충족해야 합니다.

- 1. 사용자는 MFA로 로그인해야 합니다.
- 2. 사용자의 웹 브라우저에는 문자열이 User-Agent My UserAgent 포함되어야 합니다.
- 3. 사용자가 4개 이하의 주문을 보도록 요청했어야 합니다.
- 4. 사용자 계정 코드 중 하나가 맞아야 합니다111122223333.
- 5. 사용자의 IP 주소는 미국에서 생성되어야 하고, 암호화된 세션에 있어야 하며, IP 주소는 다음으로 192.0.2. 시작해야 합니다.

6. 직원이 주문을 승인했어야 합니다. 승인 요청 entities 항목에서 다음과 같은 역할을 Bob 가진 사용자를 선언합니다. Employee

#### Request body

적절한 스키마와 정책으로 정책 저장소를 구성한 후에는 이 권한 부여 요청을 Verified Permissions API 작업에 <u>IsAuthorized</u>제출할 수 있습니다. 참고로 entities 세그먼트에는 역할이 인 사용자에 대한 Bob 정의가 포함되어 Employee 있습니다.

```
{
   "principal": {
      "entityType": "DigitalPetStore::User",
      "entityId": "Alice"
   },
   "action": {
      "actionType": "DigitalPetStore::Action",
      "actionId": "GetOrder"
   },
   "resource": {
      "entityType": "DigitalPetStore::Order",
      "entityId": "1234"
  },
   "context": {
     "contextMap": {
      "MfaAuthorized": {
       "boolean": true
      },
      "UserAgent": {
      "string": "My UserAgent 1.12"
      "RequestedOrderCount":{
       "long": 4
      },
      "AccountCodes": {
       "set": 「
        {"long": 111122223333},
        {"long": 444455556666},
        {"long": 123456789012}
        1
      },
      "NetworkInfo": {
        "record": {
```

```
"IPAddress": {"string": "192.0.2.178"},
       "Country": {"string": "United States of America"},
       "SSL": {"boolean": true}
     }
   },
   "approvedBy": {
     "entityIdentifier": {
      "entityId": "Bob",
      "entityType": "DigitalPetStore::User"
     }
  }
  }
},
"entities": {
  "entityList": [
    {
      "identifier": {
         "entityType": "DigitalPetStore::User",
         "entityId": "Alice"
      },
      "attributes": {
         "memberId": {
           "string": "801b87f2-1a5c-40b3-b580-eacad506d4e6"
     },
      "parents": [
         {
         "entityType": "DigitalPetStore::Role",
         "entityId": "Customer"
         }
      ]
    },
      "identifier": {
         "entityType": "DigitalPetStore::User",
         "entityId": "Bob"
      },
      "attributes": {
         "memberId": {
           "string": "49d9b81e-735d-429c-989d-93bec0bcfd8b"
           }
      },
      "parents": [
         {
```

```
"entityType": "DigitalPetStore::Role",
            "entityId": "Employee"
         ]
       },
       {
         "identifier": {
            "entityType": "DigitalPetStore::Order",
            "entityId": "1234"
         },
         "attributes": {
            "owner": {
              "entityIdentifier": {
                "entityType": "DigitalPetStore::User",
                "entityId": "Alice"
              }
           }
         },
         "parents": []
       }
     ]
  },
   "policyStoreId": "PSEXAMPLEabcdefg111111"
}
```

# Amazon Verified Permissions의 보안

클라우드 보안이 최우선 AWS 과제입니다. AWS 고객은 가장 보안에 민감한 조직의 요구 사항을 충족하도록 구축된 데이터 센터 및 네트워크 아키텍처의 혜택을 누릴 수 있습니다.

보안은 기업과 귀사 간의 공동 책임입니다. AWS <u>공동 책임 모델</u>은 이 사항을 클라우드의 보안 및 클라우드 내 보안으로 설명합니다.

- 클라우드 보안 AWS 클라우드에서 AWS 서비스를 실행하는 인프라를 보호하는 역할을 합니다 AWS 클라우드. AWS 또한 안전하게 사용할 수 있는 서비스를 제공합니다. Amazon Verified Permission에 적용되는 규정 준수 프로그램에 대해 자세히 알아보려면 규정 준수 프로그램별 범위 내AWS 서비스 (규정 준수 프로그램별)를 참조하십시오.
- 클라우드에서의 보안 귀하의 책임은 사용하는 AWS 서비스에 따라 결정됩니다. 또한 귀하는 귀사의 데이터의 민감도, 귀사의 요구 사항, 관련 법률 및 규정을 비롯한 기타 요소에 대해서도 책임이 있습니다.

이 설명서는 Verified Permissions를 사용할 때 공동 책임 모델을 적용하는 방법을 이해하는 데 도움이됩니다. 다음 주제에서는 보안 및 규정 준수 목표를 충족하도록 Verified Permissions를 구성하는 방법을 보여줍니다. 또한 Verified Permissions 리소스를 모니터링하고 보호하는 데 도움이 되는 다른 AWS서비스를 사용하는 방법도 알아봅니다.

#### 주제

- Amazon Verified Permissions의 데이터 보호
- Amazon Verified Permissions의 Identity and Access Management
- Amazon Verified Permissions에 대한 규정 준수 확인
- Amazon Verified Permissions의 복원력

# Amazon Verified Permissions의 데이터 보호

AWS <u>공동 책임 모델</u> Amazon 검증 권한의 데이터 보호에 적용됩니다. 이 모델에서 설명하는 것처럼 AWS 는 모든 AWS 클라우드를 실행하는 글로벌 인프라를 보호할 책임이 있습니다. 사용자는 인프라에서 호스팅되는 콘텐츠를 관리해야 합니다. 이 콘텐츠에는 AWS 서비스 사용하는 보안 구성 및 관리 작업이 포함됩니다. 데이터 프라이버시에 대한 자세한 내용은 데이터 프라이버시를 참조하십시오 FAQ. 유럽의 데이터 보호에 대한 자세한 내용은 AWS 공동 책임 모델 및AWS 보안 GDPR 블로그의 블로그 게시물을 참조하십시오.

데이터 보호 131

• 데이터 보호를 위해 AWS 계정 자격 증명을 보호하고 개별 사용자에게 AWS IAM Identity Center 또는 AWS Identity and Access Management (IAM) 를 설정하는 것이 좋습니다. 이렇게 하면 개별 사용자에게 자신의 직무를 충실히 이행하는 데 필요한 권한만 부여됩니다.

- 다음과 같은 방법으로 데이터를 보호하는 것이 좋습니다.
  - 각 계정마다 다단계 인증 (MFA) 을 사용하십시오.
  - SSLTLS/를 사용하여 AWS 리소스와 통신하세요. TLS1.2가 필요합니다.
  - API를 사용하여 사용자 활동 로깅을 설정합니다 AWS CloudTrail.
  - 포함된 모든 기본 보안 제어와 함께 AWS 암호화 솔루션을 사용하십시오 AWS 서비스.
  - Amazon S3에 저장된 민감한 데이터를 검색하고 보호하는 데 도움이 되는 Amazon Macie와 같은 고급 관리형 보안 서비스를 사용하세요.
  - 명령줄 인터페이스 또는 API an을 AWS 통해 액세스할 때 FIPS 140-2개의 검증된 암호화 모듈이 필요한 경우 엔드포인트를 사용하십시오. FIPS 사용 가능한 FIPS 엔드포인트에 대한 자세한 내용은 연방 정보 처리 표준 () 140-2를 참조하십시오. FIPS
- 고객의 이메일 주소와 같은 기밀 정보나 중요한 정보는 태그나 이름 필드와 같은 자유 양식 필드에 입력하지 않는 것이 좋습니다. 여기에는 콘솔을 AWS 서비스 사용하여 확인된 권한 또는 기타 작업 을 수행하는 경우가 API 포함됩니다. AWS CLI AWS SDKs 이름에 사용되는 태그 또는 자유 형식 텍 스트 필드에 입력하는 모든 데이터는 청구 또는 진단 로그에 사용될 수 있습니다. 외부 서버에 URL a 를 제공하는 경우 해당 서버에 대한 요청을 URL 검증하기 위해 자격 증명 정보를 에 포함하지 않는 것이 좋습니다.
- 작업 이름에는 민감한 정보가 포함되지 않아야 합니다.
- 또한 엔터티(리소스 및 보안 주체)에는 항상 고유하고 변경 불가능하며 재사용이 불가능한 식별자를 사용하는 것이 좋습니다. 테스트 환경에서는 User 유형의 엔터티 이름에 jane 또는 bob 와 같은 간단한 엔터티 식별자를 사용할 수 있습니다. 하지만 프로덕션 시스템에서는 보안상의 이유로 재사용할 수 없는 고유한 값을 사용하는 것이 중요합니다. 범용 고유 식별자 () 와 같은 값을 사용하는 것이 좋습니다. UUIDs 회사를 떠나는 사용자 jane을 예로 들어 보겠습니다. 이 예에서는 나중에 다른 사람이 이 이름(jane)을 사용하도록 허용합니다. 해당 새 사용자는 여전히 User::"jane"을 참조하는 정책에서 부여하는 모든 항목에 자동으로 액세스할 수 있게 됩니다. Verified Permissions와 Cedar는 새 사용자와 이전 사용자를 구분할 수 없습니다.

이 사례는 보안 주체 및 리소스 식별자 모두에 적용됩니다. 정책에 이전 식별자가 남아 있어 의도치 않게 액세스 권한을 부여하는 일이 없도록 항상 고유성이 보장되고 재사용되지 않는 식별자를 사용하십시오.

데이터 보호 132

• Long 및 Decimal 값을 정의하기 위해 제공하는 문자열이 각 유형의 유효 범위 내에 있는지 확인하십시오. 또한 산술 연산자를 사용해도 값이 유효 범위를 벗어나지 않도록 해야 합니다. 범위를 초과하면 작업 결과 오버플로 예외가 발생합니다. 오류가 발생하는 정책은 무시됩니다. 즉, 허용 정책이예기치 않게 액세스를 허용하지 못하거나 금지 정책이 예기치 않게 액세스를 차단하지 못할 수 있습니다.

# 데이터 암호화

Amazon Verified Permissions는 정책 등의 모든 고객 데이터를 AWS 관리형 키를 사용하여 자동으로 암호화하므로 고객 관리형 키를 사용할 필요도 없고 지원되지도 않습니다.

# Amazon Verified Permissions의 Identity and Access Management

AWS Identity and Access Management (IAM) 는 AWS 서비스 이를 통해 관리자는 다음 항목에 대한 액세스를 안전하게 제어할 수 있습니다. AWS 있습니다. IAM 관리자는 Verified Permissions 리소스를 사용할 수 있는 인증 (로그인) 및 권한 부여 (권한 보유) 를 받을 수 있는 사용자를 제어합니다. IAM 는 AWS 서비스 추가 비용 없이 사용할 수 있습니다.

#### 주제

- 고객
- ID를 통한 인증
- 정책을 사용한 액세스 관리
- Amazon 검증 권한은 다음과 함께 작동하는 방식 IAM
- IAM 검증된 권한에 대한 정책
- Amazon Verified Permissions 자격 증명 기반 정책 예제
- Amazon Verified Permissions 자격 증명 및 액세스 문제 해결

# 고객

사용 방법 AWS Identity and Access Management (IAM) 는 검증된 권한에서 수행하는 작업에 따라 다릅니다.

서비스 사용자 - Verified Permissions 서비스를 사용하여 작업을 수행하는 경우 필요한 보안 인증 정보 및 권한을 관리자가 제공합니다. Verified Permissions의 추가 기능을 사용하여 작업을 수행하게 되면 추가 권한이 필요할 수 있습니다. 액세스 권한 관리 방식을 이해하면 적절한 권한을 관리자에게 요청할

데이터 암호화 133

수 있습니다. Verified Permissions의 기능에 액세스할 수 없는 경우 <u>Amazon Verified Permissions 자격</u> 증명 및 액세스 문제 해결을(를) 참조하세요.

서비스 관리자 - 회사에서 Verified Permissions 리소스를 책임지고 있는 관리자는 Verified Permissions에 대한 전체 액세스 권한을 가지고 있을 것입니다. 서비스 관리자는 서비스 사용자가 액세스해야 하는 Verified Permissions 기능과 리소스를 결정합니다. 그런 다음 요청을 다음 주소로 제출해야 합니다. IAM 관리자는 서비스 사용자의 권한을 변경할 수 있습니다. 이 페이지의 정보를 검토하여다음의 기본 개념을 이해하십시오. IAM. 회사에서 사용할 수 있는 방법에 대해 자세히 알아보려면 IAM 검증된 권한의 경우 을 참조하십시오Amazon 검증 권한은 다음과 함께 작동하는 방식 IAM.

IAM 관리자 — 귀하가 다음과 같은 경우 IAM 관리자의 경우 검증된 권한에 대한 액세스를 관리하는 정책을 작성하는 방법에 대해 자세히 알아보는 것이 좋습니다. 에서 사용할 수 있는 검증된 권한 ID 기반 정책의 예를 보려면 IAM자세한 내용은 을 참조하십시오. <u>Amazon Verified Permissions 자격 증명 기반 정책 예제</u>

# ID를 통한 인증

인증은 로그인하는 방법입니다. AWS ID 자격 증명 사용. 인증 (로그인) 을 받아야 합니다. AWS다음과 같이) AWS 계정 루트 사용자 IAM사용자로서, 또는 다음과 같이 가정함으로써 IAM 역할.

에 로그인할 수 있습니다. AWS ID 소스를 통해 제공된 자격 증명을 사용하여 페더레이션 ID로 사용할수 있습니다. AWS IAM Identity Center 페더레이션 ID의 예로는 (IAMID 센터) 사용자, 회사의 SSO 인증, Google 또는 Facebook 자격 증명이 있습니다. 페더레이션 ID로 로그인하는 경우 관리자는 이전에다음을 사용하여 ID 페더레이션을 설정했습니다. IAM 역할. 액세스할 때 AWS 페더레이션을 사용하면간접적으로 역할을 맡게 됩니다.

사용자 유형에 따라 로그인할 수 있습니다. AWS Management Console 또는 AWS 액세스 포털. 로그인에 대한 자세한 내용은 AWS로그인하는 <u>방법을 참조하십시오</u>. AWS 계정의AWS 로그인 사용자 가이드.

액세스하는 경우 AWS 프로그래밍 방식으로 AWS 자격 증명을 사용하여 요청에 암호로 서명할 수 있는 소프트웨어 개발 키트 (SDKCLI) 와 명령줄 인터페이스 () 를 제공합니다. 사용하지 않는 경우 AWS 도구를 사용하려면 직접 요청에 서명해야 합니다. 권장 방법을 사용하여 직접 요청에 서명하는 방법에 대한 자세한 내용은 서명을 참조하십시오. AWS API내 요청 IAM 사용자 가이드.

사용하는 인증 방법에 상관없이 추가 보안 정보를 제공해야 할 수도 있습니다. 예: AWS 계정의 보안을 강화하기 위해 다단계 인증 (MFA) 을 사용할 것을 권장합니다. 자세한 내용은 의 <u>다단계 인증을</u> 참조하십시오. AWS IAM Identity Center 사용 설명서 및 다단계 인증 <u>사용 () MFA AWS</u>의IAM 사용자 가이드.

