Add a permission의

Amazon Verified Permissions

Copyright © 2026 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon Verified Permissions: Add a permission의

Copyright © 2026 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon의 상표 및 브랜드 디자인은 Amazon 외 제품 또는 서비스와 관련하여 고객에게 혼동을 일으킬 수 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 2
Verified Permissions의 이점 2
애플리케이션 개발 가속화 2
애플리케이션 보안 강화 2
최종 사용자 기능 3
관련 서비스 3
Verified Permissions에 액세스 3
Verified Permissions 요금 5
정책 스토어 시작하기 6
사전 조건 7
에 가입 AWS 계정 7
1단계: PhotoFlash 정책 스토어 생성 7
2단계: 정책 생성 7
3단계: 정책 스토어 테스트 8
4단계: 리소스 정리 9
권한 부여 모델 설계 11
하나의 올바른 모델이란 없습니다. 12
오류 반환 12
리소스에 집중하세요. 12
다중 테넌시 고려 14
공유 정책 스토어와 테넌트별 정책 스토어 비교 15
선택 방법 16
정책 스토어 18
정책 스토어 생성 18
Rust를 사용하여 정책 스토어 생성 26
API 연결 정책 스토어 31
작동 방식 33
고려 사항 34
ABAC 추가 35
프로덕션으로 전환 36
문제 해결 39
정책 스토어 삭제 41

정책 스토어 별칭 43
정책 스토어 별칭의 속성 43
정책 스토어 별칭 생성 46
정책 스토어 별칭 검색 47
정책 스토어 별칭 가져오기 47
정책 스토어 별칭 나열 48
정책 스토어 별칭 삭제 49
정책 스토어 별칭 소프트 삭제 50
정책 스토어 별칭 하드 삭제 50
정책 스토어 별칭 사용 51
작업에서 정책 스토어 별칭 사용 51
에서 정책 스토어 별칭 사용 AWS 리전 52
정책 스토어 별칭은 트래픽 제어 메커니즘이 아닙니다. 53
가동 중지 없는 정책 스토어 변경 수행 54
액세스 제어 55
verifiedpermissions:CreatePolicyStoreAlias 55
verifiedpermissions:GetPolicyStoreAlias 56
verifiedpermissions:ListPolicyStoreAliases 56
verifiedpermissions:DeletePolicyStoreAlias 56
정책 스토어 별칭 권한 제한 57
정책 스토어 스키마 59
스키마 편집 61
정책 검증 모드 64
정책 66
IAM 정책 생성 67
정적 정책 편집 69
..... 72
예제 컨텍스트 평가 74
테스트 정책 79
리소스당 정책 크기 82
정책 크기 계산 방법 82
할당량 내에서 유지 83
템플릿 연결 정책 크기 예제 84
정책 예제 86
대괄호 표기법을 사용하여 토큰 속성 참조 87
점 표기법을 사용하여 속성 참조 87

Amazon Cognito ID 토큰 속성을 반영합니다. 87
OIDC ID 토큰 속성을 반영합니다. 88
Amazon Cognito 액세스 토큰 속성을 반영합니다. 88
OIDC 액세스 토큰 속성을 반영합니다. 89
정책 템플릿 및 템플릿 연결 정책 90
정책 템플릿 생성 91
템플릿 연결 정책 생성 92
정책 템플릿 편집 94
템플릿 연결 정책 예제 96
PhotoFlash 예제 97
DigitalPetStore 예제 98
TinyToDo 예제 98
ID 원본 100
올바른 자격 증명 공급자 선택 100
Amazon Cognito 자격 증명 소스 작업 101
자격 증명 소스 생성 103
자격 증명 소스 편집 107
스키마에 토큰 매핑 109
클라이언트 및 대상 검증 119
OIDC 자격 증명 소스 작업 122
자격 증명 소스 생성 123
자격 증명 소스 편집 126
스키마에 토큰 매핑 128
클라이언트 및 대상 검증 134
통합 138
Express 사용 138
사전 조건 139
통합 설정 139
권한 부여 구성 140
권한 부여 미들웨어 구현 142
통합 테스트 143
문제 해결 144
다음 단계 144
요청 권한 부여 145
API 작업 146
테스트 모델 147

애플리케이션과 통합 148
API 요청 만들기 151
Verified Permissions 엔드포인트 151
요청 파라미터 151
요청 식별자 151
쿼리 API 인증 151
사용 가능한 라이브러리 151
POST 메서드를 사용하여 API 요청 152
보안 155
데이터 보호 155
데이터 암호화 157
고객 관리형 키 157
Identity and Access Management 176
대상 176
ID를 통한 인증 177
정책을 사용하여 액세스 관리 179
Amazon Verified Permissions의 작동 방식 IAM 180
IAM Verified Permissions에 대한 정책 186
ID 기반 정책 예시 188
AWS 관리형 정책 191
문제 해결 194
규정 준수 확인 196
복원성 196
모니터링 198
CloudTrail 로그 198
CloudTrail의 Verified Permissions 정보 198
Verified Permissions 로그 파일 항목 이해 199
작업 AWS CloudFormation 222
확인된 권한 및 CloudFormation 템플릿 222
AWS CDK 구문 223
에 대해 자세히 알아보기 CloudFormation 223
사용 AWS PrivateLink 224
고려 사항 224
인터페이스 엔드포인트 생성 224
엔드포인트 정책을 생성 225
할당량 227

리소스 할당량 227
계층 구조 할당량 229
초당 작업 할당량 230
용어 및 개념 234
권한 부여 모델 235
권한 부여 요청 235
권한 부여 응답 235
고려 대상 정책 235
컨텍스트 데이터 235
결정적 정책 236
엔터티 데이터 236
권한, 권한 부여 및 보안 주체 236
정책 적용 236
정책 스토어 236
정책 스토어 별칭 237
정책 이름 237
정책 템플릿 이름 237
충족 대상 정책 237
Cedar와의 차이점 238
정책 템플릿 지원 238
스키마 지원 238
작업 그룹 정의 238
엔터티 형식 239
길이 및 크기 제한 244
Cedar v4 FAQ 245
Amazon Verified Permissions를 호출할 때 Cedar 2가 더 이상 지원되지 않는다는 오류가 표시되는 이유는 무엇입니까? 245
일부 정책, 정책 템플릿 및 스키마가 Cedar 4와 호환되지 않는 이유는 무엇입니까? 246
정책 스토어가 Cedar 2를 사용하는지 아니면 Cedar 4를 사용하는지 어떻게 알 수 있나요? 246
Cedar 4로 업그레이드하려면 어떻게 해야 하나요? 247
정책 스토어를 Cedar 4에서 Cedar 2로 다운그레이드할 수 있나요? 248
정책 스토어가 Cedar 2에 대해 구성되었다는 오류 메시지가 표시되는 이유는 무엇입니까? 248
스키마를 Cedar 4와 호환되도록 하려면 어떻게 해야 하나요? 248
정책 및 템플릿을 Cedar 4와 호환되도록 하려면 어떻게 해야 하나요? 250
문서 이력 251
..... cclxiii

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 애플리케이션의 리소스를 보호하기 위한 세분화된 권한을 정의합니다.

Verified Permissions를 사용하여 정책 결정 시점(PDP)을 설정하는 지침과 예제는 AWS 권장 가이드의 Amazon Verified Permissions를 사용하여 PDP 구현을 참조하세요.

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 보안 주체가 애플리케이션의 지정된 컨텍스트에서 리소스에 대한 작업을 수행할 수 있는지 여부를 확인하여 권한을 부여합니다. Verified Permissions는 보안 주체가 이전에 OpenID Connect,와 같은 호스팅 공급자 Amazon Cognito또는 다른 인증 솔루션과 같은 프로토콜을 사용하는 등 다른 수단을 통해 식별되고 인증되었다고 가정합니다. Verified Permissions는 보안 주체가 관리되는 위치와 인증 방법에 구애받지 않습니다.

Verified Permissions는 고객이에서, Verified Permissions API를 사용하여 프로그래밍 방식으로 AWS Management Console또는와 같은 코드형 인프라를 통해 정책을 생성, 유지 관리 및 테스트할 수 있는 서비스입니다 CloudFormation. APIs 권한은 Cedar 정책 언어를 사용하여 표현됩니다. 클라이언트 애플리케이션은 권한 부여 API를 호출하여 서비스에 저장된 Cedar 정책을 평가하고 작업 허용 여부에 대한 액세스 결정을 제공합니다.

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 생성할 때 권한이 부여된 보안 주체, 인간 사용자 또는 시스템만 애플리케이션에 액세스할 수 있고 권한이 부여된 작업만 수행할 수 있는지 확인해야 합니다. Cedar를 사용하면 비즈니스 로직을 권한 부여 로직에서 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.

Verified Permissions는 현재 Cedar 버전 4.12를 사용합니다.

Cedar에 대한 자세한 내용은 다음을 참조하세요.

- Cedar 정책 언어 참조 가이드
- Cedar GitHub 리포지토리

Verified Permissions의 이점

애플리케이션 개발 가속화

권한 부여를 비즈니스 로직과 분리하여 애플리케이션 개발을 가속화합니다.

Verified Permissions는 널리 사용되는 개발 프레임워크와의 통합을 제공하므로 코드 변경을 최소화하면서 애플리케이션에서 권한 부여를 더 쉽게 구현할 수 있습니다. 이러한 통합을 통해 핵심 비즈니스 로직에 집중할 수 있으며 Verified Permissions가 권한 부여 결정을 처리합니다.

- Express.js - 기존 라우팅 핸들러를 수정하지 않고도 Express 애플리케이션의 API 엔드포인트를 보호할 수 있는 미들웨어 기반 통합입니다. 자세한 내용은 the section called “Express 사용” 섹션을 참조하세요.

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. Amazon Cognito는 OAuth 2.0 액세스 토큰 및 AWS 보안 인증을 위한 사용자 디렉터리, 인증 서버, 인증 서비스입니다. 정책 스토어를 생성할 때 Amazon Cognito 사용자 풀에서 보안 주체와 그룹을 빌드할 수 있습니다. 자세한 정보는 Amazon Cognito 개발자 안내서를 참조하세요.
- Amazon API Gateway – Amazon API Gateway는 규모와 관계없이 REST, HTTP 및 WebSocket API를 생성하고 게시하며 유지 관리하고 모니터링하며 보호하기 위한 AWS 서비스입니다. 정책 스토어를 생성할 때 API에서 작업과 리소스를 빌드할 수 있습니다 API Gateway. 에 대한 자세한 내용은 API Gateway 개발자 안내서를 API Gateway참조하세요.
- AWS IAM Identity Center— IAM Identity Center를 사용하면 직원 ID(직원 사용자)의 로그인 보안을 관리할 수 있습니다. IAM Identity Center는 작업 인력 사용자를 생성 또는 연결하고 모든 AWS 계정 및 애플리케이션에서 액세스를 중앙에서 관리할 수 있는 단일 위치를 제공합니다. 자세한 정보는 AWS IAM Identity Center 사용 설명서를 참조하세요.

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 Verified Permissions 콘솔

AWS 명령줄 도구

AWS 명령줄 도구를 사용하여 시스템의 명령줄에서 명령을 실행하여 Verified Permissions 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 는 AWS Command Line Interface (AWS CLI)와의 두 가지 명령줄 도구 세트를 제공합니다AWS Tools for Windows PowerShell. 설치 및 사용에 대한 자세한 내용은 AWS Command Line Interface 사용 설명서를 AWS CLI참조하세요. Tools for Windows PowerShell 도구 설치 및 사용에 대한 자세한 내용은 AWS Tools for PowerShell 사용 설명서를 참조하세요.

- AWS CLI 명령 참조의 verifiedpermissions
- 의 Amazon Verified Permissions AWS Tools for Windows PowerShell

AWS SDKs

AWS 는 다양한 프로그래밍 언어 및 플랫폼(Java, Python, Ruby, .NET, iOS, Android 등)을 위한 라이브러리 및 샘플 코드로 구성된 SDKs(소프트웨어 개발 키트)를 제공합니다. SDK를 사용하면 편리하게 Verified Permissions 및 AWS에 프로그래밍 방식으로 액세스할 수 있습니다. 예를 들어 SDK는 요청에 암호화 방식으로 서명, 오류 관리 및 자동으로 요청 재시도와 같은 작업을 처리합니다.

자세히 알아보고 AWS SDKs. Amazon Web Services

다음은 AWS SDKs.

- 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
- AWS SDK for Rust

AWS CDK 구문

AWS Cloud Development Kit (AWS CDK) 는 코드에서 클라우드 인프라를 정의하고 이를 프로비저닝하기 위한 오픈 소스 소프트웨어 개발 프레임워크입니다 CloudFormation. 구성 요소 또는 재사용 가능한 클라우드 구성 요소를 사용하여 템플릿을 생성할 CloudFormation 수 있습니다. 그런 다음 이러한 템플릿을 사용하여 클라우드 인프라를 배포할 수 있습니다.

자세한 내용을 알아보고 AWS CDK를 다운로드하려면 AWS 클라우드 개발 키트를 참조하세요.

다음은 구문과 같은 Verified Permissions AWS CDK 리소스에 대한 설명서 링크입니다.