ID를 통한 인증 134

### AWS 계정 루트 사용자

를 만들 때 AWS 계정모든 계정에 완전히 액세스할 수 있는 하나의 로그인 ID로 시작합니다. AWS 서비스 및 계정 내 리소스 이 ID를 다음과 같이 부릅니다. AWS 계정 루트 사용자는 계정을 만들 때 사용한 이메일 주소와 암호로 로그인하여 액세스할 수 있습니다. 일상적인 작업에 루트 사용자를 사용하지 않을 것을 강력히 권장합니다. 루트 사용자 보안 인증 정보를 보호하고 루트 사용자만 수행할 수 있는 작업을 수행하는 데 사용합니다. 루트 사용자로 로그인해야 하는 작업의 전체 목록은 에서 루트 사용자자격 증명이 필요한 작업을 참조하십시오. IAM 사용 설명서.

### 페더레이션 자격 증명

가장 좋은 방법은 관리자 액세스가 필요한 사용자를 포함한 수동 사용자가 ID 공급자와의 페더레이션을 사용하여 액세스하도록 하는 것입니다. AWS 서비스 임시 자격 증명을 사용하여

페더레이션 ID는 기업 사용자 디렉토리의 사용자, 웹 ID 제공업체, AWS Directory Service, ID 센터 디렉터리 또는 액세스하는 모든 사용자 AWS 서비스 ID 소스를 통해 제공된 자격 증명을 사용합니다. 페더레이션된 ID가 액세스하는 경우 AWS 계정역할을 맡고 역할은 임시 자격 증명을 제공합니다.

중앙 집중식 액세스 관리를 위해서는 다음을 사용하는 것이 좋습니다. AWS IAM Identity Center. IAMIdentity Center에서 사용자 및 그룹을 만들거나 자체 ID 소스의 사용자 및 그룹 집합에 연결하고 동기화하여 모든 위치에서 사용할 수 있습니다. AWS 계정 및 애플리케이션. ID 센터에 대한 자세한 내용은 IAM ID 센터란 IAM 무엇입니까? 를 참조하십시오. ... 에서 AWS IAM Identity Center 사용자 가이드.

#### IAM 사용자 및 그룹

IAM사용자는 내 정체성에 속해 있습니다. AWS 계정 이는 한 사람이나 애플리케이션에 대한 특정 권한을 가지고 있습니다. 가능하면 암호 및 액세스 키와 같은 장기 자격 증명을 가진 IAM 사용자를 만드는 대신 임시 자격 증명을 사용하는 것이 좋습니다. 하지만 특정 사용 사례에서 IAM 사용자의 장기 자격 증명이 필요한 경우에는 액세스 키를 교체하는 것이 좋습니다. 자세한 내용은 장기 자격 증명이 필요한 사용 사례에 대한 정기적인 액세스 키 교체를 참조하십시오. IAM 사용자 가이드.

IAM 그룹은 IAM 사용자 컬렉션을 지정하는 ID입니다. 사용자는 그룹으로 로그인할 수 없습니다. 그룹을 사용하여 여러 사용자의 권한을 한 번에 지정할 수 있습니다. 그룹을 사용하면 대규모 사용자 집합의 권한을 더 쉽게 관리할 수 있습니다. 예를 들어 다음과 같은 그룹을 만들 수 있습니다. IAM관리자및 해당 그룹에 관리 권한 부여 IAM 있습니다.

사용자는 역할과 다릅니다. 사용자는 한 사람 또는 애플리케이션과 고유하게 연결되지만, 역할은 해당역할이 필요한 사람이라면 누구나 수임할 수 있습니다. 사용자는 영구적인 장기 보안 인증 정보를 가지고 있지만, 역할은 임시 보안 인증만 제공합니다. 자세히 알아보려면 역할 대신 IAM 사용자를 생성하는시기를 참조하십시오. IAM 사용자 가이드.

ID를 통한 인증 135

#### IAM 역할

IAM 역할은 내 정체성입니다. AWS 계정 특정 권한이 있습니다. 사용자와 비슷하지만 특정 IAM 사용자와는 관련이 없습니다. 일시적으로 다음과 같이 가정할 수 있습니다. IAM 에서의 역할 AWS Management Console 역할을 바꿔서 말이죠. 를 호출하여 역할을 맡을 수 있습니다. AWS CLI 또는 AWS API오퍼레이션을 사용하거나 사용자 지정을 사용합니다URL. 역할 사용 방법에 대한 자세한 내용은 사용을 참조하십시오. IAM내 역할 IAM 사용자 가이드.

IAM 임시 자격 증명이 있는 역할은 다음과 같은 상황에서 유용합니다.

- 페더레이션 사용자 액세스 페더레이션 ID에 권한을 부여하려면 역할을 생성하고 해당 역할의 권한을 정의합니다. 페더레이션 ID가 인증되면 역할이 연결되고 역할에 정의된 권한이 부여됩니다. 페더레이션을 위한 역할에 대한 자세한 내용은 의 타사 ID 제공자를 위한 역할 생성을 참조하십시오. IAM 사용 설명서. IAMID 센터를 사용하는 경우 권한 집합을 구성합니다. ID가 인증된 후 액세스할 수 있는 대상을 제어하기 위해 IAM Identity Center는 권한 세트를 다음과 같은 역할과 상호 연관시킵니다. IAM. 권한 집합에 대한 자세한 내용은 의 권한 집합을 참조하십시오. AWS IAM Identity Center 사용설명서.
- 임시 IAM 사용자 권한 IAM 사용자 또는 역할은 다음을 수행할 수 있습니다. IAM 특정 작업에 대해 일시적으로 다른 권한을 부여하는 역할.
- 교차 계정 액세스 다음을 사용할 수 있습니다. IAM 다른 계정의 사용자 (신뢰할 수 있는 주체) 가 계정의 리소스에 액세스할 수 있도록 허용하는 역할입니다. 역할은 계정 간 액세스를 부여하는 기본 적인 방법입니다. 하지만 일부 경우에는 AWS 서비스역할을 프록시로 사용하는 대신 정책을 리소스에 직접 연결할 수 있습니다. 계정 간 액세스에 대한 역할과 리소스 기반 정책의 차이점을 알아보려면 방법을 참조하십시오. IAM 역할은 다음과 같은 점에서 리소스 기반 정책과 다릅니다. IAM 사용설명서.
- 에서 실행되는 애플리케이션 Amazon EC2— 다음을 사용할 수 있습니다. IAM 역할은 EC2 인스턴스에서 실행되고 다음을 생성하는 애플리케이션의 임시 자격 증명을 관리하는 데 사용됩니다. AWS CLI 또는 AWS API요청. EC2인스턴스 내에 액세스 키를 저장하는 것보다 이 방법이 더 좋습니다. 할당하려면 AWS EC2인스턴스에 역할을 부여하고 모든 애플리케이션에서 사용할 수 있게 하려면 인스턴스에 연결된 인스턴스 프로필을 생성합니다. 인스턴스 프로필에는 역할이 포함되며, 이를 통해 EC2 인스턴스에서 실행 중인 프로그램이 임시 자격 증명을 얻을 수 있습니다. 자세한 내용은 사용을 참조하십시오. IAM 에서 실행 중인 애플리케이션에 권한을 부여하는 역할 Amazon EC2의 인스턴스 IAM 사용자 가이드.

사용 여부를 알아보려면 IAM 역할 또는 IAM 사용자는 생성 <u>시기를 참조하십시오. IAM</u>내 역할 (사용자 대신) IAM 사용 설명서.

ID를 통한 인증 136

# 정책을 사용한 액세스 관리

에서 액세스를 제어합니다. AWS 정책을 생성하여 정책에 연결함으로써 AWS ID 또는 리소스. 정책은 다음의 객체입니다. AWS 이는 ID 또는 리소스와 연결될 경우 해당 권한을 정의합니다. AWS 보안 주체 (사용자, 루트 사용자 또는 역할 세션) 가 요청할 때 이러한 정책을 평가합니다. 정책에서 권한은 요청이 허용되거나 거부되는 지를 결정합니다. 대부분의 정책은 다음 위치에 저장됩니다. AWS JSON문서로. JSON정책 문서의 구조 및 내용에 대한 자세한 내용은 <u>JSON정책 개요를</u> 참조하십시오. IAM 사용설명서.

관리자는 다음을 사용할 수 있습니다. AWS JSON정책을 통해 누가 무엇에 액세스할 수 있는지 지정합니다. 즉, 어떤 보안 주체가 어떤 리소스와 어떤 조건에서 작업을 수행할 수 있는지를 지정할 수 있습니다.

기본적으로, 사용자와 역할에는 어떠한 권한도 없습니다. 사용자에게 필요한 리소스에서 작업을 수행할 수 있는 권한을 부여하려면 IAM 관리자는 다음을 생성할 수 있습니다. IAM 정책. 그러면 관리자가 다음을 추가할 수 있습니다. IAM 정책을 역할에 적용하면 사용자가 역할을 수임할 수 있습니다.

IAM 정책은 작업을 수행하는 데 사용하는 방법에 관계없이 작업에 대한 권한을 정의합니다. 예를 들어, iam:GetRole 작업을 허용하는 정책이 있다고 가정합니다. 해당 정책을 사용하는 사용자는 에서 역할 정보를 가져올 수 있습니다. AWS Management Console, AWS CLI, 또는 AWS API.

### 보안 인증 기반 정책

ID 기반 정책은 IAM 사용자, 사용자 그룹 또는 역할과 같은 ID에 연결할 수 있는 JSON 권한 정책 문서입니다. 이러한 정책은 사용자와 역할이 어떤 리소스와 어떤 조건에서 어떤 작업을 수행할 수 있는지를 제어합니다. ID 기반 정책을 만드는 방법을 알아보려면 생성을 참조하십시오. <u>IAM 의 정책</u> IAM 사용자가이드.

보안 인증 기반 정책은 인라인 정책 또는 관리형 정책으로 한층 더 분류할 수 있습니다. 인라인 정책은 단일 사용자, 그룹 또는 역할에 직접 포함됩니다. 관리형 정책은 조직의 여러 사용자, 그룹 및 역할에 연결할 수 있는 독립형 정책입니다. AWS 계정. 관리형 정책에는 다음이 포함됩니다. AWS 관리형 정책 및 고객 관리형 정책. 관리형 정책과 인라인 정책 중에서 선택하는 방법을 알아보려면 에서 <u>관리형 정</u>책과 인라인 정책 중 선택을 참조하십시오. IAM 사용 설명서.

# 리소스 기반 정책

리소스 기반 정책은 리소스에 첨부하는 JSON 정책 문서입니다. 리소스 기반 정책의 예는 다음과 같습니다. IAM 역할 신뢰 정책 및 Amazon S3 버킷 정책. 리소스 기반 정책을 지원하는 서비스에서 서비스 관리자는 이러한 정책을 사용하여 특정 리소스에 대한 액세스를 통제할 수 있습니다. 정책이 연결된 리

· 정책을 사용한 액세스 관리 137

소스의 경우 정책은 지정된 보안 주체가 해당 리소스와 어떤 조건에서 어떤 작업을 수행할 수 있는지를 정의합니다. 리소스 기반 정책에서 <u>보안 주체를 지정</u>해야 합니다. 보안 주체에는 계정, 사용자, 역할, 연 동 사용자 또는 AWS 서비스.

리소스 기반 정책은 해당 서비스에 있는 인라인 정책입니다. 사용할 수 없습니다. AWS 관리형 정책은 다음과 같습니다. IAM 리소스 기반 정책에서

### 액세스 제어 목록 () ACLs

액세스 제어 목록 (ACLs) 은 리소스에 액세스할 수 있는 권한을 가진 주체 (계정 구성원, 사용자 또는역할)를 제어합니다. ACLs정책 문서 형식을 사용하지 않지만 리소스 기반 정책과 JSON 비슷합니다.

아마존 S3, AWS WAF, VPC Amazon은 지원하는 서비스의 예입니다ACLs. 자세한 내용은 Amazon 심 플 스토리지 서비스 개발자 안내서의 액세스 제어 목록 (ACL) 개요를 참조하십시오. ACLs

#### 기타 정책 타입

AWS 일반적이지 않은 추가 정책 유형을 지원합니다. 이러한 정책 타입은 더 일반적인 정책 유형에 따라 사용자에게 부여되는 최대 권한을 설정할 수 있습니다.

- 권한 경계 권한 경계는 ID 기반 정책이 사용자에게 부여할 수 있는 최대 권한을 설정하는 고급 기능입니다. IAM 엔티티 (IAM사용자 또는 역할). 개체에 대한 권한 경계를 설정할 수 있습니다. 그 결과로 얻는 권한은 객체의 자격 증명 기반 정책과 그 권한 경계의 교집합입니다. Principal 필드에서 사용자나 역할을 지정하는 리소스 기반 정책은 권한 경계를 통해 제한되지 않습니다. 이러한 정책 중하나에 포함된 명시적 거부는 허용을 재정의합니다. 권한 경계에 대한 자세한 내용은 권한 경계를 참조하십시오. IAM내의 엔티티 IAM 사용 설명서.
- 서비스 제어 정책 (SCPs) SCPs 조직 또는 OU (조직 구성 단위) 에 대한 최대 권한을 지정하는 JSON 정책입니다. AWS Organizations. AWS Organizations 여러 개를 그룹화하고 중앙에서 관리하는 서비스입니다. AWS 계정 귀사가 소유한 것입니다. 조직의 모든 기능을 사용하도록 설정하면 일부 또는 모든 계정에 서비스 제어 정책 (SCPs)을 적용할 수 있습니다. 각 항목을 포함하여 구성원 계정의 엔티티에 대한 권한을 SCP 제한합니다. AWS 계정 루트 사용자. Organizations 및 SCPs 에 대한 자세한 내용은 의 서비스 제어 정책을 참조하십시오. AWS Organizations 사용 설명서.
- 세션 정책 세션 정책은 역할 또는 페더레이션 사용자에 대해 임시 세션을 프로그래밍 방식으로 생성할 때 파라미터로 전달하는 고급 정책입니다. 결과적으로 얻는 세션의 권한은 사용자 또는 역할의보안 인증 기반 정책의 교차와 세션 정책입니다. 또한 권한을 리소스 기반 정책에서 가져올 수도 있습니다. 이러한 정책 중 하나에 포함된 명시적 거부는 허용을 재정의합니다. 자세한 내용은 의 세션 정책을 참조하십시오. IAM 사용 설명서.

정책을 사용한 액세스 관리 138

#### 여러 정책 유형

여러 정책 유형이 요청에 적용되는 경우, 결과 권한은 이해하기가 더 복잡합니다. 방법을 알아보려면 AWS 여러 정책 유형이 관련된 경우 요청을 허용할지 여부를 결정합니다. 의 <u>정책 평가 논리를</u> 참조하십시오. IAM 사용 설명서.

### Amazon 검증 권한은 다음과 함께 작동하는 방식 IAM

사용하기 전 IAM 검증된 권한에 대한 액세스를 관리하려면 다음을 알아보세요. IAM 검증된 권한과 함께 기능을 사용할 수 있습니다.

#### IAM Amazon 검증 권한과 함께 사용할 수 있는 기능

IAM feature	Verified Permissions 지원
ID 기반 정책	예
리소스 기반 정책	아니요
<u>정책 작업</u>	예
<u>정책 리소스</u>	예
<u>정책 조건 키</u>	아니요
ACLs	아니요
ABAC(정책의 태그)	아니요
임시 보안 인증	예
보안 주체 권한	예
서비스 역할	아니요
서비스 연결 역할	아니요

권한 검증 방법 및 기타 방법을 개괄적으로 파악하려면 AWS 대부분의 서비스에서 사용할 수 있습니다. IAM 기능, 참조 AWS 다음과 함께 작동하는 서비스 IAM의IAM 사용자 가이드.

예

### Verified Permissions 자격 증명 기반 정책

보안 인증 기반 정책 지원

ID 기반 정책은 IAM 사용자, 사용자 그룹 또는 역할과 같은 ID에 연결할 수 있는 JSON 권한 정책 문서입니다. 이러한 정책은 사용자와 역할이 어떤 리소스와 어떤 조건에서 어떤 작업을 수행할 수 있는지를 제어합니다. ID 기반 정책을 만드는 방법을 알아보려면 생성을 참조하십시오. <u>IAM 의 정책</u> IAM 사용자가이드.

와 함께 IAM ID 기반 정책을 통해 허용 또는 거부된 작업 및 리소스는 물론 작업이 허용되거나 거부되는 조건을 지정할 수 있습니다. 보안 인증 기반 정책에서는 보안 주체가 연결된 사용자 또는 역할에 적용되므로 보안 주체를 지정할 수 없습니다. JSON정책에서 사용할 수 있는 모든 요소에 대해 알아보려면 다음을 참조하십시오. IAM JSON정책 요소 참조: IAM 사용 설명서.

Verified Permissions 자격 증명 기반 정책 예제

Verified Permissions 자격 증명 기반 정책의 예제를 보려면 <u>Amazon Verified Permissions 자격 증명 기</u>반 정책 예제을(를) 참조하세요.

Verified Permissions 내 리소스 기반 정책

리소스 기반 정책 지원 아니요

리소스 기반 정책은 리소스에 첨부하는 JSON 정책 문서입니다. 리소스 기반 정책의 예는 다음과 같습니다. IAM 역할 신뢰 정책 및 Amazon S3 버킷 정책. 리소스 기반 정책을 지원하는 서비스에서 서비스 관리자는 이러한 정책을 사용하여 특정 리소스에 대한 액세스를 통제할 수 있습니다. 정책이 연결된 리소스의 경우 정책은 지정된 보안 주체가 해당 리소스와 어떤 조건에서 어떤 작업을 수행할 수 있는지를 정의합니다. 리소스 기반 정책에서 보안 주체를 지정해야 합니다. 보안 주체에는 계정, 사용자, 역할, 연동 사용자 또는 AWS 서비스.

계정 간 액세스를 활성화하려면 전체 계정을 지정하거나 IAM 다른 계정의 엔티티를 리소스 기반 정책의 보안 주체로 사용합니다. 리소스 기반 정책에 크로스 계정 보안 주체를 추가하는 것은 트러스트 관계 설정의 절반밖에 되지 않는다는 것을 유념하십시오. 보안 주체와 리소스가 서로 다른 경우 AWS 계정, 그리고 IAM 신뢰할 수 있는 계정의 관리자는 주체 (사용자 또는 역할) 에게 리소스에 액세스할 수 있는 권한도 부여해야 합니다. 엔터티에 ID 기반 정책을 연결하여 권한을 부여합니다. 하지만 리소스

기반 정책이 동일 계정의 보안 주체에 액세스를 부여하는 경우, 추가 자격 증명 기반 정책이 필요하지 않습니다. 자세한 내용은 의 교차 계정 리소스 액세스를 참조하십시오. IAM의IAM 사용 설명서.

#### Verified Permissions 정책 작업

정책 작업 지원

예

관리자는 다음을 사용할 수 있습니다. AWS JSON정책을 통해 누가 무엇에 액세스할 수 있는지 지정합니다. 즉, 어떤 보안 주체가 어떤 리소스와 어떤 조건에서 작업을 수행할 수 있는지를 지정할 수 있습니다.

정책 Action 요소는 JSON 정책에서 액세스를 허용하거나 거부하는 데 사용할 수 있는 작업을 설명합니다. 일반적으로 정책 조치의 이름은 관련 조치와 동일합니다. AWS API오퍼레이션. 일치하는 작업이 없는 권한 전용 작업과 같은 몇 가지 예외가 있습니다. API 정책에서 여러 작업이 필요한 몇 가지 작업도 있습니다. 이러한 추가 작업을 일컬어 종속 작업이라고 합니다.

연결된 작업을 수행할 수 있는 권한을 부여하기 위한 정책에 작업을 포함하십시오.

Verified Permissions 작업 목록을 보려면 서비스 권한 부여 참조의 <u>Amazon Verified Permissions에서</u> 정의한 작업을 참조하세요.

Verified Permissions의 정책 작업은 작업 앞에 다음 접두사를 사용합니다.

verifiedpermissions

단일 문에서 여러 작업을 지정하려면 다음과 같이 쉼표로 구분합니다.

```
"Action": [
    "verifiedpermissions:action1",
    "verifiedpermissions:action2"
]
```

와일드카드(\*)를 사용하여 여러 작업을 지정할 수 있습니다. 예를 들어, Get라는 단어로 시작하는 모든 태스크를 지정하려면 다음 태스크를 포함합니다.

```
"Action": "verifiedpermissions:Get*"
```

Verified Permissions 자격 증명 기반 정책의 예제를 보려면 <u>Amazon Verified Permissions 자격 증명 기</u>반 정책 예제을(를) 참조하세요.

#### Verified Permissions 정책 리소스

정책 리소스 지원

예

관리자는 다음을 사용할 수 있습니다. AWS JSON정책을 통해 누가 무엇에 액세스할 수 있는지 지정합니다. 즉, 어떤 보안 주체가 어떤 리소스와 어떤 조건에서 작업을 수행할 수 있는지를 지정할 수 있습니다.

ResourceJSON정책 요소는 작업이 적용되는 하나 또는 여러 개의 객체를 지정합니다. 문장에는 Resource또는 NotResource요소가 반드시 추가되어야 합니다. <u>Amazon 리소스 이름 (ARN)</u>을 사용하여 리소스를 지정하는 것이 가장 좋습니다. 리소스 수준 권한이라고 하는 특정 리소스 유형을 지원하는 작업에 대해 이 태스크를 수행할 수 있습니다.

작업 나열과 같이 리소스 수준 권한을 지원하지 않는 작업의 경우, 와일드카드(\*)를 사용하여 해당 문이모든 리소스에 적용됨을 나타냅니다.

"Resource": "\*"

확인된 권한 리소스 유형 및 해당 ARNs 유형의 목록을 보려면 서비스 인증 참조의 <u>Amazon Verified</u> <u>Permission에서 정의한 리소스 유형을</u> 참조하십시오. 각 리소스에 어떤 작업을 지정할 수 있는지 알아 보려면 Amazon 검증 권한으로 정의된 작업을 참조하십시오. ARN

Verified Permissions 정책 조건 키

서비스별 정책 조건 키 지원

아니요

관리자는 다음을 사용할 수 있습니다. AWS JSON정책을 통해 누가 무엇에 액세스할 수 있는지 지정합니다. 즉, 어떤 보안 주체가 어떤 리소스와 어떤 조건에서 작업을 수행할 수 있는지를 지정할 수 있습니다.

Condition 요소(또는 Condition 블록)를 사용하면 정책이 발효되는 조건을 지정할 수 있습니다. Condition 요소는 옵션입니다. 같거나 작음과 같은 <u>조건 연산자</u>를 사용하여 정책의 조건을 요청의 값과 일치시키는 조건식을 생성할 수 있습니다.

명령문에 여러 Condition 요소를 지정하거나 단일 Condition 요소에 여러 키를 지정하는 경우 AWS 논리 AND 연산을 사용하여 요소를 평가합니다. 단일 조건 키에 여러 값을 지정하는 경우 AWS 논

리 OR 연산을 사용하여 조건을 평가합니다. 명문의 권한을 부여하기 전에 모든 조건을 충족해야 합니다.

조건을 지정할 때 자리 표시자 변수를 사용할 수도 있습니다. 예를 들어 리소스에 IAM 사용자 이름이 태그가 지정된 경우에만 리소스에 액세스할 수 있는 권한을 IAM 사용자에게 부여할 수 있습니다. 자세한 내용은 단원을 참조하세요.IAM 정책 요소: 내의 변수 및 태그 IAM 사용 설명서.

AWS 글로벌 조건 키 및 서비스별 조건 키를 지원합니다. 모두 보려면 AWS 글로벌 조건 키는 다음을 참조하십시오. AWS 글로벌 조건 컨텍스트 키는 IAM 사용자 가이드.

#### ACLs검증된 권한에서

지원 ACLs 아니요

액세스 제어 목록 (ACLs) 은 리소스에 액세스할 수 있는 권한을 가진 주체 (계정 구성원, 사용자 또는역할)를 제어합니다. ACLs정책 문서 형식을 사용하지 않지만 리소스 기반 정책과 JSON 비슷합니다.

#### ABAC검증된 권한 포함

지원 ABAC (정책의 태그) 아니요

속성 기반 액세스 제어 (ABAC) 는 속성을 기반으로 권한을 정의하는 권한 부여 전략입니다. In AWS, 이러한 속성을 태그라고 합니다. 에 태그를 첨부할 수 있습니다. IAM 엔티티 (사용자 또는 역할) 및 다수의 엔티티 AWS 있습니다. 의 ABAC 첫 번째 단계는 엔티티와 리소스에 태그를 지정하는 것입니다. 그런 다음 보안 주체의 태그가 액세스하려는 리소스의 태그와 일치할 때 작업을 허용하도록 ABAC 정책을 설계합니다.

ABAC빠르게 성장하는 환경에서 유용하며 정책 관리가 복잡해지는 상황에 도움이 됩니다.

태그에 근거하여 액세스를 제어하려면 aws:ResourceTag/key-name, aws:RequestTag/key-name 또는 aws:TagKeys 조건 키를 사용하여 정책의 조건 요소에 태그 정보를 제공합니다.

서비스가 모든 리소스 유형에 대해 세 가지 조건 키를 모두 지원하는 경우, 값은 서비스에 대해 예입니다. 서비스가 일부 리소스 유형에 대해서만 세 가지 조건 키를 모두 지원하는 경우, 값은 부분적입니다.

에 대한 자세한 내용은 What is ABAC? 를 참조하십시오. ABAC ... 에서 IAM 사용자 가이드. 설정 ABAC 단계가 포함된 자습서를 보려면 의 <u>속성 기반 액세스 제어 사용 (ABAC)</u> 을 참조하십시오. IAM 사용 설명서.

#### Verified Permissions에서 임시 자격 증명 사용

임시 보안 인증 지원 예

약간 AWS 서비스 임시 자격 증명을 사용하여 로그인하면 작동하지 않습니다. 다음을 포함한 추가 정보는 AWS 서비스 임시 자격 증명으로 작업하려면 다음을 참조하십시오. <u>AWS 서비스 다음과 함께 작동하는 IAM</u>의IAM 사용자 가이드.

에 로그인하면 임시 자격 증명을 사용하고 있는 것입니다. AWS Management Console 사용자 이름과 암호를 제외한 모든 방법을 사용합니다. 예를 들어, 액세스할 때 AWS 회사의 Single Sign-On (SSO) 링크를 사용하면 이 프로세스에서 자동으로 임시 자격 증명을 생성합니다. 또한 콘솔에 사용자로 로그인한 다음 역할을 전환할 때 임시 보안 인증을 자동으로 생성합니다. 역할 전환에 대한 자세한 내용은 역할 전환 (콘솔)을 참조하십시오. IAM 사용 설명서.

를 사용하여 임시 자격 증명을 수동으로 생성할 수 있습니다. AWS CLI 또는 AWS API. 그러면 해당 임시 자격 증명을 사용하여 액세스할 수 있습니다. AWS. AWS 장기 액세스 키를 사용하는 대신 임시 자격 증명을 동적으로 생성할 것을 권장합니다. 자세한 내용은 의 임시 보안 자격 증명을 참조하십시오. IAM

Verified Permissions의 서비스 간 보안 주체 권한

보안 주체 권한 지원 예

IAM사용자 또는 역할을 사용하여 작업을 수행하는 경우 AWS, 귀하는 주도자로 간주됩니다. 일부 서비스를 사용하는 경우 다른 서비스에서 다른 작업을 시작하는 작업을 수행할 수 있습니다. FAS전화를 건주체의 권한을 사용합니다. AWS 서비스, 요청과 결합 AWS 서비스 다운스트림 서비스에 요청할 수 있습니다. FAS요청은 서비스가 다른 서비스와의 상호 작용이 필요한 요청을 수신할 때만 이루어집니다. AWS 서비스 또는 완료해야 할 리소스. 이 경우 두 작업을 모두 수행할 수 있는 권한이 있어야 합니다. FAS요청 시 적용되는 정책 세부 정보는 전달 액세스 세션을 참조하십시오.

Verified Permissions 서비스 역할

서비스 역할 지원 아니요

서비스 역할은 <u>IAM 서비스가 사용자를 대신하여 작업을 수행하는 역할을</u> 맡습니다. 원래 요청 ping에 대한 IAM 관리자는 내에서 서비스 역할을 생성, 수정 및 삭제할 수 있습니다. IAM. 자세한 내용은 권한을 위임하기 위한 역할 만들기를 참조하십시오. AWS 서비스의IAM 사용 설명서.

#### Verified Permissions 서비스 연결 역할

서비스 연결 역할 지원	아니요

서비스 연결 역할은 다음과 연결된 서비스 역할 유형입니다. AWS 서비스. 서비스가 사용자를 대신하여 작업을 수행하는 역할을 맡을 수 있습니다. 서비스 연결 역할은 다음과 같습니다. AWS 계정 서비스가 소유합니다. 원래 요청 ping에 대한 IAM 관리자는 서비스 연결 역할에 대한 권한을 볼 수 있지만 편집할 수는 없습니다.

서비스 연결 역할을 만들거나 관리하는 방법에 대한 자세한 내용은 을 참조하십시오. <u>AWS 다음과 함</u> <u>께 작동하는 서비스 IAM</u>. 표에서 서비스 연결 역할 **Yes** 열의 a가 포함된 서비스를 찾으십시오. 해당 서비스에 대한 서비스 연결 역할 설명서를 보려면 Yes(네) 링크를 선택합니다.

### IAM 검증된 권한에 대한 정책

Verified Permissions는 애플리케이션 내 사용자 권한을 관리합니다. 애플리케이션이 검증된 권한을 APIs 호출하도록 하려면 AWS Management Console 사용자가 검증된 권한 정책 저장소에서 Cedar 정책을 관리할 수 있도록 허용하려면 필요한 항목을 추가해야 합니다. IAM 권한.

ID 기반 정책은 ID에 첨부할 수 있는 JSON 권한 정책 문서입니다 (예: IAM 사용자, 사용자 그룹 또는역할. 이러한 정책은 사용자와 역할이 어떤 리소스와 어떤 조건에서 어떤 작업을 수행할 수 있는지를제어합니다. ID 기반 정책을 만드는 방법을 알아보려면 생성을 참조하십시오. <u>IAM 의 정책</u> IAM 사용자가이드.

와 함께 IAM ID 기반 정책을 사용하면 허용 또는 거부된 작업 및 리소스는 물론 작업이 허용되거나 거부되는 조건 (아래 목록 참조)을 지정할 수 있습니다. 보안 인증 기반 정책에서는 보안 주체가 연결된 사용자 또는 역할에 적용되므로 보안 주체를 지정할 수 없습니다. JSON정책에서 사용할 수 있는 모든 요소에 대해 알아보려면 다음을 참조하십시오. IAM JSON정책 요소 참조: IAM 사용 설명서.