- Amazon Verified Permissions L2 CDK 구성

Verified Permissions API

서비스에 직접 HTTPS 요청을 실행할 수 있는 Verified Permissions API를 사용하여 AWS 프로그래밍 방식으로 Verified Permissions 및에 액세스할 수 있습니다. API를 사용할 때는 자격 증명을 사용하여 요청에 디지털 방식으로 서명하는 코드를 포함해야 합니다.

- Amazon Verified Permissions API 참조 가이드

Verified Permissions 요금

Verified Permissions는 애플리케이션이 Verified Permissions에 요청한 월별 권한 부여 요청 수를 기준으로 단계별 요금을 제공합니다. 또한 애플리케이션이 Verified Permissions에 요청한 cURL(클라이언트 URL) 정책 API 요청의 월별 양에 따라 정책 관리 작업에 대한 요금도 부과됩니다.

Verified Permissions의 요금 및 가격을 안내하는 전체 목록은 Amazon Verified Permissions 요금을 참조하세요.

청구 요금은 AWS 결제 및 비용 관리 콘솔의 청구 및 비용 관리 대시보드에서 확인할 수 있습니다. 청구서에는 요금 내역을 자세하게 확인할 수 있는 사용 보고서 링크가 포함됩니다. AWS 계정 결제에 대한 자세한 내용은 AWS Billing 사용 설명서를 참조하세요.

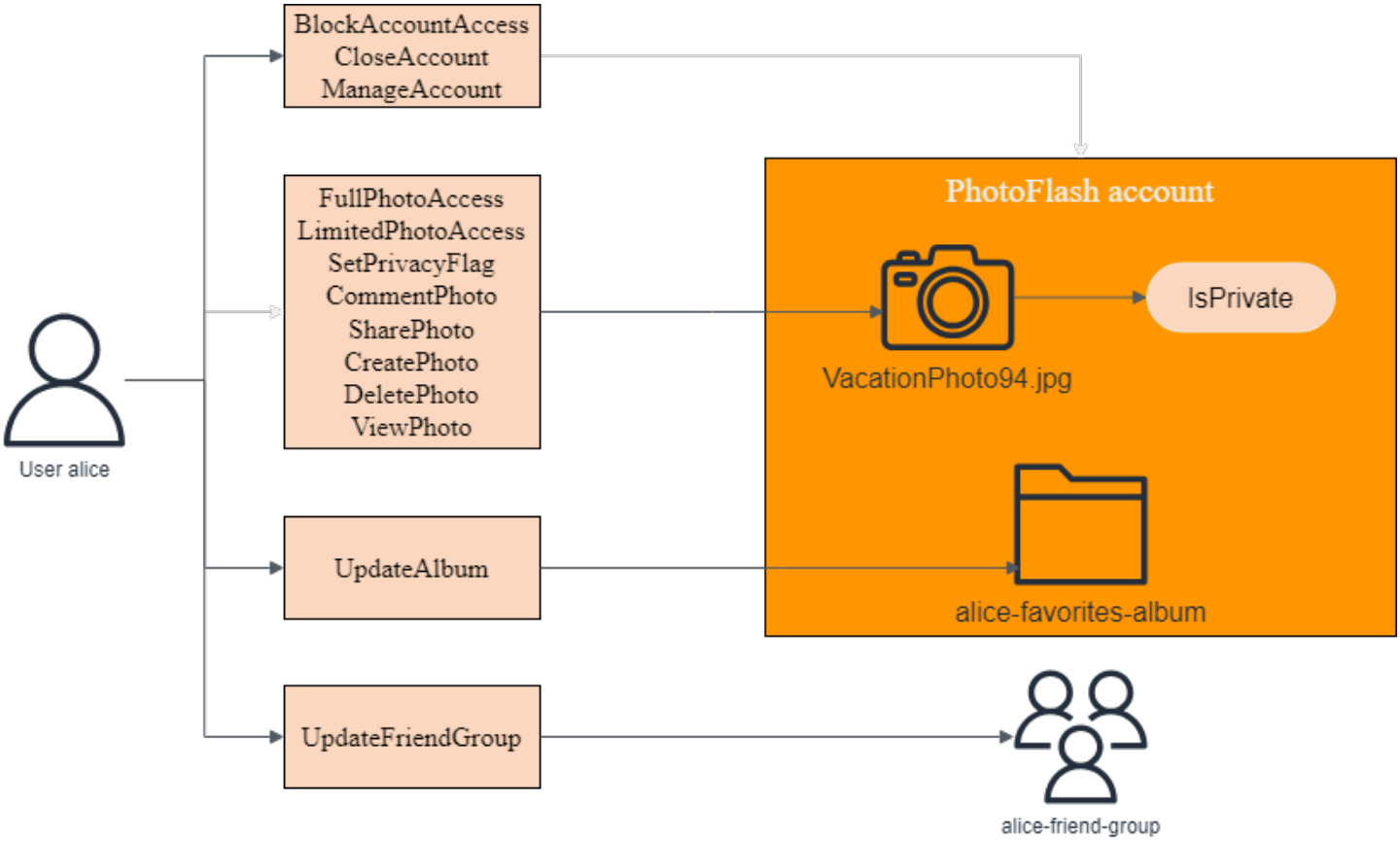
AWS 결제, 계정 및 이벤트에 대한 질문이 있는 경우에 문의 지원하세요.

첫 번째 Amazon Verified Permissions 정책 스토어 생성

이 자습서에서는 사용자가 사진 공유 애플리케이션의 개발자라고 가정하고 애플리케이션 사용자가 수행할 수 있는 작업을 제어하는 방법을 찾고 있다고 가정해 보겠습니다. 사진 및 사진 앨범을 추가, 삭제 또는 볼 수 있는 사용자를 제어하려고 합니다. 또한 사용자가 자신의 계정에 대해 수행할 수 있는 작업을 제어하려고 합니다. 자신의 계정을 관리할 수 있나요? 친구의 계정은 어떻게 되나요? 이러한 작업을 제어하려면 사용자의 자격 증명을 기반으로 이러한 작업을 허용하거나 금지하는 정책을 생성합니다. Verified Permissions는 이러한 정책을 수용하기 위한 정책 저장소 또는 컨테이너를 제공합니다.

이 자습서에서는 Amazon Verified Permissions 콘솔을 사용하여 샘플 정책 스토어를 생성하는 방법을 안내합니다. 콘솔은 몇 가지 샘플 정책 스토어 옵션을 제공하며 이제 PhotoFlash 정책 스토어를 생성하겠습니다. 이 정책 스토어를 사용하면 사용자와 같은 보안 주체가 사진 또는 앨범과 같은 리소스에서 공유와 같은 작업을 수행할 수 있습니다.

다음 다이어그램은 보안 주체, 및 다양한 리소스, 즉 PhotoFlash 계정`User::alice`, `VactionPhoto94.jpg` 파일, 사진 앨범 `alice-favorites-album`및 사용자 그룹에 대해 수행할 수 있는 작업 간의 관계를 보여줍니다`alice-friend-group`.



이제 PhotoFlash 정책 스토어를 이해했으므로 정책 스토어를 생성하고 살펴보겠습니다.

사전 조건

에 가입 AWS 계정

를 시작하려면이 AWS필요합니다 AWS 계정. 두 가지 방법으로 가입할 수 있습니다 AWS. 계정 구성 및 콘텐츠 저장 위치를 완벽하게 제어하거나 HIPAA 또는 FedRAMP와 같은 규제 대상 워크로드가 있는 경우 가입 AWS (고급)을 사용합니다. 신규 사용자이고 이미 소유한 로그인을 사용하여에 로그인 AWS 하고에서 사전 구성된 기본값이 있는 AWS 환경을 생성하려면 에 가입을 AWS AWS 리전사용합니다.

1단계: PhotoFlash 정책 스토어 생성

다음 절차에서는 AWS 콘솔을 사용하여 PhotoFlash 정책 스토어를 생성합니다.

PhotoFlash 정책 스토어를 생성하려면

1. Verified Permissions 콘솔에서 새 정책 스토어 생성을 선택합니다.
2. 시작 옵션에서 샘플 정책 스토어에서 시작을 선택합니다.
3. 샘플 프로젝트에서 PhotoFlash를 선택합니다.
4. 정책 스토어 생성을 선택합니다.

"Created and configured policy store" 메시지가 표시되면 개요로 이동을 선택하여 정책 스토어를 탐색합니다.

2단계: 정책 생성

정책 스토어를 생성할 때 사용자가 자신의 계정을 완전히 제어할 수 있도록 허용하는 기본 정책이 생성되었습니다. 이는 유용한 정책이지만, 목적을 위해 Verified Permissions의 미묘한 차이를 살펴보기 위해 보다 제한적인 정책을 생성해 보겠습니다. 자습서 앞부분에서 살펴본 다이어그램을 기억하면 작업을 수행할 수 있는 보안 주체 `User::alice`, `UpdateAlbum`리소스 ,이 있습니다`alice-favorites-album`. Alice만이 앨범을 관리할 수 있도록 허용하는 정책을 추가해 보겠습니다.

정책을 생성하려면

1. Verified Permissions 콘솔에서 1단계에서 생성한 정책 스토어를 선택합니다.
2. 탐색에서 정책을 선택합니다.

3. 정책 생성과 정적 정책 생성을 차례로 선택합니다.
4. 정책 효과에서 허용을 선택합니다.
5. 보안 주체 범위에서 특정 보안 주체를 선택한 다음 엔터티 유형 지정에서 PhotoFlash::User를 선택하고 엔터티 식별자 지정에를 입력합니다**alice**.
6. 리소스 범위에서 특정 리소스를 선택한 다음 엔터티 유형 지정에서 PhotoFlash::Album을 선택하고 엔터티 식별자 지정에를 입력합니다**alice-favorites-album**.
7. 작업 범위에서 특정 작업 세트를 선택한 다음 이 정책이 적용되어야 하는 작업(들)에서 UpdateAlbum을 선택합니다.
8. 다음(Next)을 선택합니다.
9. 세부 정보에서 정책 설명 - 선택 사항으로를 입력합니다**Policy allowing alice to update alice-favorites-album.**.
10. 정책 생성(Create policy)을 선택합니다.

이제 정책을 생성했으므로 Verified Permissions 콘솔에서 테스트할 수 있습니다.

3단계: 정책 스토어 테스트

정책 스토어와 정책을 생성한 후 Verified Permissions 테스트 벤치를 사용하여 시뮬레이션된 권한 부여 요청을 실행하여 테스트할 수 있습니다.

정책 스토어 정책을 테스트하려면

1. Verified Permissions 콘솔을 엽니다. 정책 스토어를 선택합니다.
2. 왼쪽에 있는 탐색 창에서 테스트 벤치를 선택합니다.
3. 비주얼 모드를 선택합니다.
4. 보안 주체의 경우 다음을 수행합니다.
   a. 작업을 수행하는 보안 주체에 PhotoFlash::User를 선택하고 엔터티 식별자 지정에를 입력합니다**alice**.
   b. 속성의 계정: 엔터티에서 PhotoFlash::Account 엔터티가 선택되어 있는지 확인하고 엔터티 식별자 지정에를 입력합니다**alice-account**.
5. 리소스의 보안 주체가 작업하는 리소스에서 PhotoFlash::Album 리소스 유형을 선택하고 개체 식별자 지정에를 입력합니다**alice-favorites-album**.
6. 작업의 경우 유효한 작업 목록에서 PhotoFlash::Action::"UpdateAlbum"을 선택합니다.

7. 페이지 상단에서 권한 부여 요청 실행을 선택하여 샘플 정책 스토어의 Cedar 정책에 대한 권한 부여 요청을 시뮬레이션합니다. 테스트 벤치에는 결정: 정책이 예상대로 작동하고 있음을 나타내는 허용이 표시되어야 합니다.

다음 표에는 Verified Permissions 테스트 벤치로 테스트할 수 있는 보안 주체, 리소스 및 작업에 대한 추가 값이 나와 있습니다. 이 테이블에는 PhotoFlash 샘플 정책 스토어에 포함된 정적 정책과 2단계에서 생성한 정책을 기반으로 한 권한 부여 요청 결정이 포함되어 있습니다.

보안 주체 값	보안 주체 계정: 엔터티 값	리소스 값	리소스 상위 값	작업	권한 부여 결정
PhotoFlash::User | bob	PhotoFlash::Account | alice-account	PhotoFlash::Album | alice-favorites-album	해당 사항 없음	PhotoFlash::Action::"UpdateAlbum"	거부
PhotoFlash::User | alice	PhotoFlash::Account | alice-account	PhotoFlash::Photo | photo.jpeg	PhotoFlash::Account | bob-account	PhotoFlash::Action::"ViewPhoto"	거부
PhotoFlash::User | alice	PhotoFlash::Account | alice-account	PhotoFlash::Photo | photo.jpeg	PhotoFlash::Account | alice-account	PhotoFlash::Action::"ViewPhoto"	허용
PhotoFlash::User | alice	PhotoFlash::Account | alice-account	PhotoFlash::Photo | bob-photo.jpeg	PhotoFlash::Album | Bob-Vacation-Album	PhotoFlash::Action::"DeletePhoto"	거부

4단계: 리소스 정리

정책 스토어 탐색을 완료한 후 삭제합니다.

정책 저장소 삭제

1. Verified Permissions 콘솔에서 1단계에서 생성한 정책 스토어를 선택합니다.
2. 탐색에서 설정을 선택합니다.

3. 정책 저장소 삭제에서 이 정책 저장소 삭제를 선택합니다.
4. 이 정책 스토어를 삭제하시겠습니까? 대화 상자에서 삭제를 입력한 다음 삭제를 선택합니다.

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 문이나 API 사양을 작성하여 애플리케이션의 다른 부분을 개발하기 시작하는 것과 비슷합니다. 대신 사용자 경험으로 시작해야 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? 어떻게 구성되어 있나요? 예를 들어, 파일은 폴더 내에 있나요?
- 리소스의 조직이 권한 모델에서 역할을 합니까?
- 보안 주체는 각 리소스에서 어떤 작업을 수행할 수 있나요?
- 보안 주체는 이러한 권한을 어떻게 획득하나요?
- 최종 사용자가 “관리자”, “운영자” 또는 “ReadOnly”와 같은 사전 정의된 권한 중에서 선택하게 하고 싶은가요? 아니면 임시 정책 설명을 생성해야 하나요? 아니면 둘 다 허용할까요?
- 역할은 전역적입니까 아니면 범위가 지정됩니까? 예를 들어 "작업자"가 단일 테넌트 내에서 제한됩니까, 아니면 "작업자"가 전체 애플리케이션 전반의 연산자를 의미합니까?
- 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 가장 중요한 것은 다른 것과 마찬가지로 사용자 경험 설계 연습이며 인터페이스 모형, 논리적 다이어그램, 권한이 제품에서 사용자가 수행할 수 있는 작업에 미치는 영향에 대한 전반적인 설명과 같은 아티팩트에서 나타날 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.

질문에 답하고 최적의 모델이 되도록 하려면 다음을 수행합니다.

- Cedar 정책 언어 참조 안내서의 Cedar 설계 패턴을 검토합니다.
- Cedar 정책 언어 참조 가이드의 모범 사례를 고려하세요.

- 이 페이지에 포함된 모범 사례를 고려하세요.

모범 사례

- 표준적인 “올바른” 모델은 없습니다.
- 404를 찾을 수 없음 오류가 아닌 403 금지 오류를 반환합니다.
- API 작업 이외의 리소스에 집중
- 다중 테넌시 고려 사항

표준적인 “올바른” 모델은 없습니다.

권한 부여 모델을 설계할 때 하나의 고유한 정답은 없습니다. 애플리케이션마다 유사한 개념에 대해 서로 다른 권한 부여 모델을 효과적으로 사용할 수 있으며, 그렇게 해도 괜찮습니다. 예를 들어 컴퓨터의 파일 시스템을 생각해 보세요. Unix와 유사한 운영 체제에서 파일을 생성할 때 상위 폴더에서 권한을 자동으로 상속하지 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.

404를 찾을 수 없음 오류가 아닌 403 금지 오류를 반환합니다.

404 찾을 수 없음 오류가 아닌 정책에 해당하지 않는 엔터티, 특히 리소스가 포함된 요청에 403 금지됨 오류를 반환하는 것이 가장 좋습니다. 이렇게 하면 엔터티가 존재하는지 여부를 노출하지 않으므로 요청이 정책 스토어의 정책에 있는 정책 조건을 충족하지 못하기 때문에 최고 수준의 보안이 제공됩니다.

API 작업 이외의 리소스에 집중

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

반면 다른 유형의 애플리케이션, 특히 웹 서비스는 API 작업 자체를 중심으로 권한을 설계하는 경우가 많습니다. 예를 들어 웹 서비스가 `createThing()`으로 이름이 지정된 API를 제공하는 경우 권한 부여 모델은 해당 권한을 정의하거나 `createThing`으로 이름이 지정된 Cedar `action`을 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 시작하기 프로세스에는 API에서 직접 리소스와 작업을 빌드하는 옵션이 포함되어 있습니다. 이는 정책 스토어와 정책 스토어가 권한을 부여하는 API 간의 직접 매핑이라는 유용한 기준입니다.

그러나 모델을 추가로 개발할 때이 API 중심 접근 방식은 매우 세분화된 권한 부여 모델이 있는 애플리케이션에 적합하지 않을 수 있습니다. APIs는 고객이 실제로 보호하려는 것, 즉 기본 데이터 및 리소스에 대한 프록시일 뿐이기 때문입니다. 여러 API가 동일한 리소스에 대한 액세스를 제어하는 경우 관리자는 해당 리소스에 대한 경로를 파악하고 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 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 처리하기 위해 새 API 작업을 추가하기로 선택하면 상황이 더욱 복잡해집니다.

- `isUserInGroups`(사용자가 하나 이상의 그룹에 속하는지 빠르게 테스트하기 위한 새 API)

보안 관점에서 볼 때 이 API는 그룹 멤버십을 발견할 수 있는 세 번째 경로를 열어 주어 신중하게 구성한 관리자의 권한을 방해하게 됩니다.

기본 데이터 및 리소스와 해당 연결 작업에 초점을 맞추는 것이 좋습니다. 이 접근 방식을 그룹 멤버십 예제에 적용하면 세 가지 API 작업이 각각 참조해야 하는 `viewGroupMembership`과 같은 추상 권한으로 이어집니다.

API 이름	권한
`listMembersOfGroup`	그룹에 대해 `viewGroupMembership` 권한이 필요합니다.

API 이름	권한
`listGroupMembershipsForUser`	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. 대신 이제 각 API 작업에 필요할 수 있는 몇 가지 권한을 문서화해야 하며 관리자는 권한을 구성할 때 이 설명서를 참조해야 합니다. 보안 요구 사항을 충족하기 위해 필요한 경우 이 방법은 유효한 절충안이 될 수 있습니다.

다중 테넌시 고려 사항

애플리케이션을 사용하는 기업 또는 테넌트 등 여러 고객이 사용할 애플리케이션을 개발하고 Amazon Verified Permissions와 통합하는 것이 좋습니다. 권한 부여 모델을 개발하기 전에 다중 테넌트 전략을 개발합니다. 하나의 공유 정책 저장소에서 고객의 정책을 관리하거나 각 테넌트당 정책 저장소를 할당할 수 있습니다. 자세한 내용은 AWS 권장 가이드의 Amazon Verified Permissions 멀티 테넌트 설계 고려 사항을 참조하세요.

1. 공유 정책 스토어 1개

   모든 테넌트는 단일 정책 스토어를 공유합니다. 애플리케이션은 모든 권한 부여 요청을 공유 정책 스토어로 보냅니다.

2. 테넌트별 정책 스토어

   각 테넌트에는 전용 정책 스토어가 있습니다. 애플리케이션은 요청을 하는 테넌트에 따라 권한 부여 결정을 위해 다양한 정책 스토어를 쿼리합니다.

두 전략 모두 AWS 청구서에 큰 영향을 미치지 않습니다. 그렇다면 접근 방식을 어떻게 설계해야 할까요? 다음은 Verified Permissions 다중 테넌시 권한 부여 전략에 기여할 수 있는 일반적인 조건입니다.

테넌트 정책 격리

테넌트 데이터를 보호하려면 각 테넌트의 정책을 다른 테넌트와 분리하는 것이 중요합니다. 각 테넌트에 자체 정책 스토어가 있는 경우 각 테넌트에는 자체 격리된 정책 세트가 있습니다.

권한 부여 흐름

테넌트별로 정책 저장소를 사용하여 요청에 정책 저장소 ID를 사용하여 권한 부여 요청을 하는 테넌트를 식별할 수 있습니다. 공유 정책 스토어를 사용하면 모든 요청이 동일한 정책 스토어 ID를 사용합니다.

템플릿 및 스키마 관리

애플리케이션에 여러 정책 스토어가 있는 경우 정책 템플릿과 정책 스토어 스키마는 각 정책 스토어에 일정 수준의 설계 및 유지 관리 오버헤드를 추가합니다.

글로벌 정책 관리

모든 테넌트에 일부 글로벌 정책을 적용할 수 있습니다. 글로벌 정책 관리를 위한 오버헤드 수준은 공유 및 테넌트별 정책 스토어 모델마다 다릅니다.

테넌트 오프보딩

일부 테넌트는 해당 사례에 맞는 스키마 및 정책에 요소를 제공합니다. 테넌트가 조직에서 더 이상 활성화되지 않고 데이터를 제거하려는 경우 작업 수준은 다른 테넌트와의 격리 수준에 따라 달라집니다.

서비스 리소스 할당량

Verified Permissions에는 다중 테넌시 결정에 영향을 미칠 수 있는 리소스 및 요청 속도 할당량이 있습니다. 할당량에 대한 자세한 내용은 리소스 할당량 섹션을 참조하세요.

공유 정책 스토어와 테넌트별 정책 스토어 비교

각 고려 사항에는 공유 및 테넌트별 정책 스토어 모델에서 고유한 수준의 시간과 리소스 약정이 필요합니다.

고려 사항	공유 정책 스토어의 노력 수준	테넌트별 정책 저장소의 노력 수준
테넌트 정책 격리	중간.Must include tenant identifiers in policies and authorization requests.	낮음. Isolation is default behavior. Tenant-specific policies are inaccessible to other tenants.

권한 부여 흐름	낮음. All queries target one policy store.	중간. Must maintain mappings between each tenant and their policy store ID.
템플릿 및 스키마 관리	낮음. Must make one schema work for all tenants.	높음 Schemas and templates might be less complex individually, but changes require more coordination and complexity.
글로벌 정책 관리	낮음. All policies are global and can be centrally updated.	높음 You must add global policies to each policy store in onboarding. Replicate global policy updates between many policy stores.
테넌트 오프보딩	높음 Must identify and delete only tenant-specific policies.	낮음. Delete the policy store.
서비스 리소스 할당량	높음 Tenants share resource quotas that affect policy stores like schema size, policy size per resource, and identity sources per policy store.	낮음. Each tenant has dedicated resource quotas.

선택 방법

각 멀티 테넌트 애플리케이션은 다릅니다. 아키텍처 결정을 내리기 전에 두 가지 접근 방식과 고려 사항을 신중하게 비교합니다.

애플리케이션에 테넌트별 정책이 필요하지 않고 단일 자격 증명 소스를 사용하는 경우 모든 테넌트에 대한 하나의 공유 정책 스토어가 가장 효과적인 솔루션일 수 있습니다. 그 결과 권한 부여 흐름과 글로벌 정책 관리가 간소화됩니다. 애플리케이션이 테넌트별 정책을 삭제할 필요가 없으므로 하나의 공유 정책 스토어를 사용하여 테넌트를 오프보딩하려면 노력이 덜 필요합니다.

그러나 애플리케이션에 테넌트별 정책이 많이 필요하거나 여러 자격 증명 소스를 사용하는 경우 테넌트별 정책 스토어가 가장 효과적일 수 있습니다. 각 정책 스토어에 테넌트당 권한을 부여하는 IAM 정책

을 사용하여 테넌트 정책에 대한 액세스를 제어할 수 있습니다. 테넌트 오프보딩에는 정책 스토어 삭제가 포함됩니다. shared-policy-store 환경에서는 테넌트별 정책을 찾아 삭제해야 합니다.

Amazon Verified Permissions 정책 스토어

정책 스토어는 정책 및 정책 템플릿의 컨테이너입니다. 각 정책 스토어에서 정책 스토어에 추가된 정책을 검증하는 데 사용되는 스키마를 생성할 수 있습니다. 또한 정책 검증을 켤 수 있습니다. 정책 검증이 활성화된 정책 스토어에 정책을 추가하면 정책에 정의된 엔터티 유형, 공통 유형 및 작업이 스키마에 대해 검증되고 잘못된 정책이 거부됩니다.

삭제 방지 기능은 정책 스토어의 우발적 삭제를 방지합니다. 를 통해 생성된 모든 새 정책 스토어에서 삭제 방지가 활성화됩니다 AWS Management Console. 반대로 API 또는 SDK 직접 호출을 통해 생성된 모든 정책 저장소에 대해서는 비활성화됩니다.

애플리케이션당 하나의 정책 스토어를 생성하거나 다중 테넌트 애플리케이션의 경우 테넌트당 하나의 정책 스토어를 생성하는 것이 좋습니다. 권한 부여 요청 시 정책 스토어를 지정해야 합니다. 정책 스토어 별칭을 생성하여 표시 이름으로 정책 스토어를 참조할 수도 있습니다. 자세한 내용은 Amazon Verified Permissions 정책 스토어 별칭 단원을 참조하십시오.

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. 예: `MyApplicationNamespace::exampleType`. Verified Permissions는 정책 스토어당 최대 100개의 네임스페이스를 지원합니다. 이러한 네임스페이스는 여러 유사한 애플리케이션으로 작업할 때 사물을 바로 유지하는 데 도움이 됩니다. 예를 들어 다중 테넌트 애플리케이션에서 네임스페이스를 사용하여 스키마에 정의된 유형에 테넌트 이름을 추가하면 다른 테넌트가 사용하는 유사한 것과 구별됩니다. 권한 부여 요청에 대한 로그를 볼 때 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 연결 정책 스토어
- 정책 스토어 삭제

Verified Permissions 정책 스토어 생성

다음 방법을 사용하여 정책 스토어를 생성할 수 있습니다.