작업	설명
CreatePolicyStore	새 정책 스토어를 생성하는 작업입니다.
<u>DeletePolicyStore</u>	정책 스토어를 삭제하는 작업입니다.

작업	설명
ListPolicyStores	에 있는 모든 정책 저장소를 나열하는 작업 AWS 계정.
CreatePolicy	정책 스토어에 Cedar 정책을 생성하는 작업입니다. 정적 정책을 생성하거나 정책 템플릿에 연결된 정책을 생성할 수 있습니다.
DeletePolicy	정책 스토어에서 정책을 삭제하는 작업입니다.
GetPolicy	지정된 정책에 대한 정보를 검색하는 작업입니 다.
ListPolicies	정책 스토어의 모든 정책을 나열하는 작업입니 다.
IsAuthorized	<u>권한 부여 요청</u> 에 설명된 매개 변수를 기반으로 <u>권한 부여 응답</u> 을 받기 위한 작업입니다.

### 예 IAM CreatePolicy 작업 권한 정책:

### Amazon Verified Permissions 자격 증명 기반 정책 예제

기본적으로 사용자 및 역할에는 Verified Permissions 리소스를 생성하거나 수정할 수 있는 권한이 없습니다. 또한 사용자는 다음을 사용하여 작업을 수행할 수 없습니다. AWS Management Console, AWS Command Line Interface (AWS CLI), 또는 AWS API. 원래 요청 ping에 대한 IAM 관리자가 생성

지격 증명 기반 정책 예시 146

해야 합니다. IAM 사용자 및 역할에 필요한 리소스에서 작업을 수행할 수 있는 권한을 부여하는 정책. 그런 다음 관리자는 해당 권한이 필요한 사용자에 이러한 정책을 연결해야 합니다.

생성 방법을 배우려면 IAM 이 예제 정책 문서를 사용하여 ID 기반 JSON 정책을 만들려면 생성을 참조하십시오. IAM 의 정책 IAM 사용자 가이드.

각 리소스 유형의 형식을 비롯하여 검증된 권한으로 정의된 작업 및 리소스 유형에 ARNs 대한 자세한 내용은 서비스 인증 참조의 <u>Amazon Verified Permissions에 대한 작업, 리소스 및 조건 키를</u> 참조하십시오.

#### 주제

- 정책 모범 사례
- Verified Permissions 콘솔 사용
- 사용자가 자신의 고유한 권한을 볼 수 있도록 허용

#### 정책 모범 사례

자격 증명 기반 정책에 따라 계정에서 사용자가 Verified Permissions 리소스를 생성, 액세스 또는 삭제할 수 있는지 여부가 결정됩니다. 이러한 작업으로 인해 비용이 발생할 수 있습니다. AWS 계정. ID 기반 정책을 만들거나 편집할 때는 다음 지침 및 권장 사항을 따르십시오.

- 시작해 보세요. AWS 관리형 정책 및 최소 권한 권한으로의 이동 사용자와 워크로드에 권한 부여를 시작하려면 다음을 사용하십시오. AWS 여러 일반 사용 사례에 권한을 부여하는 관리형 정책. 다음 사이트에서 사용할 수 있습니다. AWS 계정. 를 정의하여 권한을 더 줄이는 것이 좋습니다. AWS 사용 사례에 맞는 고객 관리형 정책. 자세한 내용은 단원을 참조하세요.AWS 관리형 정책 또는 AWS 의 직무 기능에 대한 관리형 정책 IAM 사용 설명서.
- 최소 권한 적용 다음과 같이 권한을 설정하는 경우 IAM 정책, 작업 수행에 필요한 권한만 부여하십시오. 이렇게 하려면 최소 권한으로 알려진 특정 조건에서 특정 리소스에 대해 수행할 수 있는 작업을 정의합니다. 사용에 대한 자세한 내용은 IAM 권한을 적용하려면 의 정책 및 권한을 참조하십시오. IAM의shared id="IAM"/> 사용 설명서.
- 사용 조건 IAM 정책을 통한 액세스 추가 제한 정책에 조건을 추가하여 작업 및 리소스에 대한 액세스를 제한할 수 있습니다. 예를 들어, 를 사용하여 모든 요청을 전송하도록 지정하는 정책 조건을 작성할 수 SSL 있습니다. 특정 작업을 통해 서비스 작업을 사용하는 경우 조건을 사용하여 서비스 작업에 대한 액세스 권한을 부여할 수도 있습니다. AWS 서비스예: AWS CloudFormation. 자세한 내용은 을 참조하십시오. IAM JSON정책 요소: 조건 IAM 사용 설명서.
- IAM액세스 분석기를 사용하여 유효성을 검사하십시오. IAM 보안 및 기능적 권한 보장을 위한 정책
   IAM Access Analyzer는 새 정책과 기존 정책을 검증하여 정책이 다음 규정을 준수하는지 확인합

-자격 증명 기반 정책 예시 147

니다. IAM 정책 언어 (JSON) 및 IAM 모범 사례. IAMAccess Analyzer는 안전하고 기능적인 정책을 작성하는 데 도움이 되는 100개 이상의 정책 검사 및 실행 가능한 권장 사항을 제공합니다. 자세한 내용은 의 IAM액세스 분석기 정책 검증을 참조하십시오. IAM 사용 설명서.

• 다단계 인증 필요 (MFA) - 사용자 또는 루트 IAM 사용자가 필요한 시나리오가 있는 경우 AWS 계정 보안을 강화하려면 MFA 켜십시오. API작업 호출 MFA 시기를 요구하려면 정책에 MFA 조건을 추가 하십시오. 자세한 내용은 의 MFA-protected API 액세스 구성을 참조하십시오. IAM 사용 설명서.

모범 사례에 대한 자세한 내용은 IAM의 보안 모범 사례를 참조하십시오. IAM의IAM 사용 설명서.

#### Verified Permissions 콘솔 사용

Amazon Verified Permissions 콘솔에 액세스하려면 최소한의 권한 집합이 있어야 합니다. 이러한 권한을 통해 자신의 검증된 권한 리소스에 대한 세부 정보를 나열하고 볼 수 있어야 합니다. AWS 계정. 필요한 최소 권한보다 더 제한적인 ID 기반 정책을 만들면 해당 정책을 사용하는 엔티티 (사용자 또는 역할)에 대해 콘솔이 의도한 대로 작동하지 않습니다.

전화만 거는 사용자에게는 최소 콘솔 권한을 허용할 필요가 없습니다. AWS CLI 또는 AWS API. 대신수행하려는 작업과 일치하는 API 작업에만 액세스를 허용하세요.

사용자와 역할이 검증된 권한 콘솔을 계속 사용할 수 있도록 하려면 검증된 권한도 *ConsoleAccess* 연결하거나 *ReadOnly* AWS 엔티티에 대한 관리형 정책. 자세한 내용은 의 <u>사용자에게 권한 추가를</u> 참 조하십시오. IAM 사용 설명서.

#### 사용자가 자신의 고유한 권한을 볼 수 있도록 허용

이 예제에서는 IAM 사용자가 자신의 사용자 ID에 연결된 인라인 및 관리형 정책을 볼 수 있도록 허용하는 정책을 만드는 방법을 보여줍니다. 이 정책에는 콘솔에서 또는 다음을 사용하여 프로그래밍 방식으로 이 작업을 완료할 수 있는 권한이 포함됩니다. AWS CLI 또는 AWS API.

자격 증명 기반 정책 예시 148

```
"iam:ListUserPolicies",
                "iam:GetUser"
            ٦,
            "Resource": ["arn:aws:iam::*:user/${aws:username}"]
        },
            "Sid": "NavigateInConsole",
            "Effect": "Allow",
            "Action": [
                "iam:GetGroupPolicy",
                "iam:GetPolicyVersion",
                "iam:GetPolicy",
                "iam:ListAttachedGroupPolicies",
                "iam:ListGroupPolicies",
                "iam:ListPolicyVersions",
                "iam:ListPolicies",
                "iam:ListUsers"
            ],
            "Resource": "*"
        }
    ]
}
```

### Amazon Verified Permissions 자격 증명 및 액세스 문제 해결

다음 정보를 사용하면 검증된 권한으로 작업할 때 발생할 수 있는 일반적인 문제를 진단하고 해결하는데 도움이 됩니다. IAM.

#### 주제

- Verified Permissions에서 작업을 수행할 권한이 없음
- 저는 iam을 수행할 권한이 없습니다. PassRole
- 제 집 밖에 있는 사람들을 허용하고 싶어요 AWS 계정 내 확인된 권한 리소스에 액세스하려면

### Verified Permissions에서 작업을 수행할 권한이 없음

작업을 수행할 수 있는 권한이 없다는 오류가 수신되면 작업을 수행할 수 있도록 정책을 업데이트해야합니다.

-문제 해결 149

다음 예제 오류는 mateojackson IAM 사용자가 콘솔을 사용하여 가상 my-example-widget 리소스에 대한 세부 정보를 보려고 하지만 가상 권한이 없는 경우 발생합니다.

verifiedpermissions: GetWidget

User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:

verifiedpermissions: GetWidget on resource: my-example-widget

이 경우 verifiedpermissions: *GetWidget*작업을 사용하여 *my-example-widget*리소스에 액세 스할 수 있도록 mateojackson사용자 정책을 업데이트해야 합니다.

도움이 필요한 경우 다음 연락처로 문의하십시오. AWS 관리자에게 관리자는 로그인 자격 증명을 제공한 사람입니다.

저는 iam을 수행할 권한이 없습니다. PassRole

iam: PassRole 작업을 수행할 수 있는 권한이 없다는 오류가 수신되면 Verified Permissions에 역할을 전달할 수 있도록 정책을 업데이트해야 합니다.

약간 AWS 서비스 새 서비스 역할 또는 서비스 연결 역할을 만드는 대신 기존 역할을 해당 서비스에 전달할 수 있습니다. 이렇게 하려면 사용자가 서비스에 역할을 전달할 수 있는 권한을 가지고 있어야 합니다.

다음 예제 오류는 라는 IAM 사용자가 콘솔을 사용하여 확인된 권한에서 작업을 marymajor 수행하려고 할 때 발생합니다. 하지만 작업을 수행하려면 서비스 역할이 부여한 권한이 서비스에 있어야 합니다. Mary는 서비스에 역할을 전달할 수 있는 권한을 가지고 있지 않습니다.

User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole

이 경우, Mary가 iam: PassRole 작업을 수행할 수 있도록 Mary의 정책을 업데이트해야 합니다.

도움이 필요한 경우 다음 연락처로 문의하십시오. AWS 관리자에게 관리자는 로그인 자격 증명을 제공한 사람입니다.

제 집 밖에 있는 사람들을 허용하고 싶어요 AWS 계정 내 확인된 권한 리소스에 액세스 하려면

다른 계정의 사용자 또는 조직 외부의 사람이 리소스에 액세스할 때 사용할 수 있는 역할을 생성할 수 있습니다. 역할을 수임할 신뢰할 수 있는 사람을 지정할 수 있습니다. 리소스 기반 정책 또는 액세스 제어 목록 (ACLs) 을 지원하는 서비스의 경우 해당 정책을 사용하여 사용자에게 리소스에 대한 액세스 권한을 부여할 수 있습니다.

-문제 해결 150

사용자 가이드 Amazon Verified Permissions

자세히 알아보려면 다음을 참조하세요.

• Verified Permissions에서 이러한 기능을 지원하는지 여부를 알아보려면 Amazon 검증 권한은 다음 과 함께 작동하는 방식 IAM을(를) 참조하세요.

- 전 세계의 리소스에 대한 액세스를 제공하는 방법을 알아보려면 AWS 계정 소유한 사용자는 다른 IAM 사용자에게 액세스 권한 제공을 참조하십시오. AWS 계정 자신이 소유하고 있는 IAM 사용자 가이드.
- 리소스에 대한 액세스 권한을 제3자에게 제공하는 방법을 알아보려면 AWS 계정액세스 제공을 참 조하십시오. AWS 계정 제3자가 소유한 IAM 사용 설명서.
- ID 페더레이션을 통해 액세스를 제공하는 방법을 알아보려면 외부 인증된 사용자에게 액세스 제공 (ID 페더레이션) 을 참조하십시오. IAM 사용 설명서.
- 계정 간 액세스에 대한 역할 사용과 리소스 기반 정책 사용의 차이점을 알아보려면 의 교차 계정 리 소스 액세스를 참조하십시오. IAM의IAM 사용 설명서.

### Amazon Verified Permissions에 대한 규정 준수 확인

특정 규정 준수 프로그램의 범위 내에 AWS 서비스 있는지 알아보려면AWS 서비스 규정 준수 프로그 램별 범위 내 규정 준수AWS 서비스 프로그램별 참조하여 관심 있는 규정 준수 프로그램을 선택하십시 오. 일반 정보는 AWS 규정 준수 프로그램AWS 보증 프로그램 규정AWS 참조하십시오.

를 사용하여 AWS Artifact타사 감사 보고서를 다운로드할 수 있습니다. 자세한 내용은 의 보고서 https://docs.aws.amazon.com/artifact/latest/ug/downloading-documents.html 참조하십시오 AWS Artifact.

사용 시 규정 준수 AWS 서비스 책임은 데이터의 민감도. 회사의 규정 준수 목표. 관련 법률 및 규정에 따라 결정됩니다. AWS 규정 준수에 도움이 되는 다음 리소스를 제공합니다.

- 보안 및 규정 준수 퀵 스타트 가이드 이 배포 가이드에서는 아키텍처 고려 사항을 설명하고 보안 및 규정 준수에 AWS 중점을 둔 기본 환경을 배포하기 위한 단계를 제공합니다.
- HIPAA보안 및 규정 준수를 위한 설계 Amazon Web Services 이 백서에서는 기업이 적합한 애플 리케이션을 만드는 AWS 데 사용할 수 있는 방법을 설명합니다. HIPAA

#### Note

모든 AWS 서비스 사람이 자격이 있는 것은 아닙니다. HIPAA 자세한 내용은 HIPAA적격 서 비스 참조를 참조하십시오.

규정 준수 확인 151

• AWS 규정AWS 준수 리소스 — 이 워크북 및 가이드 모음은 해당 산업 및 지역에 적용될 수 있습니다.

- AWS 고객 규정 준수 가이드 규정 준수의 관점에서 공동 책임 모델을 이해하십시오. 이 가이드에서는 보안을 유지하기 위한 모범 사례를 AWS 서비스 요약하고 여러 프레임워크 (국립 표준 기술 연구소 (NIST), 결제 카드 산업 보안 표준 위원회 (), 국제 표준화 기구 ())를 포함한PCI) 전반의 보안 제어에 대한 지침을 매핑합니다. ISO
- AWS Config 개발자 안내서의 <u>규칙을 사용하여 리소스 평가</u> 이 AWS Config 서비스는 리소스 구성이 내부 관행, 업계 지침 및 규정을 얼마나 잘 준수하는지 평가합니다.
- AWS Security Hub— 이를 AWS 서비스 통해 내부 AWS보안 상태를 포괄적으로 파악할 수 있습니다. Security Hub는 보안 제어를 사용하여 AWS 리소스를 평가하고 보안 업계 표준 및 모범 사례에 대한 규정 준수를 확인합니다. 지원되는 서비스 및 제어 목록은 Security Hub 제어 참조를 참조하세요.
- Amazon GuardDuty 환경에 의심스럽고 악의적인 활동이 있는지 AWS 계정모니터링하여 워크로 드, 컨테이너 및 데이터에 대한 잠재적 위협을 AWS 서비스 탐지합니다. GuardDuty 특정 규정 준수 프레임워크에서 요구하는 침입 탐지 요구 사항을 충족하는 PCI DSS 등 다양한 규정 준수 요구 사항을 해결하는 데 도움이 될 수 있습니다.
- <u>AWS Audit Manager</u>— 이를 AWS 서비스 통해 AWS 사용량을 지속적으로 감사하여 위험을 관리하고 규정 및 업계 표준을 준수하는 방법을 단순화할 수 있습니다.