- 안내 설정 따르기 - 첫 번째 정책을 생성하기 전에 유효한 작업과 보안 주체 유형을 사용하여 리소스 유형을 정의합니다.

- API Gateway 및 자격 증명 소스로 설정 - 자격 증명 공급자(IdP)로 로그인하는 사용자와 Amazon API Gateway API의 작업 및 리소스 엔터티를 사용하여 보안 주체 엔터티를 정의합니다. 애플리케이션이 사용자의 그룹 멤버십 또는 기타 속성을 사용하여 API 요청을 승인하도록 하려면이 옵션을 사용하는 것이 좋습니다.
- 샘플 정책 스토어에서 시작 - 사전 정의된 샘플 프로젝트 정책 스토어를 선택합니다. 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. Verified Permissions 콘솔에서 새 정책 스토어 생성을 선택합니다.
2. 시작 옵션 섹션에서 안내형 설정을 선택합니다.
3. 정책 스토어 설명을 입력합니다. 이 텍스트는 날씨 업데이트 웹 애플리케이션과 같이 현재 정책 스토어의 함수에 대한 친숙한 참조로 조직에 적합한 텍스트일 수 있습니다.
4. 세부 정보 섹션에서 스키마의 네임스페이스를 입력합니다. 네임스페이스에 대한 자세한 내용은 Cedar 설명서의 네임스페이스를 참조하세요.
5. 다음을 선택합니다.
6. 리소스 유형 창에서 리소스 유형의 이름을 입력합니다. 예를 들어는 날씨 업데이트 웹 애플리케이션의 리소스일 `currentTemperature` 수 있습니다.
7. (선택 사항) 속성 추가를 선택하여 리소스 속성을 추가합니다. 속성 이름을 입력하고 리소스의 각 속성에 대해 속성 유형을 선택합니다. 각 속성이 필수인지 여부를 선택합니다. 예를 들어는 `currentTemperature` 리소스의 속성일 `temperatureFormat` 수 있으며 화씨 또는 섭씨일 수 있습니다. 리소스 유형에 추가된 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. 예를 들어는 날씨 업데이트 웹 애플리케이션의 작업일 `viewTemperature` 수 있습니다. 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. 기본적으로 사용자는이 필드에 추가되지만 교체할 수 있습니다.
10. 다음을 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. Verified Permissions는 스키마에 대해 정책을 검증할 때 지정된 속성 값을 사용합니다. 보안 주체 유형에 추가된 속성을 제거하려면 속성 옆에 있는 제거를 선택합니다.
    - 에서 생성한 ID 또는 액세스 토큰에서 보안 주체의 ID 및 속성을 제공할 경우 Cognito 사용자 풀을 선택합니다 Amazon Cognito. 사용자 풀 연결을 선택합니다. 를 선택하고 연결할 사용자 풀의 사용자 풀 ID를 AWS 리전 입력합니다. Amazon Cognito 연결을 선택합니다. 자세한 내용은 Amazon Cognito 개발자 안내서의 Amazon Verified Permissions를 통한 권한 부여를 참조하세요.
    - 외부 OIDC 공급자가 생성한 ID 및/또는 액세스 토큰에서 보안 주체의 ID 및 속성을 추출할 경우 외부 OIDC 공급자를 선택하고 공급자 및 토큰 세부 정보를 추가합니다.
12. 다음을 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. 예를 들어는 날씨 업데이트 웹 애플리케이션의 엔터티 식별자일 `user-id` 수 있습니다.

      > **Note**
      >
      > 를 사용하는 경우 Amazon Cognito엔터티 식별자의 형식은 여야 합니다`<userpool-id>|<sub>`.

    - 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 리소스 필드에서 리소스를 선택하고 리소스의 개체 식별자를 입력합니다. 예를 들어는 날씨 업데이트 웹 애플리케이션의 엔터티 식별자일 `temperature-id` 수 있습니다.
- 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

로 설정 API Gateway 및 자격 증명 소스 구성 방법을 사용하여 정책 스토어를 생성하려면

이 API Gateway 옵션은 사용자의 그룹 또는 역할에서 권한 부여 결정을 내리도록 설계된 Verified Permissions 정책으로 APIs를 보호합니다. 이 옵션은 자격 증명 소스 그룹을 사용하여 권한 부여를 테스트하고 Lambda 권한 부여자를 사용하여 API를 테스트하기 위한 정책 스토어를 빌드합니다.

IdP의 사용자 및 해당 그룹은 보안 주체(ID 토큰) 또는 컨텍스트(액세스 토큰)가 됩니다. API의 메서 API Gateway 드와 경로는 정책이 승인하는 작업이 됩니다. 애플리케이션이 리소스가 됩니다. 이 워크플로의 결과로 Verified Permissions는 정책 스토어, Lambda 함수 및 API Lambda 권한 부여자를 생성합니다. 이 워크플로를 완료한 후 API에 Lambda 권한 부여자를 할당해야 합니다.

1. Verified Permissions 콘솔에서 새 정책 스토어 생성을 선택합니다.
2. 시작 옵션 섹션에서 API Gateway 및 자격 증명 소스를 사용하여 설정을 선택하고 다음을 선택합니다.
3. 리소스 및 작업 가져오기 단계의 API에서 정책 스토어 리소스 및 작업에 대한 모델로 작동하는 API를 선택합니다.

    a. API에 구성된 단계에서 배포 단계를 선택하고 API 가져오기를 선택합니다. API 단계에 대한 자세한 내용은 Amazon API Gateway 개발자 안내서의 REST API에 대한 단계 설정을 참조하세요.
    b. 가져온 리소스 및 작업 맵을 미리 봅니다.
    c. 리소스 또는 작업을 업데이트하려면 API Gateway 콘솔에서 API 경로 또는 메서드를 수정하고 API 가져오기를 선택하여 업데이트를 확인합니다.

d. 선택한 항목에 만족하면 다음을 선택합니다.

4. 자격 증명 소스에서 자격 증명 공급자 유형을 선택합니다. Amazon Cognito 사용자 풀 또는 OpenID Connect(OIDC) IdP 유형을 선택할 수 있습니다.

5. 를 선택한 경우Amazon Cognito:

   a. 정책 스토어와 동일한 AWS 리전 및 AWS 계정 에서 사용자 풀을 선택합니다.

   b. 권한 부여를 위해 제출하려는 API에 전달할 토큰 유형을 선택합니다. 두 토큰 유형 모두이 API 연결 권한 부여 모델의 기반인 사용자 그룹을 포함합니다.

   c. 앱 클라이언트 검증에서 정책 스토어의 범위를 다중 테넌트 사용자 풀의 Amazon Cognito 앱 클라이언트 하위 집합으로 제한할 수 있습니다. 사용자가 사용자 풀에서 하나 이상의 지정된 앱 클라이언트로 인증하도록 하려면 예상 앱 클라이언트 IDs가 있는 토큰만 수락을 선택합니다. 사용자 풀로 인증하는 사용자를 수락하려면 앱 클라이언트 IDs 선택합니다.

   d. 다음을 선택합니다.

6. 외부 OIDC 공급자를 선택한 경우:

   a. 발급자 URL에 OIDC 발급자의 URL을 입력합니다. 이는 권한 부여 서버, 서명 키 및 와 같은 공급자에 대한 기타 정보를 제공하는 서비스 엔드포인트입니다`https://auth.example.com`. 발급자 URL은에서 OIDC 검색 문서를 호스팅해야 합니다`/.well-known/openid-configuration`.

   b. 토큰 유형에서 애플리케이션이 권한 부여를 위해 제출할 OIDC JWT 유형을 선택합니다. 자세한 내용은 스키마에 Amazon Cognito 토큰 매핑 및 스키마에 OIDC 토큰 매핑을 참조하세요.

   c. (선택 사항) 토큰 클레임 - 선택 사항에서 토큰 클레임 추가를 선택하고 토큰 이름을 입력한 다음 값 유형을 선택합니다.

   d. 사용자 및 그룹 토큰 클레임에서 다음을 수행합니다.

      i. 자격 증명 소스의 토큰에 사용자 클레임 이름을 입력합니다. 이는 일반적으로 평가할 엔터티의 고유 식별자를 포함하는 ID 또는 액세스 토큰`sub`의 클레임입니다. 연결된 OIDC IdP의 ID는 정책 스토어의 사용자 유형에 매핑됩니다.

      ii. 자격 증명 소스의 토큰에 그룹 클레임 이름을 입력합니다. 이는 일반적으로 사용자 그룹 목록이 포함된 ID 또는 액세스 토큰`groups`의 클레임입니다. 정책 스토어는 그룹 멤버십을 기반으로 요청을 승인합니다.

   e. 대상 검증에서 정책 스토어가 권한 부여 요청에서 수락할 값을 `Add value` 선택하고 추가합니다.

f. 다음을 선택합니다.

7. 를 선택한 경우 Amazon CognitoVerified Permissions는 사용자 풀에서 그룹을 쿼리합니다. OIDC 공급자의 경우 그룹 이름을 수동으로 입력합니다. 그룹에 작업 할당 단계에서는 그룹 구성원이 작업을 수행하도록 허용하는 정책 스토어에 대한 정책을 생성합니다.

   a. 정책에 포함할 그룹을 선택하거나 추가합니다.

   b. 선택한 각 그룹에 작업을 할당합니다.

   c. 다음을 선택합니다.

8. 앱 통합 배포에서 나중에 수동으로 Lambda 권한 부여자를 연결할지 아니면 Verified Permissions가 대신 연결하도록 할지 선택하고 Verified Permissions가 정책 스토어 및 Lambda 권한 부여자를 생성하기 위해 수행할 단계를 검토합니다.

9. 새 리소스를 생성할 준비가 되면 정책 스토어 생성을 선택합니다.

10. 브라우저에서 정책 스토어 상태 단계를 열어 두어 Verified Permissions의 리소스 생성 진행 상황을 모니터링합니다.

11. 보통 약 1시간 후 또는 Lambda 권한 부여자 배포 단계에 성공이 표시되면 권한 부여자를 수동으로 연결하도록 선택한 경우 권한 부여자를 구성합니다.

    Verified Permissions는 API에 Lambda 함수와 Lambda 권한 부여자를 생성했습니다. API 열기를 선택하여 API로 이동합니다.

    Lambda 권한 부여자를 할당하는 방법을 알아보려면 Amazon API Gateway 개발자 안내서의 API Gateway Lambda 권한 부여자 사용을 참조하세요.

    a. API의 권한 부여자로 이동하여 Verified Permissions가 생성한 권한 부여자의 이름을 기록해 둡니다.

    b. 리소스로 이동하여 API에서 최상위 메서드를 선택합니다.

    c. 메서드 요청 설정에서 편집을 선택합니다.

    d. 권한 부여자를 이전에 기록한 권한 부여자 이름으로 설정합니다.

    e. HTTP 요청 헤더를 확장AUTHORIZATION하고 이름 또는를 입력한 다음 필수를 선택합니다.

    f. API 단계를 배포합니다.

    g. 변경 사항을 저장합니다.

12. 자격 증명 소스 선택 단계에서 선택한 토큰 유형의 사용자 풀 토큰으로 권한 부여자를 테스트합니다. 사용자 풀 로그인 및 토큰 검색에 대한 자세한 내용은 Amazon Cognito 개발자 안내서의 사용자 풀 인증 흐름을 참조하세요.

13. API에 대한 요청 AUTHORIZATION 헤더에서 사용자 풀 토큰으로 인증을 다시 테스트합니다.
14. 새 정책 스토어를 검사합니다. 정책을 추가하고 구체화합니다.

Sample policy store

샘플 정책 스토어 구성 방법을 사용하여 정책 스토어를 생성하려면

1. 시작 옵션 섹션에서 샘플 정책 스토어를 선택합니다.
2. 샘플 프로젝트 섹션에서 사용할 샘플 Verified Permissions 애플리케이션의 유형을 선택합니다.

   - PhotoFlash는 사용자가 개별 사진 및 앨범을 친구와 공유할 수 있는 고객용 샘플 웹 애플리케이션입니다. 사용자는 사진을 보거나 댓글을 달고 재공유할 수 있는 사람의 권한을 세분화하여 설정할 수 있습니다. 계정 소유자는 친구 그룹을 만들고 사진을 앨범으로 정리할 수도 있습니다.
   - DigitalPetStore는 누구나 등록하여 고객이 될 수 있는 샘플 애플리케이션입니다. 고객은 판매할 반려동물을 추가하거나 반려동물을 검색하고 주문할 수 있습니다. 반려동물을 추가한 고객은 반려동물 소유자로 기록됩니다. 반려동물 소유자는 반려동물의 세부 정보를 업데이트하거나 반려동물 이미지를 업로드하고 반려동물 등록 내용을 삭제할 수 있습니다. 반려동물을 주문한 고객은 주문 소유자로 기록됩니다. 주문 소유자는 주문에 대한 세부 정보를 확인하거나 주문을 취소할 수 있습니다. 반려동물 스토어 매니저에게는 관리자 액세스 권한이 부여됩니다.

     > **Note**
     >
     > DigitalPetStore 샘플 정책 스토어에는 정책 템플릿이 포함되어 있지 않습니다. PhotoFlash 및 TinyTodo 샘플 정책 스토어에는 정책 템플릿이 포함되어 있습니다.

   - TinyTodo는 사용자가 작업 및 작업 목록을 생성할 수 있는 샘플 애플리케이션입니다. 목록 소유자는 목록을 관리 및 공유하고 목록을 보거나 편집할 수 있는 사람을 지정할 수 있습니다.
3. 샘플 정책 스토어의 스키마 네임스페이스는 선택한 샘플 프로젝트에 따라 자동으로 생성됩니다.
4. 정책 스토어 생성을 선택합니다.

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 Verified Permissions 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 업데이트에 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 Verified Permissions 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 생성하는 정책 스토어 AWS CLI 는 비어 있습니다.
>
> - 스키마를 추가하려면 Amazon Verified Permissions 정책 스토어 스키마를 참조하세요.
> - 정책을 추가하려면 Amazon Verified Permissions 정적 정책 생성을 참조하세요.
> - 정책 템플릿을 추가하려면 Amazon Verified Permissions 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

`CreatePolicyStore` API를 사용하여 정책 스토어를 생성할 수 있습니다. 자세한 내용은 Amazon Verified Permissions API 참조 가이드의 CreatePolicyStore를 참조하세요.

AWS SDK를 사용하여 Rust에서 Amazon Verified Permissions 구현

이 주제에서는 SDK를 사용하여 Rust에서 Amazon Verified Permissions를 구현하는 실제 예제를 AWS 제공합니다. 이 예제에서는 사용자가 사진을 볼 수 있는지 테스트할 수 있는 권한 부여 모델을 개발하는 방법을 보여줍니다. 샘플 코드는의 aws-sdk-verifiedpermissions 크레이트를 사용합니다. AWS SDK for Rust이 크레이트는와 상호 작용하기 위한 강력한 도구 세트를 제공합니다 AWS 서비스.

사전 조건

시작하기 전에 시스템에 AWS CLI가 구성되어 있고 Rust에 익숙해야 합니다.

- 설치에 대한 지침은 AWS CLI 설치 가이드를 AWS CLI참조하세요.
- 구성에 대한 지침은 의 구성 및 AWS CLI 구성 및 자격 증명 파일 설정을 AWS CLI참조하세요. AWS CLI
- Rust에 대한 자세한 내용은 rust-lang.org 및 AWS SDK for Rust 개발자 안내서를 참조하세요.

환경을 준비하면서 Rust에서 Verified Permissions를 구현하는 방법을 살펴보겠습니다.

샘플 코드 테스트

샘플 코드는 다음을 수행합니다.

- 와 통신할 SDK 클라이언트를 설정합니다. AWS
- 정책 스토어를 생성합니다.
- 스키마를 추가하여 정책 스토어의 구조를 정의합니다.
- 권한 부여 요청을 확인하는 정책을 추가합니다.
- 테스트 권한 부여 요청을 전송하여 모든 항목이 올바르게 설정되었는지 확인합니다.

샘플 코드를 테스트하려면

1. Rust 프로젝트를 생성합니다.
2. 의 기존 코드를 다음 코드main.rs로 바꿉니다.

```
use std::time::Duration;
use std::thread::sleep;
use aws_config::BehaviorVersion;
use aws_sdk_verifiedpermissions::Client;
use aws_sdk_verifiedpermissions::{
    operation::{
        create_policy::CreatePolicyOutput,
        create_policy_store::CreatePolicyStoreOutput,
        is_authorized::IsAuthorizedOutput,
        put_schema::PutSchemaOutput,
    },
    types::{
        ActionIdentifier, EntityIdentifier, PolicyDefinition, SchemaDefinition,
 StaticPolicyDefinition, ValidationSettings
    },
};

//Function that creates a policy store in the client that's passed
async fn create_policy_store(client: &Client, valid_settings: &ValidationSettings)-
> CreatePolicyStoreOutput {
    let policy_store =
 client.create_policy_store().validation_settings(valid_settings.clone()).send().await;
    return policy_store.unwrap();
}

//Function that adds a schema to the policy store in the client
async fn put_schema(client: &Client, ps_id: &str, schema: &str) -> PutSchemaOutput
 {
    let schema =
 client.put_schema().definition(SchemaDefinition::CedarJson(schema.to_string())).policy_sto
    return schema.unwrap();
}

//Function that creates a policy in the policy store in the client
async fn create_policy(client: &Client, ps_id: &str,
 policy_definition:&PolicyDefinition) -> CreatePolicyOutput {
    let create_policy =
 client.create_policy().definition(policy_definition.clone()).policy_store_id(ps_id).send()
```

```
    return create_policy.unwrap();
}

//Function that tests the authorization request to the policy store in the client
async fn authorize(client: &Client, ps_id: &str, principal: &EntityIdentifier,
 action: &ActionIdentifier, resource: &EntityIdentifier) -> IsAuthorizedOutput {
    let is_auth =
 client.is_authorized().principal(principal.to_owned()).action(action.to_owned()).resource(
    return is_auth.unwrap();
}

#[::tokio::main]
async fn main() -> Result<(), aws_sdk_verifiedpermissions::Error> {

//Set up SDK client
    let config = aws_config::load_defaults(BehaviorVersion::latest()).await;
    let client = aws_sdk_verifiedpermissions::Client::new(&config);

//Create a policy store
    let valid_settings = ValidationSettings::builder()
    .mode({aws_sdk_verifiedpermissions::types::ValidationMode::Strict
    })
    .build()
    .unwrap();
    let policy_store = create_policy_store(&client, &valid_settings).await;
    println!(
    "Created Policy store with ID: {:?}",
    policy_store.policy_store_id
    );

//Add schema to policy store
    let schema= r#"{
        "PhotoFlash": {
            "actions": {
                "ViewPhoto": {
                    "appliesTo": {
                        "context": {
                            "type": "Record",
                            "attributes": {}
                        },
                        "principalTypes": [
                            "User"
                        ],
                        "resourceTypes": [
```

```
                            "Photo"
                        ]
                    },
                    "memberOf": []
                }
            },
            "entityTypes": {
                "Photo": {
                    "memberOfTypes": [],
                    "shape": {
                        "type": "Record",
                        "attributes": {
                            "IsPrivate": {
                                "type": "Boolean"
                            }
                        }
                    }
                },
                "User": {
                    "memberOfTypes": [],
                    "shape": {
                        "attributes": {},
                        "type": "Record"
                    }
                }
            }
        }
    }"#;
    let put_schema = put_schema(&client, &policy_store.policy_store_id,
 schema).await;
    println!(
        "Created Schema with Namespace: {:?}",
        put_schema.namespaces
    );

//Create policy
    let policy_text = r#"
        permit (
            principal in PhotoFlash::User::"alice",
            action == PhotoFlash::Action::"ViewPhoto",
            resource == PhotoFlash::Photo::"VacationPhoto94.jpg"
        );
        "#;
```

```
    let policy_definition =
 PolicyDefinition::Static(StaticPolicyDefinition::builder().statement(policy_text).build().
    let policy = create_policy(&client, &policy_store.policy_store_id,
 &policy_definition).await;
    println!(
        "Created Policy with ID: {:?}",
        policy.policy_id
    );

//Break to make sure the resources are created before testing authorization
    sleep(Duration::new(2, 0));

//Test authorization
    let principal=
 EntityIdentifier::builder().entity_id("alice").entity_type("PhotoFlash::User").build().unw
    let action =
 ActionIdentifier::builder().action_type("PhotoFlash::Action").action_id("ViewPhoto").build
    let resource =
 EntityIdentifier::builder().entity_id("VacationPhoto94.jpg").entity_type("PhotoFlash::Phot
    let auth = authorize(&client, &policy_store.policy_store_id, &principal,
 &action, &resource).await;
    println!(
        "Decision: {:?}",
        auth.decision
        );
        println!(
        "Policy ID: {:?}",
        auth.determining_policies
        );
     Ok(())
}
```

3. 터미널에 `cargo run`를 입력하여 코드를 실행합니다.

코드가 올바르게 실행되면 터미널에 결정 정책의 정책 ID가 `Decision: Allow` 표시됩니다. 즉, 정책 스토어를 성공적으로 생성하고 AWS SDK for Rust를 사용하여 테스트했습니다.

리소스 정리

정책 스토어 탐색을 완료한 후 삭제합니다.

정책 저장소 삭제

를 삭제하려는 정책 스토어 ID*PSEXAMPLEabcdefg111111*로 바꾸는 `delete-policy-store` 작업을 사용하여 정책 스토어를 삭제할 수 있습니다.

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

성공하면 이 명령은 출력을 생성하지 않습니다.

API 연결 정책 스토어

일반적인 사용 사례는 Amazon Verified Permissions를 사용하여 Amazon API Gateway에서 호스팅되는 APIs에 대한 사용자 액세스를 승인하는 것입니다. AWS 콘솔에서 마법사를 사용하여 Amazon Cognito 또는 OIDC 자격 증명 공급자(IdP)에서 관리되는 사용자를 위한 역할 기반 액세스 정책을 생성하고 Verified Permissions를 호출하는 AWS Lambda 권한 부여자를 배포하여 이러한 정책을 평가할 수 있습니다.

마법사를 완료하려면 새 정책 스토어를 생성할 때 API Gateway 및 자격 증명 공급자로 설정을 선택하고 단계를 따릅니다.

API 연결 정책 스토어가 생성되고 권한 부여 요청에 대한 권한 부여 모델과 리소스를 프로비저닝합니다. 정책 스토어에는 API Gateway Verified Permissions에 연결하는 자격 증명 소스와 Lambda 권한 부여자가 있습니다. 정책 스토어가 생성되면 사용자의 그룹 멤버십을 기반으로 API 요청을 승인할 수 있습니다. 예를 들어 Verified Permissions는 `Directors` 그룹의 구성원인 사용자에게만 액세스 권한을 부여할 수 있습니다.

애플리케이션이 확장되면 Cedar 정책 언어를 사용하여 사용자 속성 및 OAuth 2.0 범위를 사용하여 세분화된 권한 부여를 구현할 수 있습니다. 예를 들어 Verified Permissions는 도메인에 `email` 속성이 있는 사용자에게만 액세스 권한을 부여할 수 있습니다`mycompany.co.uk`.

API에 대한 권한 부여 모델을 설정한 후 나머지 책임은 사용자를 인증하고 애플리케이션에서 API 요청을 생성하고 정책 스토어를 유지하는 것입니다.

데모를 보려면 Amazon Web Services YouTube 채널의 Amazon Verified Permissions - Quick Start Overview and Demo를 참조하세요.

주제

- Verified Permissions가 API 요청을 승인하는 방법
- API 연결 정책 스토어에 대한 고려 사항

- 속성 기반 액세스 제어(ABAC) 추가
- 를 사용하여 프로덕션으로 전환 AWS CloudFormation
- API 연결 정책 스토어 문제 해결

> **Important**
>
> 로 설정 API Gateway 및 Verified Permissions 콘솔의 자격 증명 소스 옵션으로 생성한 정책 스토어는 프로덕션에 즉시 배포하기 위한 것이 아닙니다. 초기 정책 스토어를 사용하여 권한 부여 모델을 완료하고 정책 스토어 리소스를 CloudFormation으로 내보냅니다. 를 사용하여 프로그래밍 방식으로 Verified Permissions를 프로덕션에 배포합니다AWS Cloud Development Kit (AWS CDK). 자세한 내용은 를 사용하여 프로덕션으로 전환 AWS CloudFormation 단원을 참조하십시오.

API 및 자격 증명 소스에 연결된 정책 스토어에서 애플리케이션은 API에 요청할 때 권한 부여 헤더에 사용자 풀 토큰을 제공합니다. 정책 스토어의 자격 증명 소스는 Verified Permissions에 대한 토큰 검증을 제공합니다. 토큰은 IsAuthorizedWithToken API를 사용하여 권한 부여 요청`principal`에서를 형성합니다. Verified Permissions는 사용자 풀과 같은 ID(ID) 및 액세스 토큰의 그룹 클레임에 표시된 대로 사용자의 그룹 멤버십에 `cognito:groups` 대한 정책을 빌드합니다. API는 Lambda 권한 부여자의 애플리케이션에서 토큰을 처리하고 권한 부여 결정을 위해 Verified Permissions에 제출합니다. API는 Lambda 권한 부여자로부터 권한 부여 결정을 받으면 요청을 데이터 소스에 전달하거나 요청을 거부합니다.