### Amazon Verified Permissions의 복원력

AWS 글로벌 인프라는 가용 영역을 중심으로 AWS 리전 구축됩니다. AWS 리전 물리적으로 분리되고 격리된 여러 가용 영역을 제공합니다. 이 가용 영역은 지연 시간이 짧고 처리량이 높으며 중복성이 높은 네트워킹으로 연결됩니다. 가용 영역을 사용하면 중단 없이 영역 간에 자동으로 장애 극복 조치가 이루어지는 애플리케이션 및 데이터베이스를 설계하고 운영할 수 있습니다. 가용 영역은 기존의 단일 또는 다중 데이터 센터 인프라보다 가용성, 내결함성, 확장성이 뛰어납니다.

Verified Permissions 정책 저장소를 생성하면 개인 AWS 리전내에서 생성되며 해당 지역의 가용 영역을 구성하는 데이터 센터 전체에 자동으로 복제됩니다. 현재 Verified Permissions는 리전 간 복제는 지원하지 않습니다.

가용 영역 AWS 리전 및 가용 영역에 대한 자세한 내용은 <u>AWS 글로벌 인프라를</u> 참조하십시오.

복원력 152

# Amazon 검증 권한 API 호출 모니터링

모니터링은 Amazon Verified Permissions 및 기타 AWS 솔루션의 안정성, 가용성 및 성능을 유지하는데 있어 중요한 부분입니다. AWS 검증된 권한을 모니터링하고, 문제 발생 시 보고하고, 적절한 경우 자동 조치를 취할 수 있는 다음 도구를 제공합니다.

• AWS CloudTrail사용자 계정에서 또는 AWS 계정을 대신하여 이루어진 API 호출 및 관련 이벤트를 캡처하고 지정한 Amazon S3 버킷으로 로그 파일을 전송합니다. 어떤 사용자와 계정이 전화를 걸었는지 AWS, 어떤 소스 IP 주소에서 호출이 이루어졌는지, 언제 호출이 발생했는지 식별할 수 있습니다. 자세한 내용은 AWS CloudTrail 사용 설명서를 참조하십시오.

로 확인된 권한을 모니터링하는 방법에 대한 자세한 내용은 CloudTrail 을 참조하십시오<u>를 사용하여</u> Amazon 검증 권한 API 호출 로깅 AWS CloudTrail.

# 를 사용하여 Amazon 검증 권한 API 호출 로깅 AWS CloudTrail

Amazon Verified Permissions는 검증된 권한에서 사용자, 역할 또는 서비스가 수행한 작업의 기록을 제공하는 AWS 서비스와 통합되어 있습니다. AWS CloudTrail CloudTrail 검증된 권한에 대한 모든 API 호출을 이벤트로 캡처합니다. 캡처된 호출에는 검증된 권한 콘솔에서의 호출 및 검증된 권한 API 작업에 대한 코드 호출이 포함됩니다. 트레일을 생성하면 검증된 권한에 대한 CloudTrail 이벤트를 포함하여 Amazon S3 버킷에 이벤트를 지속적으로 전송할 수 있습니다. 트레일을 구성하지 않아도 CloudTrail 콘솔의 이벤트 기록에서 가장 최근 이벤트를 계속 볼 수 있습니다. 에서 수집한 CloudTrail 정보를 사용하여 Verified Permissions에 대한 요청, 요청이 이루어진 IP 주소, 요청한 사람, 요청 시기 및 추가 세부 정보를 확인할 수 있습니다.

자세한 CloudTrail 내용은 AWS CloudTrail 사용 설명서를 참조하십시오.

### 검증된 권한 정보: CloudTrail

CloudTrail 계정을 만들 AWS 계정 때 활성화됩니다. 검증된 권한에서 활동이 발생하면 해당 활동이 CloudTrail 이벤트 기록의 다른 AWS 서비스 이벤트와 함께 이벤트에 기록됩니다. AWS 계정에서 최신 이벤트를 확인, 검색 및 다운로드할 수 있습니다. 자세한 내용은 이벤트 <u>기록으로 CloudTrail 이벤트 보</u>기를 참조하십시오.

검증된 권한에 대한 이벤트를 AWS 계정포함하여 내 이벤트의 진행 중인 기록을 보려면 트레일을 생성하십시오. 트레일을 사용하면 CloudTrail Amazon S3 버킷으로 로그 파일을 전송할 수 있습니다. 콘솔에서 추적을 생성하면 기본적으로 모든 AWS 리전에 추적이 적용됩니다. 트레일은 AWS 파티션

CloudTrail 로그 153

에 있는 모든 지역의 이벤트를 기록하고 지정한 Amazon S3 버킷으로 로그 파일을 전송합니다. 또한 CloudTrail 로그에서 수집된 이벤트 데이터를 추가로 분석하고 이에 따라 조치를 취하도록 다른 AWS 서비스를 구성할 수 있습니다. 자세한 내용은 다음 자료를 참조하십시오.

- 추적 생성 개요
- CloudTrail 지원되는 서비스 및 통합
- 다음에 대한 Amazon SNS 알림 구성 CloudTrail
- 여러 지역에서 CloudTrail 로그 파일 수신 및 여러 계정으로부터 CloudTrail 로그 파일 수신

모든 검증된 권한 작업은 Amazon 검증 권한 API 참조 가이드에 의해 CloudTrail 기록되고 문서화됩니다. 예를 들어, CreateIdentitySourceDeletePolicy, 및 ListPolicyStores 작업을 호출하면 CloudTrail 로그 파일에 항목이 생성됩니다.

모든 이벤트 및 로그 항목에는 요청을 생성한 사용자에 대한 정보가 들어 있습니다. 보안 인증 정보를 이용하면 다음을 쉽게 판단할 수 있습니다.

- 요청이 루트 또는 AWS Identity and Access Management (IAM) 사용자 자격 증명으로 이루어졌는지 여부.
- 역할 또는 페더레이션 사용자에 대한 임시 보안 보안 인증을 사용하여 요청이 생성되었는지 여부.
- 다른 AWS 서비스에서 요청했는지 여부.

자세한 내용은 CloudTrail userIdentity 요소를 참조하십시오.

트레일 또는 이벤트 데이터 저장소를 생성할 때 기본적으로 기록되거나 기록되지 않는 데이터 이벤트가 기록됩니다. <u>IsAuthorizedIsAuthorizedWithToken</u> CloudTrail 데이터 이벤트를 기록하려면 활동을 수집하려는 지원되는 리소스 또는 리소스 유형을 명시적으로 추가해야 합니다. 자세한 내용을 알아보려면 AWS CloudTrail 사용 설명서의 데이터 이벤트를 참조하세요.

### Verified Permissions 로그 파일 항목 이해

트레일은 지정한 Amazon S3 버킷에 이벤트를 로그 파일로 전송할 수 있는 구성입니다. CloudTrail 로그 파일에는 하나 이상의 로그 항목이 포함되어 있습니다. 이벤트는 모든 소스의 단일 요청을 나타내며 요청된 작업, 작업 날짜 및 시간, 요청 매개 변수 등에 대한 정보를 포함합니다. CloudTrail 로그 파일은 공개 API 호출의 정렬된 스택 트레이스가 아니므로 특정 순서로 표시되지 않습니다.

#### 주제

IsAuthorized

- BatchIsAuthorized
- CreatePolicyStore
- ListPolicyStores
- DeletePolicyStore
- PutSchema
- GetSchema
- CreatePolicyTemplate
- DeletePolicyTemplate
- CreatePolicy
- GetPolicy
- CreateIdentitySource
- GetIdentitySource
- ListIdentitySources
- DeleteIdentitySource

#### Note

데이터 프라이버시의 예에서 일부 필드가 수정되었습니다.

#### **IsAuthorized**

```
"eventVersion": "1.08",
    "userIdentity": {
    "type": "AssumedRole",
    "principalId": "EXAMPLE_PRINCIPAL_ID",
    "arn": "arn:aws:iam::123456789012:role/ExampleRole",
    "accountId": "123456789012",
    "accessKeyId": "AKIAIOSFODNN7EXAMPLE"
    },
    "eventTime": "2023-11-20T22:55:03Z",
    "eventSource": "verifiedpermissions.amazonaws.com",
    "eventName": "IsAuthorized",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "203.0.113.0",
```

```
"userAgent": "aws-cli/2.11.18 Python/3.11.3 Linux/5.4.241-160.348.amzn2int.x86_64
 exe/x86_64.amzn.2 prompt/off command/verifiedpermissions.is-authorized",
    "requestParameters": {
        "principal": {
            "entityType": "PhotoFlash::User",
            "entityId": "alice"
        },
        "action": {
            "actionType": "PhotoFlash::Action",
            "actionId": "ViewPhoto"
        },
        "resource": {
            "entityType": "PhotoFlash::Photo",
            "entityId": "VacationPhoto94.jpg"
        },
        "policyStoreId": "PSEXAMPLEabcdefg111111"
    },
    "responseElements": null,
    "additionalEventData": {
        "decision": "ALLOW"
    },
    "requestID": "346c4b6a-d12f-46b6-bc06-6c857bd3b28e",
    "eventID": "8a4fed32-9605-45dd-a09a-5ebbf0715bbc",
    "readOnly": true,
    "resources": [
        {
    "accountId": "123456789012",
    "type": "AWS::VerifiedPermissions::PolicyStore",
    "ARN": "arn:aws:verifiedpermissions::123456789012:policy-store/
PSEXAMPLEabcdefg111111"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": false,
    "recipientAccountId": "123456789012",
    "eventCategory": "Data"
}
```

#### **BatchIsAuthorized**

```
{
   "eventVersion": "1.08",
   "userIdentity": {
```

```
"type": "AssumedRole",
 "principalId": "EXAMPLE_PRINCIPAL_ID",
 "arn": "arn:aws:iam::123456789012:role/ExampleRole",
 "accountId": "123456789012",
 "accessKeyId": "AKIAIOSFODNN7EXAMPLE"
   },
   "eventTime": "2023-11-20T23:02:33Z",
   "eventSource": "verifiedpermissions.amazonaws.com",
   "eventName": "BatchIsAuthorized",
   "awsRegion": "us-west-2",
   "sourceIPAddress": "203.0.113.0",
   "userAgent": "aws-cli/2.11.18 Python/3.11.3 Linux/5.4.241-160.348.amzn2int.x86_64
exe/x86_64.amzn.2 prompt/off command/verifiedpermissions.is-authorized",
   "requestParameters": {
       "requests": [
           {
               "principal": {
                   "entityType": "PhotoFlash::User",
                   "entityId": "alice"
               },
               "action": {
                   "actionType": "PhotoFlash::Action",
                   "actionId": "ViewPhoto"
               },
               "resource": {
                   "entityType": "PhotoFlash::Photo",
                   "entityId": "VacationPhoto94.jpg"
               }
           },
               "principal": {
                   "entityType": "PhotoFlash::User",
                   "entityId": "annalisa"
               },
               "action": {
                   "actionType": "PhotoFlash::Action",
                   "actionId": "DeletePhoto"
               },
               "resource": {
                   "entityType": "PhotoFlash::Photo",
                   "entityId": "VacationPhoto94.jpg"
               }
           }
       ],
```

```
"policyStoreId": "PSEXAMPLEabcdefg111111"
},
"responseElements": null,
"additionalEventData": {
    "results": [
        {
            "request": {
                "principal": {
                    "entityType": "PhotoFlash::User",
                    "entityId": "alice"
                },
                "action": {
                    "actionType": "PhotoFlash::Action",
                    "actionId": "ViewPhoto"
                },
                "resource": {
                    "entityType": "PhotoFlash::Photo",
                    "entityId": "VacationPhoto94.jpg"
                }
            },
            "decision": "ALLOW"
        },
        {
            "request": {
                "principal": {
                    "entityType": "PhotoFlash::User",
                    "entityId": "annalisa"
                },
                "action": {
                    "actionType": "PhotoFlash::Action",
                    "actionId": "DeletePhoto"
                },
                "resource": {
                    "entityType": "PhotoFlash::Photo",
                    "entityId": "VacationPhoto94.jpg"
                }
            },
            "decision": "DENY"
        }
    ]
},
"requestID": "a8a5caf3-78bd-4139-924c-7101a8339c3b",
"eventID": "7d81232f-f3d1-4102-b9c9-15157c70487b",
"readOnly": true,
```

#### CreatePolicyStore

```
"eventVersion": "1.08",
  "userIdentity": {
    "type": "AssumedRole",
    "principalId": "EXAMPLE_PRINCIPAL_ID",
    "arn": "arn:aws:iam::123456789012:role/ExampleRole",
    "accountId": "123456789012",
    "accessKeyId": "AKIAIOSFODNN7EXAMPLE"
  },
  "eventTime": "2023-05-22T07:43:33Z",
  "eventSource": "verifiedpermissions.amazonaws.com",
  "eventName": "CreatePolicyStore",
  "awsRegion": "us-west-2",
  "sourceIPAddress": "203.0.113.0",
  "userAgent": "aws-sdk-rust/0.55.2 os/linux lang/rust/1.69.0",
  "requestParameters": {
    "clientToken": "a1b2c3d4-e5f6-a1b2-c3d4-TOKEN1111111",
    "validationSettings": {
      "mode": "OFF"
    }
  },
  "responseElements": {
    "policyStoreId": "PSEXAMPLEabcdefg111111",
    "arn": "arn:aws:verifiedpermissions::123456789012:policy-store/
PSEXAMPLEabcdefg111111",
    "createdDate": "2023-05-22T07:43:33.962794Z",
    "lastUpdatedDate": "2023-05-22T07:43:33.962794Z"
```

```
},
"requestID": "1dd9360e-e2dc-4554-ab65-b46d2cf45c29",
"eventID": "b6edaeee-3584-4b4e-a48e-311de46d7532",
"readOnly": false,
"eventType": "AwsApiCall",
"managementEvent": true,
"recipientAccountId": "123456789012",
"eventCategory": "Management"
}
```

### ListPolicyStores

```
{
  "eventVersion": "1.08",
  "userIdentity": {
    "type": "AssumedRole",
    "principalId": "EXAMPLE_PRINCIPAL_ID",
    "arn": "arn:aws:iam::123456789012:role/ExampleRole",
    "accountId": "123456789012",
    "accessKeyId": "AKIAIOSFODNN7EXAMPLE"
  },
  "eventTime": "2023-05-22T07:43:33Z",
  "eventSource": "verifiedpermissions.amazonaws.com",
  "eventName": "ListPolicyStores",
  "awsRegion": "us-west-2",
  "sourceIPAddress": "203.0.113.0",
  "userAgent": "aws-sdk-rust/0.55.2 os/linux lang/rust/1.69.0",
  "requestParameters": {
    "maxResults": 10
  },
  "responseElements": null,
  "requestID": "5ef238db-9f87-4f37-ab7b-6cf0ba5df891",
  "eventID": "b0430fb0-12c3-4cca-8d05-84c37f99c51f",
  "readOnly": true,
  "eventType": "AwsApiCall",
  "managementEvent": true,
  "recipientAccountId": "123456789012",
  "eventCategory": "Management"
}
```

### DeletePolicyStore

```
{
```

```
"eventVersion": "1.08",
  "userIdentity": {
    "type": "AssumedRole",
    "principalId": "EXAMPLE_PRINCIPAL_ID",
    "arn": "arn:aws:iam::123456789012:role/ExampleRole",
    "accountId": "123456789012",
    "accessKeyId": "AKIAIOSFODNN7EXAMPLE"
  },
  "eventTime": "2023-05-22T07:43:32Z",
  "eventSource": "verifiedpermissions.amazonaws.com",
  "eventName": "DeletePolicyStore",
  "awsRegion": "us-west-2",
  "sourceIPAddress": "203.0.113.0",
  "userAgent": "aws-sdk-rust/0.55.2 os/linux lang/rust/1.69.0",
  "requestParameters": {
    "policyStoreId": "PSEXAMPLEabcdefg111111"
  },
  "responseElements": null,
  "requestID": "1368e8f9-130d-45a5-b96d-99097ca3077f",
  "eventID": "ac482022-b2f6-4069-879a-dd509123d8d7",
  "readOnly": false,
  "resources": [
    {
      "accountId": "123456789012",
      "type": "AWS::VerifiedPermissions::PolicyStore",
      "arn": "arn:aws:verifiedpermissions::123456789012:policy-store/
PSEXAMPLEabcdefg111111"
    }
  ],
  "eventType": "AwsApiCall",
  "managementEvent": true,
  "recipientAccountId": "123456789012",
  "eventCategory": "Management"
}
```

#### **PutSchema**

```
{
  "eventVersion": "1.08",
  "userIdentity": {
    "type": "AssumedRole",
    "principalId": "EXAMPLE_PRINCIPAL_ID",
    "arn": "arn:aws:iam::123456789012:role/ExampleRole",
```

```
"accountId": "123456789012",
    "accessKeyId": "AKIAIOSFODNN7EXAMPLE"
 },
  "eventTime": "2023-05-16T12:58:57Z",
  "eventSource": "verifiedpermissions.amazonaws.com",
  "eventName": "PutSchema",
  "awsRegion": "us-west-2",
  "sourceIPAddress": "203.0.113.0",
  "userAgent": "aws-sdk-rust/0.55.2 os/linux lang/rust/1.69.0",
  "requestParameters": {
    "policyStoreId": "PSEXAMPLEabcdefg111111"
  },
  "responseElements": {
    "lastUpdatedDate": "2023-05-16T12:58:57.513442Z",
    "namespaces": "[some_namespace]",
    "createdDate": "2023-05-16T12:58:57.513442Z",
    "policyStoreId": "PSEXAMPLEabcdefg111111",
  },
  "requestID": "631fbfa1-a959-4988-b9f8-f1a43ff5df0d",
  "eventID": "7cd0c677-733f-4602-bc03-248bae581fe5",
  "readOnly": false,
  "resources": [
    {
      "accountId": "123456789012",
      "type": "AWS::VerifiedPermissions::PolicyStore",
      "ARN": "arn:aws:verifiedpermissions::123456789012:policy-store/
PSEXAMPLEabcdefg111111"
    }
  ],
  "eventType": "AwsApiCall",
  "managementEvent": true,
  "recipientAccountId": "123456789012",
  "eventCategory": "Management"
}
```

### GetSchema

```
{
  "eventVersion": "1.08",
  "userIdentity": {
    "type": "AssumedRole",
    "principalId": "EXAMPLE_PRINCIPAL_ID",
    "arn": "arn:aws:iam::222222222222:role/ExampleRole",
```

```
"accountId": "2222222222",
    "accessKeyId": "AKIAIOSFODNN7EXAMPLE"
  },
  "eventTime": "2023-05-25T01:12:07Z",
  "eventSource": "verifiedpermissions.amazonaws.com",
  "eventName": "GetSchema",
  "awsRegion": "us-west-2",
  "sourceIPAddress": "203.0.113.0",
  "userAgent": "aws-sdk-rust/0.55.2 os/linux lang/rust/1.69.0",
  "requestParameters": {
    "policyStoreId": "PSEXAMPLEabcdefg111111"
  },
  "responseElements": null,
  "requestID": "a1f4d4cd-6156-480a-a9b8-e85a71dcc7c2",
  "eventID": "0b3b8e3d-155c-46f3-a303-7e9e8b5f606b",
  "readOnly": true,
  "resources": [
    {
      "accountId": "2222222222",
      "type": "AWS::VerifiedPermissions::PolicyStore",
      "ARN": "arn:aws:verifiedpermissions::222222222222policy-store/
PSEXAMPLEabcdefg111111"
    }
  ],
  "eventType": "AwsApiCall",
  "managementEvent": true,
  "recipientAccountId": "22222222222",
  "eventCategory": "Management"
}
```

### CreatePolicyTemplate

```
{
  "eventVersion": "1.08",
  "userIdentity": {
    "type": "AssumedRole",
    "principalId": "EXAMPLE_PRINCIPAL_ID",
    "arn": "arn:aws:iam::123456789012:role/ExampleRole",
    "accountId": "123456789012",
    "accessKeyId": "AKIAIOSFODNN7EXAMPLE"
},
  "eventTime": "2023-05-16T13:00:24Z",
  "eventSource": "verifiedpermissions.amazonaws.com",
```

```
"eventName": "CreatePolicyTemplate",
  "awsRegion": "us-west-2",
  "sourceIPAddress": "203.0.113.0",
  "userAgent": "aws-sdk-rust/0.55.2 os/linux lang/rust/1.69.0",
  "requestParameters": {
    "policyStoreId": "PSEXAMPLEabcdefg111111"
  },
  "responseElements": {
    "lastUpdatedDate": "2023-05-16T13:00:23.444404Z",
    "createdDate": "2023-05-16T13:00:23.444404Z",
    "policyTemplateId": "PTEXAMPLEabcdefg111111",
    "policyStoreId": "PSEXAMPLEabcdefg111111",
  },
  "requestID": "73953bda-af5e-4854-afe2-7660b492a6d0",
  "eventID": "7425de77-ed84-4f91-a4b9-b669181cc57b",
  "readOnly": false,
  "resources": [
    {
      "accountId": "123456789012",
      "type": "AWS::VerifiedPermissions::PolicyStore",
      "arn": "arn:aws:verifiedpermissions::123456789012:policy-store/
PSEXAMPLEabcdefg111111"
    }
  ],
  "eventType": "AwsApiCall",
  "managementEvent": true,
  "recipientAccountId": "123456789012",
  "eventCategory": "Management"
}
```

### DeletePolicyTemplate

```
"eventVersion": "1.08",
"userIdentity": {
    "type": "AssumedRole",
    "principalId": "EXAMPLE_PRINCIPAL_ID",
    "arn": "arn:aws:iam::222222222222222:role/ExampleRole",
    "accountId": "2222222222222",
    "accessKeyId": "AKIAIOSFODNN7EXAMPLE"
},
"eventTime": "2023-05-25T01:11:48Z",
"eventSource": "verifiedpermissions.amazonaws.com",
```

```
"eventName": "DeletePolicyTemplate",
  "awsRegion": "us-west-2",
  "sourceIPAddress": "203.0.113.0",
  "userAgent": "aws-sdk-rust/0.55.2 os/linux lang/rust/1.69.0",
  "requestParameters": {
    "policyStoreId": "PSEXAMPLEabcdefg111111",
    "policyTemplateId": "PTEXAMPLEabcdefg111111"
  },
  "responseElements": null,
  "requestID": "5ff0f22e-6bbd-4b85-a400-4fb74aa05dc6",
  "eventID": "c0e0c689-369e-4e95-a9cd-8de113d47ffa",
  "readOnly": false,
  "resources": [
    {
      "accountId": "22222222222",
      "type": "AWS::VerifiedPermissions::PolicyStore",
      "ARN": "arn:aws:verifiedpermissions::22222222222:policy-store/
PSEXAMPLEabcdefg111111"
    }
  ],
  "eventType": "AwsApiCall",
  "managementEvent": true,
  "recipientAccountId": "22222222222",
  "eventCategory": "Management"
}
```

### CreatePolicy

```
"eventVersion": "1.08",
"userIdentity": {
    "type": "AssumedRole",
    "principalId": "EXAMPLE_PRINCIPAL_ID",
    "arn": "arn:aws:iam::123456789012:role/ExampleRole",
    "accountId": "123456789012",
    "accessKeyId": "AKIAIOSFODNN7EXAMPLE"
},
"eventTime": "2023-05-22T07:42:30Z",
"eventSource": "verifiedpermissions.amazonaws.com",
"eventName": "CreatePolicy",
"awsRegion": "us-west-2",
"sourceIPAddress": "203.0.113.0",
"userAgent": "aws-sdk-rust/0.55.2 os/linux lang/rust/1.69.0",
```

```
"requestParameters": {
    "clientToken": "a1b2c3d4-e5f6-a1b2-c3d4-TOKEN1111111",
    "policyStoreId": "PSEXAMPLEabcdefg111111"
  },
  "responseElements": {
    "policyStoreId": "PSEXAMPLEabcdefg111111",
    "policyId": "SPEXAMPLEabcdefg111111",
    "policyType": "STATIC",
    "principal": {
        "entityType": "PhotoApp::Role",
        "entityId": "PhotoJudge"
   },
    "resource": {
        "entityType": "PhotoApp::Application",
        "entityId": "PhotoApp"
    },
    "lastUpdatedDate": "2023-05-22T07:42:30.70852Z",
    "createdDate": "2023-05-22T07:42:30.70852Z"
  },
  "requestID": "93ffa151-3841-4960-9af6-30a7f817ef93",
  "eventID": "30ab405f-3dff-43ff-8af9-f513829e8bde",
  "readOnly": false,
  "resources": [
    {
      "accountId": "123456789012",
      "type": "AWS::VerifiedPermissions::PolicyStore",
      "arn": "arn:aws:verifiedpermissions::123456789012:policy-store/
PSEXAMPLEabcdefg111111"
    }
  ],
  "eventType": "AwsApiCall",
  "managementEvent": true,
  "recipientAccountId": "123456789012",
  "eventCategory": "Management"
}
```

### **GetPolicy**

```
"eventVersion": "1.08",
"userIdentity": {
    "type": "AssumedRole",
    "principalId": "EXAMPLE_PRINCIPAL_ID",
```

```
"arn": "arn:aws:iam::123456789012:role/ExampleRole",
    "accountId": "123456789012",
    "accessKeyId": "AKIAIOSFODNN7EXAMPLE"
  },
  "eventTime": "2023-05-22T07:43:29Z",
  "eventSource": "verifiedpermissions.amazonaws.com",
  "eventName": "GetPolicy",
  "awsRegion": "us-west-2",
  "sourceIPAddress": "203.0.113.0",
  "userAgent": "aws-sdk-rust/0.55.2 os/linux lang/rust/1.69.0",
  "requestParameters": {
    "policyStoreId": "PSEXAMPLEabcdefg111111",
    "policyId": "SPEXAMPLEabcdefg111111"
  },
  "responseElements": null,
  "requestID": "23022a9e-2f5c-4dac-b653-59e6987f2fac",
  "eventID": "9b4d5037-bafa-4d57-b197-f46af83fc684",
  "readOnly": true,
  "resources": [
    {
      "accountId": "123456789012",
      "type": "AWS::VerifiedPermissions::PolicyStore",
      "arn": "arn:aws:verifiedpermissions::123456789012:policy-store/
PSEXAMPLEabcdefg111111"
    }
  ],
  "eventType": "AwsApiCall",
  "managementEvent": true,
  "recipientAccountId": "123456789012",
  "eventCategory": "Management"
}
```

### CreateIdentitySource

```
"eventTime": "2023-05-19T01:27:44Z",
  "eventSource": "verifiedpermissions.amazonaws.com",
  "eventName": "CreateIdentitySource",
  "awsRegion": "us-west-2",
  "sourceIPAddress": "203.0.113.0",
  "userAgent": "aws-sdk-rust/0.55.2 os/linux lang/rust/1.69.0",
  "requestParameters": {
    "clientToken": "a1b2c3d4-e5f6-a1b2-c3d4-TOKEN1111111",
    "configuration": {
      "cognitoUserPoolConfiguration": {
        "userPoolArn": "arn:aws:cognito-idp:000011112222:us-east-1:userpool/us-
east-1_aaaaaaaaaa"
      }
    },
    "policyStoreId": "PSEXAMPLEabcdefg111111",
    "principalEntityType": "User"
  },
  "responseElements": {
    "createdDate": "2023-07-14T15:05:01.599534Z",
    "identitySourceId": "ISEXAMPLEabcdefg111111",
    "lastUpdatedDate": "2023-07-14T15:05:01.599534Z",
    "policyStoreId": "PSEXAMPLEabcdefg111111"
    },
  "requestID": "afcc1e67-d5a4-4a9b-a74c-cdc2f719391c",
  "eventID": "f13a41dc-4496-4517-aeb8-a389eb379860",
  "readOnly": false,
  "resources": [
      "accountId": "333333333333",
      "type": "AWS::VerifiedPermissions::PolicyStore",
      "arn": "arn:aws:verifiedpermissions::333333333333:policy-store/
PSEXAMPLEabcdefg111111"
    }
  ],
  "eventType": "AwsApiCall",
  "managementEvent": true,
  "recipientAccountId": "333333333333",
  "eventCategory": "Management"
}
```

### GetIdentitySource

```
{
```

```
"eventVersion": "1.08",
  "userIdentity": {
    "type": "AssumedRole",
    "principalId": "EXAMPLE_PRINCIPAL_ID",
    "arn": "arn:aws:iam::33333333333:role/ExampleRole",
    "accountId": "333333333333",
    "accessKeyId": "AKIAIOSFODNN7EXAMPLE"
  },
  "eventTime": "2023-05-24T19:55:31Z",
  "eventSource": "verifiedpermissions.amazonaws.com",
  "eventName": "GetIdentitySource",
  "awsRegion": "us-west-2",
  "sourceIPAddress": "203.0.113.0",
  "userAgent": "aws-sdk-rust/0.55.2 os/linux lang/rust/1.69.0",
  "requestParameters": {
    "identitySourceId": "ISEXAMPLEabcdefg111111",
    "policyStoreId": "PSEXAMPLEabcdefg111111"
  },
  "responseElements": null,
  "requestID": "7a6ecf79-c489-4516-bb57-9ded970279c9",
  "eventID": "fa158e6c-f705-4a15-a731-2cdb4bd9a427",
  "readOnly": true,
  "resources": [
    {
      "accountId": "333333333333",
      "type": "AWS::VerifiedPermissions::PolicyStore",
      "arn": "arn:aws:verifiedpermissions::333333333333:policy-store/
PSEXAMPLEabcdefg111111"
    }
  ],
  "eventType": "AwsApiCall",
  "managementEvent": true,
  "recipientAccountId": "333333333333",
  "eventCategory": "Management"
}
```