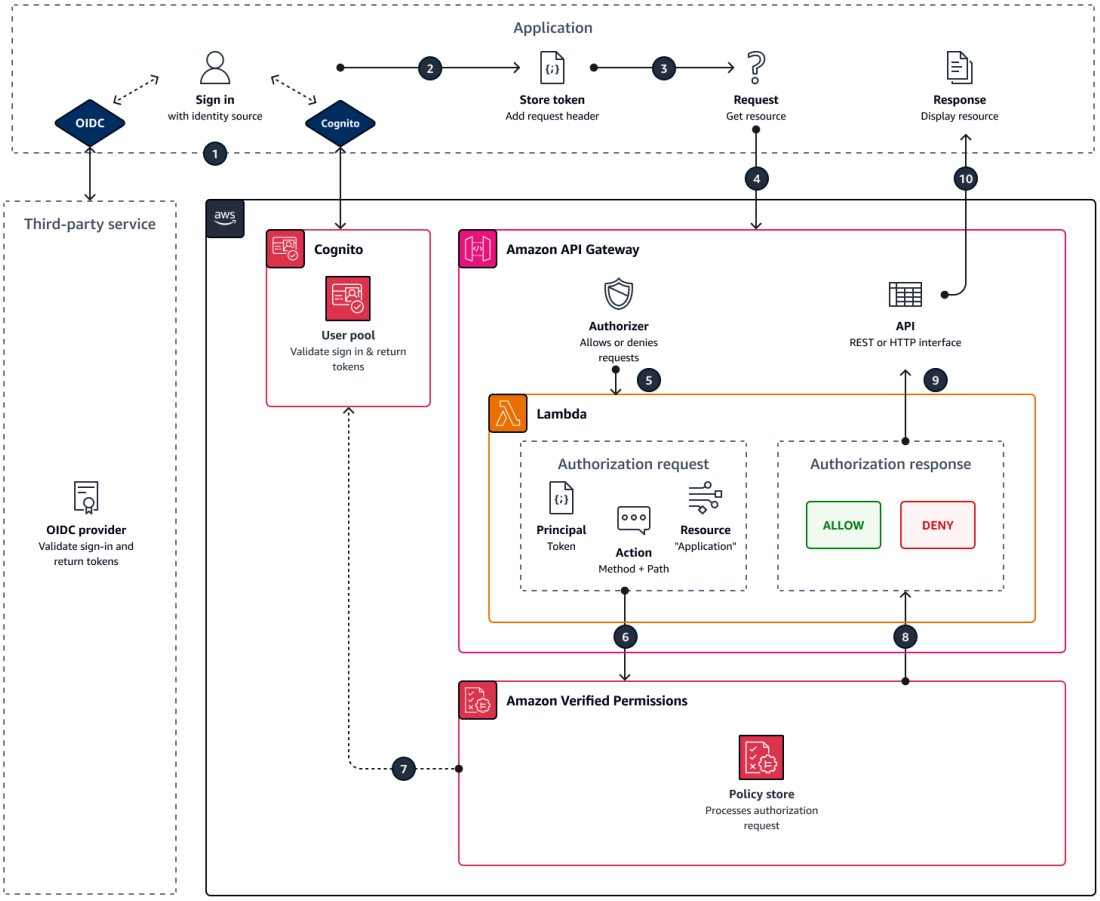
Verified Permissions를 사용한 자격 증명 소스 및 API Gateway 권한 부여의 구성 요소

- Amazon Cognito 사용자를 인증하고 그룹화하는 사용자 풀 또는 OIDC IdP입니다. 사용자의 토큰은 그룹 멤버십과 Verified Permissions가 정책 스토어에서 평가하는 보안 주체 또는 컨텍스트를 채웁니다.
- API Gateway REST API. Verified Permissions는와 같은 API 경로 및 API 메서드의 작업을 정의합니다`MyAPI::Action::get /photo`.
- API에 대한 Lambda 함수 및 Lambda 권한 부여자. Lambda 함수는 사용자 풀에서 보유자 토큰을 가져와 Verified Permissions에 권한 부여를 요청하고 결정을 반환합니다 API Gateway. 로 설정 API Gateway 및 자격 증명 소스 워크플로는 자동으로이 Lambda 권한 부여자를 생성합니다.
- Verified Permissions 정책 스토어. 정책 스토어 자격 증명 소스는 Amazon Cognito 사용자 풀 또는 OIDC 공급자 그룹입니다. 정책 스토어 스키마는 API의 구성을 반영하며, 정책은 사용자 그룹을 허용된 API 작업에 연결합니다.

- IdP로 사용자를 인증하고 API 요청에 토큰을 추가하는 애플리케이션입니다.

Verified Permissions가 API 요청을 승인하는 방법

새 정책 스토어를 생성하고 로 설정 API Gateway 및 자격 증명 소스 옵션을 선택하면 Verified Permissions가 정책 스토어 스키마 및 정책을 생성합니다. 스키마 및 정책은 API 작업과 작업을 수행할 권한을 부여하려는 사용자 그룹을 반영합니다. Verified Permissions는 Lambda 함수 및 권한 부여자도 생성합니다.



1. 사용자가 Amazon Cognito 또는 다른 OIDC IdP를 통해 애플리케이션으로 로그인합니다. IdP는 사용자 정보와 함께 ID 및 액세스 토큰을 발급합니다.
2. 애플리케이션JWTs를 저장합니다. 자세한 내용은 Amazon Cognito 개발자 안내서의 사용자 풀에서 토큰 사용을 참조하세요.
3. 사용자는 애플리케이션이 외부 API에서 검색해야 하는 데이터를 요청합니다.
4. 애플리케이션이의 REST API에서 데이터를 요청합니다 API Gateway. ID 또는 액세스 토큰을 요청 헤더로 추가합니다.

5. API에 권한 부여 결정을 위한 캐시가 있는 경우 이전 응답을 반환합니다. 캐싱이 비활성화되거나 API에 현재 캐시가 없는 경우 API Gateway 는 요청 파라미터를 토큰 기반 Lambda 권한 부여자에게 전달합니다.
6. Lambda 함수는 IsAuthorizedWithToken API를 사용하여 Verified Permissions 정책 스토어에 권한 부여 요청을 보냅니다. Lambda 함수는 권한 부여 결정의 요소를 전달합니다.
   a. 보안 주체로서의 사용자 토큰입니다.
   b. API 경로와 결합된 API 메서드, 예: 작업`GetPhoto`.
   c. 리소스`Application`로서의 용어입니다.
7. Verified Permissions는 토큰을 검증합니다. Amazon Cognito 토큰 검증 방법에 대한 자세한 내용은 Amazon Cognito 개발자 안내서의 Amazon Verified Permissions 권한 부여를 참조하세요.
8. Verified Permissions는 정책 스토어의 정책을 기준으로 권한 부여 요청을 평가하고 권한 부여 결정을 반환합니다.
9. Lambda 권한 부여자는 `Allow` 또는 `Deny` 응답을 반환합니다 API Gateway.
10. API는 애플리케이션에 데이터 또는 `ACCESS_DENIED` 응답을 반환합니다. 애플리케이션은 API 요청의 결과를 처리하고 표시합니다.

API 연결 정책 스토어에 대한 고려 사항

Verified Permissions 콘솔에서 API 연결 정책 스토어를 빌드하면 최종 프로덕션 배포에 대한 테스트를 생성합니다. 프로덕션으로 이동하기 전에 API 및 사용자 풀에 대한 고정 구성을 설정합니다. 다음 요인을 고려하세요.

API Gateway 는 응답을 캐싱합니다.

API 연결 정책 스토어에서 Verified Permissions는 권한 부여 캐싱 TTL이 120초인 Lambda 권한 부여자를 생성합니다. 이 값을 조정하거나 권한 부여자에서 캐싱을 끌 수 있습니다. 캐싱이 활성화된 권한 부여자에서 권한 부여자는 TTL이 만료될 때까지 매번 동일한 응답을 반환합니다. 이렇게 하면 요청된 단계의 캐싱 TTL과 동일한 기간만큼 사용자 풀 토큰의 유효 수명을 연장할 수 있습니다.

Amazon Cognito 그룹을 재사용할 수 있음

Amazon Verified Permissions는 사용자 ID 또는 액세스 토큰의 `cognito:groups` 클레임에서 사용자 풀 사용자의 그룹 멤버십을 결정합니다. 이 클레임의 값은 사용자가 속한 사용자 풀 그룹의 기억하기 쉬운 이름 배열입니다. 사용자 풀 그룹을 고유 식별자와 연결할 수 없습니다.

정책 스토어에 동일한 이름으로 삭제하고 다시 생성하는 사용자 풀 그룹은 동일한 그룹과 동일합니다. 사용자 풀에서 그룹을 삭제할 때 정책 스토어에서 그룹에 대한 모든 참조를 삭제합니다.

API에서 파생된 네임스페이스 및 스키마는 point-in-time

Verified Permissions는 특정 시점에 API를 캡처합니다. 정책 스토어를 생성할 때만 API를 쿼리합니다. API의 스키마 또는 이름이 변경되면 정책 저장소 및 Lambda 권한 부여자를 업데이트하거나 새 API 연결 정책 저장소를 생성해야 합니다. Verified Permissions는 API 이름에서 정책 스토어 네임스페이스를 파생합니다.

Lambda 함수에는 VPC 구성이 없습니다.

API 권한 부여자에 대해 Verified Permissions가 생성하는 Lambda 함수는 기본 VPC에서 시작됩니다. 기본적으로 입니다. 프라이빗 VPCs로 제한된 네트워크 액세스 권한이 있는 APIs는 Verified Permissions로 액세스 요청을 승인하는 Lambda 함수와 통신할 수 없습니다.

Verified Permissions는 CloudFormation에 권한 부여자 리소스를 배포합니다.

API 연결 정책 스토어를 생성하려면 Verified Permissions 콘솔에 권한이 AWS 높은 보안 주체에 로그인해야 합니다. 이 사용자는 여러에 리소스를 생성하는 CloudFormation 스택을 배포합니다 AWS 서비스. 이 보안 주체는 Verified Permissions, IAM Lambda 및에서 리소스를 추가하고 수정할 수 있는 권한이 있어야 합니다 API Gateway. 이러한 자격 증명을 조직의 다른 관리자와 공유하지 않는 것이 가장 좋습니다.

Verified Permissions가 생성하는 리소스에 대한 개요는 를 사용하여 프로덕션으로 전환 AWS CloudFormation 섹션을 참조하세요.

속성 기반 액세스 제어(ABAC) 추가

IdP가 있는 일반적인 인증 세션은 ID 및 액세스 토큰을 반환합니다. 애플리케이션 요청에서 이러한 토큰 유형 중 하나를 보유자 토큰으로 API에 전달할 수 있습니다. 정책 스토어를 생성할 때의 선택에 따라 Verified Permissions는 두 가지 유형의 토큰 중 하나를 기대합니다. 두 유형 모두 사용자의 그룹 멤버십에 대한 정보를 포함합니다. 의 토큰 유형에 대한 자세한 내용은 Amazon Cognito 개발자 안내서의 사용자 풀과 함께 토큰 사용을 Amazon Cognito참조하세요. https://docs.aws.amazon.com/cognito/latest/developerguide/amazon-cognito-user-pools-using-tokens-with-identity-providers.html

정책 스토어를 생성한 후 정책을 추가하고 확장할 수 있습니다. 예를 들어 정책을 사용자 풀에 추가할 때 새 그룹을 정책에 추가할 수 있습니다. 정책 스토어는 사용자 풀이 토큰으로 그룹을 제공하는 방식을 이미 알고 있으므로 새 정책이 있는 새 그룹에 대해 일련의 작업을 허용할 수 있습니다.

정책 평가의 그룹 기반 모델을 사용자 속성을 기반으로 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 세분화된 액세스 제어 수준이 높습니다. 이메일 주소, 전화번호 또는 부서 및 관리자와 같은 사용자 지정 속성을 평가하려면 ID 토큰을 평가합니다.

액세스 토큰

액세스 토큰은 OAuth 2.0 범위가 있는 사용자의 권한을 나타냅니다. 권한 부여 계층을 추가하거나 추가 리소스에 대한 요청을 설정하려면 액세스 토큰을 평가합니다. 예를 들어 사용자가 적절한 그룹에 있고 일반적으로 API에 대한 액세스를 승인`PetStore.read`하는 와 같은 범위를 포함하는지 확인할 수 있습니다. 사용자 풀은 런타임 시 리소스 서버 및 토큰 사용자 지정을 사용하여 토큰에 사용자 지정 범위를 추가할 수 있습니다. https://docs.aws.amazon.com/cognito/latest/developerguide/user-pool-lambda-pre-token-generation.html#user-pool-lambda-pre-token-generation-accesstoken

ID 및 액세스 Amazon Cognito 토큰의 클레임을 처리하는 정책의 예는 스키마에 토큰 매핑 및 스키마에 OIDC 토큰 매핑을 참조하세요.

를 사용하여 프로덕션으로 전환 AWS CloudFormation

API 연결 정책 스토어는 API Gateway API에 대한 권한 부여 모델을 빠르게 구축하는 방법입니다. 애플리케이션의 권한 부여 구성 요소에 대한 테스트 환경 역할을 하도록 설계되었습니다. 테스트 정책 스토어를 생성한 후 정책, 스키마 및 Lambda 권한 부여자를 구체화하는 데 시간을 할애합니다.

API의 아키텍처를 조정하려면 정책 스토어 스키마 및 정책을 동등하게 조정해야 할 수 있습니다. API 연결 정책 스토어는 API 아키텍처에서 스키마를 자동으로 업데이트하지 않습니다. 확인 권한은 정책 스토어를 생성할 때만 API를 폴링합니다. API가 충분히 변경되면 새 정책 스토어로 프로세스를 반복해야 할 수 있습니다.