### ListIdentitySources

```
{
  "eventVersion": "1.08",
  "userIdentity": {
    "type": "AssumedRole",
    "principalId": "EXAMPLE_PRINCIPAL_ID",
```

```
"arn": "arn:aws:iam::3333333333:role/ExampleRole",
    "accountId": "333333333333",
    "accessKeyId": "AKIAIOSFODNN7EXAMPLE"
  },
  "eventTime": "2023-05-24T20:05:32Z",
  "eventSource": "verifiedpermissions.amazonaws.com",
  "eventName": "ListIdentitySources",
  "awsRegion": "us-west-2",
  "sourceIPAddress": "203.0.113.0",
  "userAgent": "aws-sdk-rust/0.55.2 os/linux lang/rust/1.69.0",
  "requestParameters": {
    "policyStoreId": "PSEXAMPLEabcdefg111111"
  },
  "responseElements": null,
  "requestID": "95d2a7bc-7e9a-4efe-918e-97e558aacaf7",
  "eventID": "d3dc53f6-1432-40c8-9d1d-b9eeb75c6193",
  "readOnly": true,
  "resources": [
   {
      "accountId": "333333333333",
      "type": "AWS::VerifiedPermissions::PolicyStore",
      "arn": "arn:aws:verifiedpermissions::333333333333:policy-store/
PSEXAMPLEabcdefg111111"
    }
  ],
  "eventType": "AwsApiCall",
  "managementEvent": true,
  "recipientAccountId": "333333333333",
  "eventCategory": "Management"
}
```

### **DeleteIdentitySource**

```
"eventSource": "verifiedpermissions.amazonaws.com",
  "eventName": "DeleteIdentitySource",
  "awsRegion": "us-west-2",
  "sourceIPAddress": "203.0.113.0",
  "userAgent": "aws-sdk-rust/0.55.2 os/linux lang/rust/1.69.0",
  "requestParameters": {
    "identitySourceId": "ISEXAMPLEabcdefg111111",
    "policyStoreId": "PSEXAMPLEabcdefg111111"
  },
  "responseElements": null,
  "requestID": "d554d964-0957-4834-a421-c417bd293086",
  "eventID": "fe4d867c-88ee-4e5d-8d30-2fbc208c9260",
  "readOnly": false,
  "resources": [
    {
      "accountId": "333333333333",
      "type": "AWS::VerifiedPermissions::PolicyStore",
      "arn": "arn:aws:verifiedpermissions::333333333333:policy-store/
PSEXAMPLEabcdefg111111"
    }
  ],
  "eventType": "AwsApiCall",
  "managementEvent": true,
  "recipientAccountId": "333333333333",
  "eventCategory": "Management"
}
```

사용자 가이드 Amazon Verified Permissions

# AWS CloudFormation으로 Amazon Verified Permissions 리 소스 생성

Amazon Verified Permissions는 리소스와 AWS CloudFormation인프라를 생성하고 관리하는 데 소요 되는 시간을 줄일 수 있도록 AWS 리소스를 모델링하고 설정하는 데 도움이 되는 서비스인 와 통합되 어 있습니다. 원하는 모든 AWS 리소스 (예: 정책 저장소) 를 설명하는 템플릿을 생성하고 해당 리소스 를 AWS CloudFormation 프로비저닝 및 구성합니다.

를 사용하면 템플릿을 재사용하여 AWS CloudFormation Verified Permissions 리소스를 일관되고 반복 적으로 설정할 수 있습니다. 리소스를 한 번 설명한 다음 여러 AWS 계정 지역과 지역에 동일한 리소스 를 반복해서 프로비저닝하세요.

#### ↑ Important

Amazon Cognito 자격 증명은 AWS 리전 아마존 검증 권한과 동일하게 모두 사용할 수 있는 것 은 아닙니다. 예를 들어 Amazon Cognito 자격 AWS CloudFormation 증명과 관련하여 오류가 발생하는 경우, Amazon Cognito Identity를 사용할 수 있는 지리적으로 가장 가까운 AWS 리 전 곳에 Amazon Cognito 사용자 풀과 클라이언트를 생성하는 것이 좋습니다. Unrecognized resource types: AWS::Cognito::UserPool, AWS::Cognito::UserPoolClient Verified Permissions 자격 증명 소스를 생성할 때는 새로 생성된 이 사용자 풀을 사용하십시오.

### 검증된 권한 및 템플릿 AWS CloudFormation

Verified Permissions 및 관련 서비스에 대한 리소스를 프로비저닝하고 구성하려면 AWS CloudFormation 템플릿을 이해해야 합니다. 템플릿은 JSON 또는 YAML 형식의 텍스트 파일입니 다. 이 템플릿은 AWS CloudFormation 스택에 프로비저닝하려는 리소스를 설명합니다. JSON또는 YAML 에 익숙하지 않은 경우 AWS CloudFormation Designer를 사용하여 템플릿을 시작하는 AWS CloudFormation 데 도움을 받을 수 있습니다. 자세한 내용은 AWS CloudFormation 디자이너란 무엇입 니까? 를 참조하십시오. AWS CloudFormation 사용 설명서에서

검증된 권한은 에서 ID 소스, 정책, 정책 저장소 및 정책 템플릿을 생성할 수 있도록 지원합니다 AWS CloudFormation. 검증된 권한 리소스의 JSON 예와 YAML 템플릿을 비롯한 자세한 내용은 사용 AWS CloudFormation 설명서의 Amazon Verified Permissions 리소스 유형 참조를 참조하십시오.

### AWS CDK 구성

코드로 클라우드 인프라를 정의하고 이를 통해 프로비저닝하기 위한 오픈 소스 소프트웨어 개발 AWS Cloud Development Kit (AWS CDK) 프레임워크입니다. AWS CloudFormation구성 또는 재사용 가능한 클라우드 구성 요소를 사용하여 템플릿을 만들 수 있습니다. AWS CloudFormation 그런 다음 이러한 템플릿을 사용하여 클라우드 인프라를 배포할 수 있습니다.

자세히 알아보고 다운로드하려면 AWS Cloud AWS CDKs Development Kit를 참조하십시오.

다음은 구문과 같은 검증된 권한 AWS CDK 리소스에 대한 설명서로 연결되는 링크입니다.

• 아마존 검증 권한 L2 구성 CDK

### 에 대해 자세히 알아보십시오. AWS CloudFormation

자세히 AWS CloudFormation알아보려면 다음 리소스를 참조하십시오.

- AWS CloudFormation
- AWS CloudFormation 사용 설명서
- AWS CloudFormation API참조
- AWS CloudFormation 명령줄 인터페이스 사용자 가이드

AWS CDK 구성 173

# 다음을 사용하여 Amazon 검증 권한에 액세스하십시오. AWS PrivateLink

를 AWS PrivateLink 사용하여 사용자 VPC 권한과 Amazon 검증 권한 간에 비공개 연결을 생성할 수있습니다. 인터넷 게이트웨이, NAT 디바이스VPC, VPN 연결 또는 AWS Direct Connect 연결을 사용하지 않고도 마치 내 계정에 있는 것처럼 검증된 권한에 액세스할 수 있습니다. 내 인스턴스는 검증된 권한에 액세스하는 데 퍼블릭 IP 주소가 VPC 필요하지 않습니다.

AWS PrivateLink에서 제공되는 인터페이스 엔드포인트를 생성하여 이 프라이빗 연결을 설정합니다. 인터페이스 엔드포인트에 대해 사용 설정하는 각 서브넷에서 엔드포인트 네트워크 인터페이스를 생성합니다. 이는 Verified Permissions로 향하는 트래픽의 진입점 역할을 하는 요청자 관리형 네트워크 인터페이스입니다.

자세한 내용은 AWS PrivateLink 가이드의 <u>AWS PrivateLink를 통해 AWS 서비스 에 액세스</u>를 참조하세요.

### Verified Permissions에 대한 고려 사항

Verified Permissions에 대한 인터페이스 엔드포인트를 설정하려면 먼저 AWS PrivateLink 가이드의  $\underline{\mathcal{I}}$  려 사항을 검토합니다.

검증된 권한은 인터페이스 엔드포인트를 통해 모든 API 작업에 대한 호출을 지원합니다.

VPC검증된 권한에는 엔드포인트 정책이 지원되지 않습니다. 기본적으로 엔드포인트를 통해 Verified Permissions에 대한 전체 액세스가 허용됩니다. 또는 보안 그룹을 엔드포인트 네트워크 인터페이스와 연결하여 인터페이스 엔드포인트를 통해 Verified Permissions로 향하는 트래픽을 제어할 수 있습니다.

## Verified Permissions에 대한 인터페이스 엔드포인트 생성

Amazon VPC 콘솔 또는 AWS Command Line Interface (AWS CLI) 를 사용하여 검증된 권한에 대한 인터페이스 엔드포인트를 생성할 수 있습니다. 자세한 내용은 AWS PrivateLink 설명서의 <u>인터페이스</u>엔드포인트 생성을 참조하십시오.

다음 서비스 이름을 사용하여 Verified Permissions에 대한 인터페이스 엔드포인트를 생성합니다.

com.amazonaws.region.verifiedpermissions

고려 사항 174

인터페이스 엔드포인트에서 DNS 프라이빗을 활성화한 경우 기본 지역 DNS 이름을 사용하여 검증된 권한을 API 요청할 수 있습니다. 예: verifiedpermissions.us-east-1.amazonaws.com.

인터페이스 엔드포인트 생성 175

# Amazon Verified Permissions 할당량

AWS 계정 Your에는 각 서비스에 대한 기본 할당량 (이전에는 한도라고 함) 이 있습니다. AWS 다르게 표시되지 않는 한 리전별로 각 할당량이 적용됩니다. 일부 할당량에 대한 증가를 요청할 수 있으며 다른 할당량은 늘릴 수 없습니다.

Verified Permissions에 대한 할당량을 보려면 <u>Service Quotas 콘솔</u>을 엽니다. 탐색 창에서 AWS 서비스를 선택하고 Verified Permissions를 선택합니다.

할당량 증가를 요청하려면 Service Quotas 사용 설명서의 <u>할당량 증가 요청</u>을 참조하십시오. Service Quotas에서 아직 할당량을 사용할 수 없는 경우 한도 증가 양식을 사용합니다.

검증된 권한과 관련된 AWS 계정 할당량은 다음과 같습니다.

#### 주제

- 리소스 할당량
- 계층 구조 할당량
- 초당 작업 할당량

# 리소스 할당량

명칭	기본값	조 정 가 능	설명
계정별 리전별 정책 스토어	각 지원되는 리전: 1,000	<u>예</u>	정책 스토어의 최대 수입 니다.
정책 스토어당 정책 템플릿	각 지원되는 리전: 40	<u>예</u>	정책 스토어의 최대 정책 템플릿 수입니다.
정책 스토어당 자격 증명 소스	1	아 니 요	정책 스토어에 대해 정의 할 수 있는 최대 자격 증명 소스 수입니다.

리소스 할당량 176

명칭	기본값	조 정 가 능	설명
권한 부여 요청 크기¹	1MB	아 니 요	권한 부여 요청의 최대 크 기입니다.
정책 크기	10,000바이트	아 니 요	개별 정책의 최대 크기입 니다.
스키마 크기	100,000바이트	아 니 요	정책 저장소 스키마의 최 대 크기.
리소스당 정책 크기	20만 바이트²	아 니 요	특정 리소스를 참조하는 모든 정책의 최대 크기입 니다.

¹ 승인 요청의 할당량은 및 의 할당량이 동일합니다. IsAuthorizedIsAuthorizedWithToken

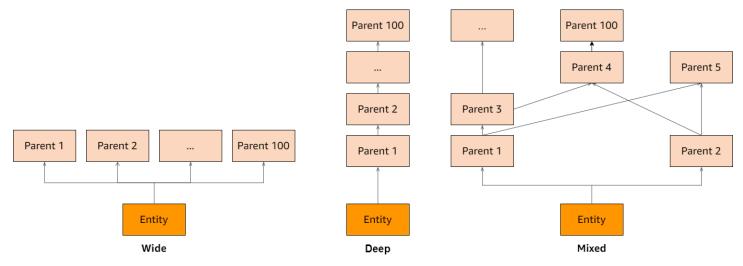
리소스 할당량 177

<sup>&</sup>lt;sup>2</sup> 단일 리소스와 관련된 모든 정책의 총 크기는 200,000바이트를 초과할 수 없습니다. 또한 "모든 리소스"를 지정하는 모든 정책의 총 크기는 200,000바이트를 초과할 수 없습니다. 템플릿 연결 정책의 경우 정책 템플릿의 크기가 한 번만 계산되며 각 템플릿 연결 정책을 인스턴스화하는 데 사용되는 각 매개 변수 집합의 크기를 더합니다.

# 계층 구조 할당량

명칭	기본값	조 정 가 능	설명
보안 주체당 전이적 상위 엔터티	100	아 니 요	각 보안 주체에 대한 최대 전이적 상위 엔터티 수입 니다.
작업당 전이적 상위 엔터티	100	아 니 요	각 작업에 대한 최대 전이 적 상위 엔터티 수입니다.
리소스당 전이적 상위 엔터티	100	아 니 요	각 리소스에 대한 최대 전 이적 상위 엔터티 수입니 다.

아래 다이어그램은 엔티티(보안 주체, 작업 또는 리소스)에 대해 전이적 상위 엔터티를 정의하는 방법을 보여줍니다.



계층 구조 할당량 178

# 초당 작업 할당량

Verified Permissions는 애플리케이션 요청이 작업 할당량을 초과하는 AWS 리전 경우 서비스 엔드포인트에 대한 요청을 제한합니다. API 검증된 권한은 초당 요청 할당량을 초과하거나 동시 쓰기 작업을 시도할 경우 예외를 반환할 수 있습니다. Service RPS Quotas에서 현재 할당량을 확인할 수 있습니다. 응용 프로그램이 작업 할당량을 초과하지 않도록 하려면 재시도 및 지수 백오프를 위해 응용 프로그램을 최적화해야 합니다. 자세한 내용은 백오프 패턴을 사용한 재시도 및 워크로드의 제한 관리 및 모니터링을 참조하십시오. API

명칭	기본값	조 정 가 능	설명
BatchIsAuthorized 계정별 지역별 초당 요청 수	각 지원되는 리전: 30	<u>예</u>	초당 최대 BatchIsAu thorized 요청 수입니다.
BatchlsAuthorizedWithToken 계정당 지역별 초당 요청 수	지원되는 각 지역: 30개	예	초당 최대 BatchIsAu thorizedWithToken 요청 수
CreatePolicy 계정당 지역별 초당 요청 수	각 지원되는 리전: 10	<u>예</u>	초당 최대 CreatePolicy 요 청 수입니다.
CreatePolicyStore 계정당 지역별 초당 요청 수	지원되는 각 리전: 1	아 니 요	초당 최대 CreatePol icyStore 요청 수
CreatePolicyTemplate 계정당 지역별 초 당 요청 수	각 지원되는 리전: 10	<u>예</u>	초당 최대 CreatePol icyTemplate 요청 수입니 다.
DeletePolicy 계정당 지역별 초당 요청 수	각 지원되는 리전: 10	<u>예</u>	초당 최대 DeletePolicy 요 청 수입니다.
DeletePolicyStore 계정당 지역별 초당 요청 수	지원되는 각 리전: 1	아 니 요	초당 최대 DeletePol icyStore 요청 수

명칭	기본값	조 정 가 능	설명
DeletePolicyTemplate 계정당 지역별 초 당 요청 수	각 지원되는 리전: 10	<u>예</u>	초당 최대 DeletePol icyTemplate 요청 수입니 다.
GetPolicy 계정당 지역별 초당 요청 수	각 지원되는 리전: 10	<u>예</u>	초당 최대 GetPolicy 요청 수입니다.
GetPolicyTemplate 계정당 지역별 초당 요청 수	각 지원되는 리전: 10	<u>예</u>	초당 최대 GetPolicy Template 요청 수입니다.
GetSchema 계정당 지역별 초당 요청 수	각 지원되는 리전: 10	<u>예</u>	초당 최대 GetSchema 요 청 수입니다.
IsAuthorized 계정당 지역별 초당 요청 수	지원되는 각 리전: 200	<u>예</u>	초당 최대 IsAuthorized 요 청 수입니다.
IsAuthorizedWithToken 계정당 지역별 초당 요청 수	지원되는 각 리전: 200	<u>예</u>	초당 최대 IsAuthori zedWithToken 요청 수입 니다.
ListPolicies 계정당 지역별 초당 요청 수	각 지원되는 리전: 10	<u>예</u>	초당 최대 ListPolicies 요 청 수입니다.
ListPolicyStores 계정당 지역별 초당 요 청 수	각 지원되는 리전: 10	<u>예</u>	초당 최대 ListPolic yStores 요청 수입니다.
ListPolicyTemplates 계정당 지역별 초당 요청 수	각 지원되는 리전: 10	<u>예</u>	초당 최대 ListPolic yTemplates 요청 수입니 다.
PutSchema 계정당 지역별 초당 요청 수	각 지원되는 리전: 10	<u>예</u>	초당 최대 PutSchema 요 청 수입니다.