애플리케이션 및 권한 부여 모델을 프로덕션에 배포할 준비가 되면 자동화 프로세스와 함께 개발한 API 연결 정책 스토어를 통합합니다. 가장 좋은 방법은 정책 스토어 스키마와 정책을 다른 AWS 계정 및에 배포할 수 있는 AWS CloudFormation 템플릿으로 내보내는 것입니다 AWS 리전.

API 연결 정책 스토어 프로세스의 결과는 초기 정책 스토어와 Lambda 권한 부여자입니다. Lambda 권한 부여자에는 여러 종속 리소스가 있습니다. Verified Permissions는 이러한 리소스를 자동으로 생성된 CloudFormation 스택에 배포합니다. 프로덕션에 배포하려면 정책 스토어와 Lambda 권한 부여자 리소스를 템플릿으로 수집해야 합니다. API 연결 정책 스토어는 다음 리소스로 구성됩니다.

1. AWS::VerifiedPermissions::PolicyStore: 스키마를 `SchemaDefinition` 객체에 복사합니다. " 문자를 로 이스케이프합니다\".

2. AWS::VerifiedPermissions::IdentitySource: 테스트 정책 스토어의 GetIdentitySource 출력에서 값을 복사하고 필요에 따라 수정합니다.
3. 하나 이상의 AWS::VerifiedPermissions::Policy: 정책 설명을 `Definition` 객체에 복사합니다. " 문자를 로 이스케이프합니다\".
4. AWS::Lambda::Function, AWS::IAM::Role, AWS::IAM::Policy, AWS::ApiGateway::Authorizer, AWS::Lambda::Permission

다음 템플릿은 정책 스토어의 예입니다. 기존 스택의 Lambda 권한 부여자 리소스를이 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
\":{}}}}}}"
                }
            }
        },
        "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": [
                "MyExamplePolicyStore"
            ]
        }
    }
}
```

API 연결 정책 스토어 문제 해결

Amazon Verified Permissions API 연결 정책 스토어를 빌드할 때 일반적인 문제를 진단하고 해결하는 데 도움이 되도록 여기의 정보를 사용하세요.

주제

- 정책을 업데이트했지만 권한 부여 결정이 변경되지 않음
- API에 Lambda 권한 부여자를 연결했지만 권한 부여 요청을 생성하지 않습니다.
- 예상치 못한 권한 부여 결정을 받았으며 권한 부여 로직을 검토하고 싶습니다.
- Lambda 권한 부여자에서 로그를 찾고 싶습니다.
- Lambda 권한 부여자가 존재하지 않음
- API가 프라이빗 VPC에 있으며 권한 부여자를 호출할 수 없음
- 권한 부여 모델에서 추가 사용자 속성을 처리하고 싶습니다.
- 새 작업, 작업 컨텍스트 속성 또는 리소스 속성을 추가하려고 함

정책을 업데이트했지만 권한 부여 결정이 변경되지 않음

기본적으로 Verified Permissions는 120초 동안 권한 부여 결정을 캐싱하도록 Lambda 권한 부여자를 구성합니다. 2분 후에 다시 시도하거나 권한 부여자의 캐시를 비활성화합니다. 자세한 내용은 Amazon API Gateway API Gateway 개발자 안내서의 API 캐싱 활성화를 참조하세요.

API에 Lambda 권한 부여자를 연결했지만 권한 부여 요청을 생성하지 않습니다.

요청 처리를 시작하려면 권한 부여자를 연결한 API 단계를 배포해야 합니다. 자세한 내용은 Amazon API Gateway API Gateway 개발자 안내서의 REST API 배포를 참조하세요.

예상치 못한 권한 부여 결정을 받았으며 권한 부여 로직을 검토하고 싶습니다.

API 연결 정책 스토어 프로세스는 권한 부여자를 위한 Lambda 함수를 생성합니다. Verified Permissions는 권한 부여 결정의 로직을 권한 부여자 함수에 자동으로 빌드합니다. 정책 스토어를 생성한 후 돌아가 함수의 로직을 검토하고 업데이트할 수 있습니다.

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

AWS Lambda 콘솔에서 함수를 찾을 수도 있습니다. 정책 스토어 AWS 리전 의에서 콘솔로 이동하여 접두사가 인 함수 이름을 검색합니다`AVPAuthorizerLambda`. API 연결 정책 스토어를 두 개 이상 생성한 경우 함수의 마지막 수정 시간을 사용하여 이를 정책 스토어 생성과 상호 연관시킵니다.

Lambda 권한 부여자에서 로그를 찾고 싶습니다.

Lambda 함수는 지표를 수집하고 Amazon CloudWatch에 호출 결과를 기록합니다. 로그를 검토하려면 Lambda 콘솔에서 함수를 찾아 모니터 탭을 선택합니다. CloudWatch 로그 보기를 선택하고 로그 그룹의 항목을 검토합니다.

Lambda 함수 로그에 대한 자세한 내용은 AWS Lambda 개발자 안내서의 에서 Amazon CloudWatch Logs 사용을 AWS Lambda 참조하세요.

Lambda 권한 부여자가 존재하지 않음

API 연결 정책 스토어 설정을 완료한 후에는 Lambda 권한 부여자를 API에 연결해야 합니다. API Gateway 콘솔에서 권한 부여자를 찾을 수 없는 경우 정책 스토어의 추가 리소스가 실패했거나 아직 배포되지 않았을 수 있습니다. API 연결 정책 스토어는 이러한 리소스를 CloudFormation 스택에 배포합니다.

Verified Permissions는 생성 프로세스 종료 시 배포 확인 레이블이 있는 링크를 표시합니다. 이미이 화면에서 다른 곳으로 이동한 경우 CloudFormation 콘솔로 이동하여 최근 스택에서 접두사가 붙은 이름을 검색합니다`AVPAuthorizer-<policy store ID>`. CloudFormation은 스택 배포의 출력에 중요한 문제 해결 정보를 제공합니다.

CloudFormation 스택 문제 해결에 대한 도움말은 AWS CloudFormation 사용 설명서의 CloudFormation 문제 해결을 참조하세요.

API가 프라이빗 VPC에 있으며 권한 부여자를 호출할 수 없음

Verified Permissions는 VPC 엔드포인트를 통한 Lambda 권한 부여자에 대한 액세스를 지원하지 않습니다. API와 권한 부여자 역할을 하는 Lambda 함수 간에 네트워크 경로를 열어야 합니다.

권한 부여 모델에서 추가 사용자 속성을 처리하고 싶습니다.

API 연결 정책 스토어 프로세스는 사용자 토큰의 그룹 클레임에서 Verified Permissions 정책을 파생합니다. 추가 사용자 속성을 고려하도록 권한 부여 모델을 업데이트하려면 해당 속성을 정책에 통합합니다.

Amazon Cognito 사용자 풀의 ID 및 액세스 토큰에 있는 여러 클레임을 Verified Permissions 정책 설명에 매핑할 수 있습니다. 예를 들어 대부분의 사용자는 ID 토큰에 `email` 클레임이 있습니다. 자격 증명 소스의 클레임을 정책에 추가하는 방법에 대한 자세한 내용은 스키마에 Amazon Cognito 토큰 매핑 및 스키마에 OIDC 토큰 매핑을 참조하세요.

새 작업, 작업 컨텍스트 속성 또는 리소스 속성을 추가하려고 함

API 연결 정책 스토어와 생성하는 Lambda 권한 부여자는 point-in-time 리소스입니다. 생성 시 API의 상태를 반영합니다. 정책 스토어 스키마는 작업에 컨텍스트 속성이나 기본 `Application` 리소스에 속성 또는 상위 속성을 할당하지 않습니다.

API에 경로 및 메서드와 같은 작업을 추가할 때 새 작업을 인식하도록 정책 스토어를 업데이트해야 합니다. 또한 새 작업에 대한 권한 부여 요청을 처리하도록 Lambda 권한 부여자를 업데이트해야 합니다. 새 정책 스토어로 다시 시작하거나 기존 정책 스토어를 업데이트할 수 있습니다.

기존 정책 스토어를 업데이트하려면 함수를 찾습니다. 자동으로 생성된 함수에서 로직을 검사하고 업데이트하여 새 작업, 속성 또는 컨텍스트를 처리합니다. 그런 다음 스키마를 편집하여 새 작업과 속성을 포함합니다.

정책 스토어 삭제

AWS Management Console 또는를 사용하여 Amazon Verified Permissions 정책 스토어를 삭제할 수 있습니다 AWS CLI. 정책 스토어를 삭제하면 스키마와 정책 스토어의 정책 및 정책 템플릿이 영구적으로 삭제됩니다. 삭제된 정책 스토어와 연결된 모든 정책 스토어 별칭도 삭제됩니다.

삭제 방지 기능은 정책 스토어의 우발적 삭제를 방지합니다. 를 통해 생성된 모든 새 정책 스토어에서 삭제 방지가 활성화됩니다 AWS Management Console. 반대로 API 또는 SDK 직접 호출을 통해 생성된 모든 정책 저장소에 대해서는 비활성화됩니다.

다음과 같은 이유로 정책 스토어를 삭제할 수 있습니다.

- 지정된 리전에서 사용 가능한 정책 스토어의 할당량에 도달했습니다. 자세한 내용은 리소스 할당량 단원을 참조하십시오.
- 다중 테넌트 애플리케이션에서 더 이상 테넌트를 지원하지 않으므로 더 이상 해당 정책 스토어가 필요하지 않습니다.

AWS Management Console

정책 저장소 삭제

1. Verified Permissions 콘솔을 엽니다. 정책 스토어를 선택합니다.
2. 왼쪽의 탐색 창에서 설정을 선택합니다.
3. 이 정책 스토어 삭제를 선택합니다.

4. 텍스트 상자에 delete를 입력하고 삭제를 선택합니다.

> **Note**
>
> 삭제 방지가 활성화된 경우 삭제를 선택하려면 먼저 삭제 방지를 비활성화해야 합니다. 비활성화하려면 삭제 방지 비활성화를 선택합니다.

AWS CLI

정책 저장소 삭제

를 삭제하려는 정책 스토어 ID*PSEXAMPLEabcdefg111111*로 바꾸는 delete-policy-store 작업을 사용하여 정책 스토어를 삭제할 수 있습니다.

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

성공하면 이 명령은 출력을 생성하지 않습니다.

> **Note**
>
> 이 정책 스토어에 대해 삭제 방지가 활성화된 경우 먼저 update-policy-store 작업을 실행하고 삭제 방지를 비활성화해야 합니다.
>
> ```
> aws verifiedpermissions update-policy-store \
>     --deletion-protection "DISABLED" \
>     --policy-store-id PSEXAMPLEabcdefg111111
> ```

Amazon Verified Permissions 정책 스토어 별칭

정책 스토어 별칭은 정책 스토어의 친숙한 이름입니다. 예를 들어 정책 스토어 별칭을 사용하면 `policy-store-alias/example-policy-store` 대신를 사용하여 정책 스토어를 참조할 수 있습니다`PSEXAMPLEabcdefg111111`. 정책 스토어 별칭은 `policyStoreId` 입력 파라미터를 수락하는 모든 Verified Permissions 작업에 사용할 수 있습니다.

`CreatePolicyStoreAlias` API를 사용하거나 `AWS::VerifiedPermissions::PolicyStoreAlias` CloudFormation 리소스를 사용하여 정책 스토어의 정책 스토어 별칭을 생성할 수 있습니다.

Amazon Verified Permissions API는 각 AWS 계정 및 리전의 정책 스토어 별칭을 완벽하게 제어할 수 있습니다. API에는 정책 스토어 별칭(`CreatePolicyStoreAlias`)을 생성하고, 정책 스토어 별칭 이름과 정책 스토어 별칭 ARNs(`GetPolicyStoreAlias`, `ListPolicyStoreAliases`)을 보고, 정책 스토어 별칭()을 삭제하는 작업이 포함되어 있습니다`DeletePolicyStoreAlias`.

주제

- 정책 스토어 별칭의 속성
- Amazon Verified Permissions 정책 스토어 별칭 생성
- Amazon Verified Permissions 정책 스토어 별칭 검색
- Amazon Verified Permissions 정책 스토어 별칭 삭제
- API 작업에서 Amazon Verified Permissions 정책 스토어 별칭 사용
- 정책 스토어 별칭에 대한 액세스 제어

정책 스토어 별칭의 속성

Amazon Verified Permissions에서 정책 스토어 별칭의 작동 방식.

정책 스토어 별칭은 독립 AWS 리소스입니다.

정책 스토어 별칭은 정책 스토어의 속성이 아닙니다. 정책 스토어 별칭에 대해 수행하는 작업은 연결된 정책 스토어에 영향을 주지 않습니다. 연결된 정책 스토어에 영향을 주지 않고 정책 스토어 별칭을 삭제할 수 있습니다. 정책 스토어를 삭제하면 해당 정책 스토어와 연결된 모든 정책 스토어 별칭도 삭제됩니다.

각 정책 스토어 별칭에는 정책 스토어 별칭을 고유하게 식별하는 Amazon 리소스 이름(ARN)이 있습니다. 정책 스토어 별칭을 IAM 정책의 리소스로 지정하는 경우 정책은 연결된 정책 스토어가 아닌 정책 스토어 별칭을 참조합니다.

각 정책 스토어 별칭에는 두 가지 형식이 있습니다.

정책 스토어 별칭을 생성할 때 정책 스토어 별칭 이름을 지정합니다. Amazon Verified Permissions는 정책 스토어 별칭 ARN을 생성합니다.

- 정책 스토어 별칭 ARN은 정책 스토어 별칭을 고유하게 식별하는 Amazon 리소스 이름(ARN)입니다.