초당 작업 할당량 180

명칭	기본값	조 정 가 능	설명
UpdatePolicy 계정당 지역별 초당 요청 수	각 지원되는 리전: 10	<u>예</u>	초당 최대 UpdatePolicy 요청 수입니다.
UpdatePolicyStore 계정당 지역별 초당 요청 수	지원되는 각 지역: 10개	아 니 요	초당 최대 UpdatePol icyStore 요청 수
UpdatePolicyTemplate 계정당 지역별 초 당 요청 수	각 지원되는 리전: 10	<u>예</u>	초당 최대 UpdatePol icyTemplate 요청 수입니 다.

초당 작업 할당량 181

# Amazon 검증 권한 및 Cedar 정책 언어 용어 및 개념

Amazon Verified Permissions를 사용하려면 다음 개념을 이해해야 합니다.

#### Verified Permissions 개념

- 권한 부여 모델
- 권한 부여 요청
- 권한 부여 응답
- 고려 대상 정책
- 컨텍스트 데이터
- 결정적 정책
- 엔터티 데이터
- 권한, 권한 부여 및 보안 주체
- 정책 적용
- 정책 스토어
- 충족 대상 정책
- Amazon 검증 권한과 Cedar 정책 언어 간의 차이점

#### Cedar 정책 언어 개념

- 권한 부여
- 엔터티
- 그룹 및 계층
- 네임스페이스
- 정책
- 정책 템플릿
- 스키마

### 권한 부여 모델

권한 부여 모델은 애플리케이션이 수행하는 <u>권한 부여 요청</u>의 범위와 이러한 요청을 평가하는 기준을 설명합니다. 이는 다양한 유형의 리소스, 해당 리소스에 대해 수행되는 작업, 해당 작업을 수행하는 보안 주체 유형에 따라 정의됩니다. 또한 이러한 작업이 수행되는 상황도 고려됩니다.

역할 기반 액세스 제어 (RBAC) 는 역할을 정의하고 권한 집합과 연결하는 평가 기준입니다. 이러한 역할은 하나 이상의 자격 증명에 할당할 수 있습니다. 할당된 자격 증명은 역할에 연결된 권한을 획득하게 됩니다. 역할에 연결된 권한이 변경되면 변경 내용은 역할이 할당된 모든 자격 증명에 자동으로 영향을 미칩니다. Cedar는 주도자 그룹을 사용하여 RBAC 의사 결정을 지원할 수 있습니다.

속성 기반 액세스 제어 (ABAC) 는 ID와 관련된 권한을 해당 ID의 속성에 따라 결정하는 평가 기준입니다. Cedar는 주도자의 속성을 참조하는 정책 조건을 사용하여 ABAC 의사 결정을 지원할 수 있습니다.

Cedar 정책 언어를 사용하면 속성 기반 조건을 가진 사용자 그룹에 대해 권한을 정의할 수 있으므로 단일 정책으로 두 가지를 결합할 수 있습니다. RBAC ABAC

## 권한 부여 요청

권한 부여 요청은 애플리케이션이 일련의 정책을 평가하여 보안 주체가 해당 컨텍스트에서 리소스에 대한 작업을 수행할 수 있는지 여부를 확인하기 위해 수행하는 Verified Permissions 요청입니다.

## 권한 부여 응답

권한 부여 응답은 <u>권한 부여 요청</u>에 대한 응답입니다. 여기에는 허용 또는 거부 결정과 함께 결정 정책과 같은 추가 정보가 포함됩니다. IDs

## 고려 대상 정책

고려 대상 정책은 <u>인증 요청</u>을 평가할 때 Verified Permissions이 포함하도록 선택한 전체 정책 집합입니다.

## 컨텍스트 데이터

컨텍스트 데이터는 평가할 추가 정보를 제공하는 속성 값입니다.

### 결정적 정책

결정적 정책은 <u>권한 부여 응답</u>을 결정하는 정책입니다. 예를 들어 <u>충족된 정책</u>이 두 개일 때 하나는 거부이고 다른 하나는 허용인 경우 거부 정책이 결정적 정책이 됩니다. 충족된 허용 정책이 여러 개 있고 충족된 금지 정책이 없는 경우 여러 결정적 정책이 있는 것입니다. 일치하는 정책이 없고 응답이 거부인 경우에는 결정적 정책이 없습니다.

## 엔터티 데이터

엔터티 데이터는 보안 주체, 작업, 리소스에 대한 데이터입니다. 정책 평가와 관련된 엔터티 데이터는 보안 주체 및 리소스의 엔터티 계층 구조 및 속성 값까지 포함하는 그룹 멤버십입니다.

## 권한, 권한 부여 및 보안 주체

Verified Permissions는 사용자가 빌드한 사용자 지정 애플리케이션 내에서 세분화된 권한 및 권한 부여를 관리합니다.

보안 주체는 사용자 이름 또는 시스템 ID와 같은 식별자에 연결된 자격 증명을 보유한 애플리케이션의 사용자로 사용자 또는 시스템입니다. 인증 프로세스는 보안 주체가 실제로 해당 자격 증명이 맞는지 여 부를 판단합니다.

이러한 자격 증명에는 해당 애플리케이션 내에서 해당 주체가 무엇을 할 수 있는지 결정하는 애플리케이션 권한 집합이 연결되어 있습니다. 권한 부여는 이러한 권한을 평가하여 보안 주체가 애플리케이션에서 특정 작업을 수행할 수 있는지 여부를 결정하는 프로세스입니다. 이러한 권한은 <u>정책</u>으로 표현될수 있습니다.

## 정책 적용

정책 적용은 Verified Permissions 외부에서 애플리케이션 내 평가 결정을 적용하는 프로세스입니다. Verified Permissions 평가 결과로 거부가 반환되는 경우 정책 적용을 통해 보안 주체가 리소스에 액세스하지 못하도록 할 수 있습니다.

### 정책 스토어

정책 스토어는 정책 및 템플릿의 컨테이너입니다. 각 스토어에는 스토어에 추가된 정책을 검증하는 데 사용되는 스키마가 있습니다. 기본적으로 각 애플리케이션에는 자체 정책 스토어가 있지만 여러 애플 리케이션이 단일 정책 스토어를 공유할 수 있습니다. 애플리케이션이 권한 부여 요청을 하면 해당 요청

-결정적 정책 184

을 평가하는 데 사용된 정책 스토어를 식별합니다. 정책 스토어는 일련의 정책을 분리하는 방법을 제공하므로 다중 테넌트 애플리케이션에서 각 테넌트에 대한 스키마와 정책을 포함하는 데 사용할 수 있습니다. 단일 애플리케이션은 각 테넌트에 대해 별도의 정책 스토어를 가질 수 있습니다.

권한 부여 요청을 평가할 때 Verified Permissions는 해당 요청과 관련된 정책 스토어의 일부 정책만 고려합니다. 관련성은 정책 범위에 따라 결정됩니다. 범위는 정책이 적용되는 특정 보안 주체 및 리소스, 그리고 보안 주체가 해당 리소스에서 수행할 수 있는 작업을 식별합니다. 범위를 정의하면 고려 대상 정책의 범위를 좁혀 성능을 개선하는 데 도움이 됩니다.

### 충족 대상 정책

충족 대상 정책은 권한 부여 요청의 매개변수와 일치하는 정책입니다.

## Amazon 검증 권한과 Cedar 정책 언어 간의 차이점

Amazon Verified Permissions는 Cedar 정책 언어 엔진을 사용하여 권한 부여 작업을 수행합니다. 하지만 기본 Cedar 구현과 Verified Permissions에서 확인할 수 있는 Cedar 구현 간에는 몇 가지 차이점이 있습니다. 이 항목에서는 이러한 차이점을 설명합니다.

### 네임스페이스 정의

Cedar의 Verified Permissions 구현은 기본 Cedar 구현과 다음과 같은 차이점이 있습니다.

- Verified Permissions는 정책 스토어에 정의된 스키마에서 네임스페이스를 하나만 지원합니다.
- Verified Permissions에서는 aws, amazon 또는 cedar 값을 포함하는 <u>네임스페이스</u>를 생성할 수 없습니다.

### 정책 템플릿 지원

Verified Permissions과 Cedar는 모두 범위에서 principal 및 resource에 대한 자리 표시자만 허용합니다. 하지만 Verified Permissions를 사용하려면 principal 및 resource에 제약이 있어야 합니다.

다음 정책은 Cedar에서 유효하지만 principal에 제약이 없기 때문에 Verified Permissions에서는 거부됩니다.

```
permit(principal, action == Action::"view", resource == ?resource);
```

충족 대상 정책 185

다음 두 예제에서는 principal 및 resource에 제약이 있기 때문에 Cedar와 Verified Permissions 모두에서 유효합니다.

```
permit(principal == User::"alice", action == Action::"view", resource == ?resource);
```

```
permit(principal == ?principal, action == Action::"a", resource in ?resource);
```

### 스키마 지원

검증된 권한을 사용하려면 모든 스키마 JSON 키 이름이 비어 있지 않은 문자열이어야 합니다. Cedar는 속성과 같은 몇 가지 경우에 빈 문자열을 허용합니다.

### 익스텐션 유형 지원

Verified Permissions는 정책에서 Cedar <u>익스텐션 유형</u>을 지원하지만 스키마 정의에 또는 IsAuthorized 및 IsAuthorizedWithToken 작업의 entities 매개 변수의 일부로 포함하는 것은 현재 지원하지 않습니다.

익스텐션 유형에는 고정 소수점(decimal) 및 IP 주소(ipaddr) 데이터 형식이 포함됩니다.

### 엔티티의 시더 JSON 형식

현재 Verified Permissions를 사용하려면 에 정의된 구조 (요소 배열) 를 사용하여 권한 부여 요청에서 고려할 엔티티 목록을 전달해야 합니다. EntitiesDefinitionEntityItem 검증된 권한은 현재 권한 부여 요청에서 고려할 엔티티 목록을 <u>JSONCedar</u> 형식으로 전달하는 것을 지원하지 않습니다. Verified Permissions에서 사용할 엔터티 형식을 지정하는 데 필요한 특정 요구 사항은 <u>Amazon 검증 권한의 정</u>책 내 엔티티 형식 지정을(를) 참조하세요.

### 작업 그룹 정의

Cedar 권한 부여 방법을 사용하려면 정책에 따라 권한 부여 요청을 평가할 때 고려해야 할 엔터티 목록이 필요합니다.

애플리케이션이 사용하는 작업 및 작업 그룹은 스키마에서 정의할 수 있습니다. 하지만 Cedar는 평가요청의 일부로 스키마를 포함시키지 않습니다. 대신 Cedar는 사용자가 제출한 정책 및 정책 템플릿을 검증하는 데만 스키마를 사용합니다. Cedar는 평가 요청 중에 스키마를 참조하지 않으므로 스키마에 작업 그룹을 정의한 경우에도 권한 부여 API 작업에 전달해야 하는 엔티티 목록의 일부로 작업 그룹 목록을 포함해야 합니다.

스키마 지원 186

Verified Permissions는 이 작업을 사용자 대신 수행합니다. 스키마에서 정의한 모든 작업 그룹은 IsAuthorized 또는 IsAuthorizedWithToken 작업에 매개 변수로 전달하는 엔터티 목록에 자동으로 추가됩니다.

### 길이 및 크기 제한

Verified Permissions는 스키마, 정책 및 정책 템플릿을 보관하기 위한 정책 스토어 형태의 스토리지를 지원합니다. 이러한 스토리지로 인해 Verified Permissions에는 Cedar와 관련이 없는 일부 길이 및 크기 제한이 적용됩니다.

객체	Verified Permissions 제한(바 이트 단위)	Cedar 제한
정책 크기¹	10,000개	None
인라인 정책 설명	150	Cedar에는 해당 사항 없음
정책 템플릿 크기	10,000개	None
스키마 크기	10,000개	None
엔터티 유형	200	None
정책 ID	64	None
정책 템플릿 ID	64	None
엔터티 ID	200	None
정책 스토어 ID	64	Cedar에는 해당 사항 없음

<sup>&</sup>lt;sup>1</sup> Verified Permissions의 정책 스토어당 정책 한도는 정책 스토어에서 생성된 정책의 보안 주체, 작업 및 리소스의 합산 크기를 기준으로 합니다. 단일 리소스와 관련된 모든 정책의 총 크기는 200,000바이 트를 초과할 수 없습니다. 템플릿 연결 정책의 경우 정책 템플릿의 크기가 한 번만 계산되며 각 템플릿 연결 정책을 인스턴스화하는 데 사용되는 각 매개 변수 집합의 크기를 더합니다.

길이 및 크기 제한 187

# Amazon Verified Permissions 사용 설명서에 대한 문서 기록

다음 표에서는 Verified Permissions에 대한 문서 릴리스를 소개합니다.

변경 사항	설명	날짜
OIDC ID 소스	이제 OpenID Connect (OIDC) ID 공급자로부터 사용자에게 권한을 부여할 수 있습니다.	2024년 6월 8일
<u>ID 소스 토큰을 사용한 Batch</u> <u>인증</u>	이제 BatchIsAuthorizedW ithToken 단일 API 요청으 로 Amazon Cognito 사용자 풀 의 사용자를 승인할 수 있습니 다.	2024년 4월 5일
API Gateway를 사용하여 정책 저장소 생성	이제 기존 API 및 Amazon Cognito 사용자 풀에서 정책 스 토어를 생성할 수 있습니다.	2024년 4월 1일
컨텍스트 개념 및 예제	검증된 권한을 포함한 권한 부 여 요청의 컨텍스트에 대한 정 보를 추가했습니다.	2024년 2월 1일
권한 부여 개념 및 예제	검증된 권한을 포함한 권한 부 여 요청에 대한 정보가 추가되 었습니다.	2024년 2월 1일
AWS CloudFormation 통합	검증된 권한은 에서 AWS CloudFormation ID 소스, 정책, 정책 저장소 및 정책 템플릿을 생성할 수 있도록 지원합니다.	2023년 6월 30일
<u>최초 릴리스</u>	Amazon Verified Permissions 사용 설명서의 최초 릴리스	2023년 6월 13일

기계 번역으로 제공되는 번역입니다. 제공된 번역과 원본 영어의 내용이 상충하는 경우에는 영어 버전이 우선합니다.