```
# Alias ARN
arn:aws:verifiedpermissions:us-east-1:123456789012:policy-store-alias/example-policy-store
```

- AWS 계정 및 리전에서 고유한 정책 스토어 별칭 이름입니다. Amazon Verified Permissions API에서 정책 스토어 별칭 이름에는 항상 접두사가 붙습니다`policy-store-alias/`.

```
# Alias name
policy-store-alias/example-policy-store
```

정책 스토어 별칭이 비밀이 아님

정책 스토어 별칭은 CloudTrail 로그 및 기타 출력의 일반 텍스트로 표시될 수 있습니다. 정책 스토어 별칭 이름에 기밀 또는 민감한 정보를 포함하지 마세요.

각 정책 스토어 별칭은 한 번에 하나의 정책 스토어와 연결됩니다.

정책 스토어 별칭과 연결된 정책 스토어는 동일한 AWS 계정 및 리전에 속해야 합니다. 정책 스토어 별칭을 동일한 AWS 계정 및 리전의 모든 정책 스토어와 연결할 수 있습니다.

예를 들어이 `ListPolicyStoreAliases` 출력은 `example-policy-store` 정책 스토어 별칭이 `policyStoreId` 속성으로 표시되는 정확히 하나의 대상 정책 스토어와 연결되어 있음을 보여줍니다.

```
{
    "aliasName": "policy-store-alias/example-policy-store",
    "policyStoreId": "PSEXAMPLEabcdefg111111",
    "aliasArn": "arn:aws:verifiedpermissions:us-west-2:123456789012:policy-store-alias/example-policy-store",
    "createdAt": "2024-01-15T12:30:00.000000+00:00",
    "state": "Active"
```

```
}
```

여러 별칭을 동일한 정책 스토어에 연결할 수 있습니다.

예를 들어 example-policy-store 및 example-policy-store-2별칭을 동일한 정책 스토어와 연결할 수 있습니다.

```
[
    {
        "aliasName": "policy-store-alias/example-policy-store",
        "policyStoreId": "PSEXAMPLEabcdefg111111",
        "aliasArn": "arn:aws:verifiedpermissions:us-west-2:123456789012:policy-store-
alias/example-policy-store",
        "createdAt": "2024-01-15T12:30:00.000000+00:00",
        "state": "Active"
    },
    {
        "aliasName": "policy-store-alias/example-policy-store-2",
        "policyStoreId": "PSEXAMPLEabcdefg111111",
        "aliasArn": "arn:aws:verifiedpermissions:us-west-2:123456789012:policy-store-
alias/example-policy-store-2",
        "createdAt": "2024-01-16T09:15:00.000000+00:00",
        "state": "Active"
    }
]
```

정책 스토어 별칭은 AWS 계정 및 리전에서 고유해야 합니다.

예를 들어, AWS 계정 및 리전마다 이름이 인 정책 스토어 별칭을 하나만 가질 수 example-policy-store 있습니다. 정책 스토어 별칭은 대/소문자를 구분합니다. 정책 스토어 별칭 이름은 변경할 수 없습니다. 그러나 24시간 예약 기간이 만료된 후 정책 스토어 별칭을 삭제하고 원하는 이름으로 새 정책 스토어 별칭을 생성할 수 있습니다.

다른 리전에서 동일한 이름으로 정책 스토어 별칭을 생성할 수 있습니다. 각 정책 스토어 별칭에는 고유한 ARN이 있습니다. 코드가와 같은 정책 스토어 별칭 이름을 참조policy-store-alias/example-policy-store하는 경우 여러 리전에서 실행할 수 있습니다. 각 리전에서 다른 정책 스토어를 사용합니다.

정책 스토어 별칭에서 소프트 삭제 및 하드 삭제 지원

기본적으로 정책 스토어 별칭이 삭제되면 해당 별칭은 소프트 삭제됩니다. 정책 스토어 별칭 이름은 24시간 동안 예약됩니다. 이 기간 동안 이름이 같은 정책 스토어 별칭을 생성하려고 하

면 요청이 거부됩니다. 이 기간 동안는 PendingDeletion 상태와 함께 정책 스토어 별칭을 GetPolicyStoreAlias 반환합니다. 를 HardDelete로 지정하여 정책 스토어 별칭을 하드 삭제할 수도 있습니다deletionMode. 하드 삭제된 정책 스토어 별칭은 즉시 삭제되고 정책 스토어 별칭 이름은 즉시 재사용할 수 있게 됩니다. 자세한 내용은 Amazon Verified Permissions 정책 스토어 별칭 삭제 단원을 참조하십시오.

별칭을 사용하여 정책 스토어를 식별할 수 있습니다.

정책 스토어 별칭을 사용하여를 수락하는 모든 작업에서 정책 스토어를 식별할 수 있습니다policyStoreId(예: IsAuthorized). 이러한 경우 정책 스토어 별칭 이름 앞에 접두사를 붙여야 합니다policy-store-alias/. 정책 스토어 별칭은 DeletePolicyStore 작업에 대한 정책 스토어를 식별하는 데 사용할 수 없습니다.

정책 스토어 별칭 이름 또는 정책 스토어 별칭 ARN을 사용하여 IAM 정책의 Resource 요소에서 정책 스토어를 식별할 수 없습니다. 정책 스토어 별칭을 통해 참조될 때 정책 스토어에 대한 액세스를 제어하려면 섹션을 참조하세요정책 스토어 별칭에 대한 액세스 제어.

Amazon Verified Permissions 정책 스토어 별칭 생성

표시 이름을 사용하여 정책 스토어 별칭을 생성하여 정책 스토어를 참조할 수 있습니다. 정책 스토어 별칭의 이름은 AWS 계정 및 리전별로 고유해야 합니다. 정책 스토어 별칭은 정책 스토어 별칭과 동일한 리전에서 소유하고 AWS 계정 활성 상태인 정책 스토어에만 연결할 수 있습니다. 정책 스토어 별칭은 자체 ARNs과 IAM 권한이 있는 별도의 리소스입니다.

기본적으로 10개의 정책 스토어 별칭만 동일한 정책 스토어에 연결할 수 있습니다.

> **Note**
>
> CreatePolicyStoreAlias는 멱등성입니다. 기존 정책 스토어 별칭과 일치하는 정책 스토어 별칭 이름과 정책 스토어 ID를 사용하여 CreatePolicyStoreAlias 작업을 호출하면 CreatePolicyStoreAlias 작업이 성공하고 기존 정책 스토어 별칭을 반환합니다. 그러나 기존 정책 스토어 별칭 이름과 다른 정책 스토어 ID로 CreatePolicyStoreAlias 작업을 호출하면 작업이를 반환합니다ConflictException.

AWS CLI

정책 스토어 별칭을 생성하려면

CreatePolicyStoreAlias 작업을 사용하여 정책 스토어 별칭을 생성할 수 있습니다. 다음 예제에서는 이름이 인 정책 스토어 별칭을 생성합니다example-policy-store.

```
$ aws verifiedpermissions create-policy-store-alias \
    --alias-name policy-store-alias/example-policy-store \
    --policy-store-id PSEXAMPLEabcdefg111111
{
    "aliasName": "policy-store-alias/example-policy-store",
    "policyStoreId": "PSEXAMPLEabcdefg111111",
    "aliasArn": "arn:aws:verifiedpermissions:us-west-2:123456789012:policy-store-
alias/example-policy-store",
    "createdAt": "2024-01-15T12:30:00.000000+00:00"
}
```

Amazon Verified Permissions 정책 스토어 별칭 검색

GetPolicyStoreAlias 작업을 사용하여 정책 스토어 별칭에 대한 정보를 검색하여 특정 정책 스토어 별칭에 대한 세부 정보를 가져오거나 AWS 계정 및 리전의 모든 정책 스토어 별칭을 나열하는 ListPolicyStoreAliases 작업을 가져올 수 있습니다.

정책 스토어 별칭 가져오기

GetPolicyStoreAlias 작업을 사용하여 연결된 정책 스토어 ID를 포함하여 특정 정책 스토어 별칭에 대한 세부 정보를 검색합니다.

AWS CLI

정책 스토어 별칭에 대한 세부 정보를 검색하려면

GetPolicyStoreAlias 작업을 사용하여 정책 스토어 별칭을 검색할 수 있습니다. 다음 예시에서는 이름이 인 정책 스토어 별칭의 세부 정보를 검색합니다example-policy-store.

```
$ aws verifiedpermissions get-policy-store-alias \
    --alias-name policy-store-alias/example-policy-store
{
    "aliasName": "policy-store-alias/example-policy-store",
    "policyStoreId": "PSEXAMPLEabcdefg111111",
    "aliasArn": "arn:aws:verifiedpermissions:us-west-2:123456789012:policy-store-
alias/example-policy-store",
    "createdAt": "2024-01-15T12:30:00.000000+00:00",
```

```
    "state": "Active"
}
```

정책 스토어 별칭 나열

`ListPolicyStoreAliases` 작업을 사용하여 AWS 계정 및 리전의 모든 정책 스토어 별칭을 나열합니다. `filter` 파라미터를 사용하여 특정 정책 스토어와 연결된 정책 스토어 별칭만 나열할 수 있습니다.

AWS CLI

모든 정책 스토어 별칭을 나열하려면

ListPolicyStoreAliases 작업을 사용하여 정책 스토어 별칭을 나열할 수 있습니다. 다음 예제에서는 us-west-2 리전에서 123456789012 AWS 계정 가 소유한 모든 정책 스토어 별칭을 나열합니다.

```
$ aws verifiedpermissions list-policy-store-aliases
{
    "policyStoreAliases": [
        {
            "aliasName": "policy-store-alias/example-policy-store",
            "policyStoreId": "PSEXAMPLEabcdefg111111",
            "aliasArn": "arn:aws:verifiedpermissions:us-west-2:123456789012:policy-
store-alias/example-policy-store",
            "createdAt": "2024-01-15T12:30:00.000000+00:00",
            "state": "Active"
        },
        {
            "aliasName": "policy-store-alias/example-policy-store-2",
            "policyStoreId": "PSEXAMPLEabcdefg111111",
            "aliasArn": "arn:aws:verifiedpermissions:us-west-2:123456789012:policy-
store-alias/example-policy-store-2",
            "createdAt": "2024-01-16T09:15:00.000000+00:00",
            "state": "Active"
        },
        {
            "aliasName": "policy-store-alias/example-policy-store-3",
            "policyStoreId": "PSEXAMPLEabcdefg222222",
            "aliasArn": "arn:aws:verifiedpermissions:us-west-2:123456789012:policy-
store-alias/example-policy-store-3",
            "createdAt": "2024-01-17T14:45:00.000000+00:00",
            "state": "Active"
```

```
        }
    ]
}
```

특정 정책 스토어의 정책 스토어 별칭을 나열하려면

`filter` 파라미터를 사용하여 특정 정책 스토어와 연결된 별칭만 나열합니다.

```
$ aws verifiedpermissions list-policy-store-aliases \
    --filter '{"policyStoreId": "PSEXAMPLEabcdefg111111"}'
{
    "policyStoreAliases": [
        {
            "aliasName": "policy-store-alias/example-policy-store",
            "policyStoreId": "PSEXAMPLEabcdefg111111",
            "aliasArn": "arn:aws:verifiedpermissions:us-west-2:123456789012:policy-
store-alias/example-policy-store",
            "createdAt": "2024-01-15T12:30:00.000000+00:00",
            "state": "Active"
        },
        {
            "aliasName": "policy-store-alias/example-policy-store-2",
            "policyStoreId": "PSEXAMPLEabcdefg111111",
            "aliasArn": "arn:aws:verifiedpermissions:us-west-2:123456789012:policy-
store-alias/example-policy-store-2",
            "createdAt": "2024-01-16T09:15:00.000000+00:00",
            "state": "Active"
        }
    ]
}
```

Amazon Verified Permissions 정책 스토어 별칭 삭제

정책 스토어 별칭이 더 이상 필요하지 않은 경우 삭제할 수 있습니다. 정책 스토어 별칭을 삭제해도 연결된 정책 스토어에는 영향을 주지 않습니다. 정책 스토어를 삭제하면 해당 정책 스토어와 연결된 모든 정책 스토어 별칭이 삭제됩니다.

Amazon Verified Permissions는 정책 스토어 별칭에 대해 두 가지 삭제 모드를 지원합니다.

- 소프트 삭제(기본값): 정책 스토어 별칭이 `PendingDeletion` 상태가 됩니다. 정책 스토어 별칭 이름은 24시간 동안 예약되어 있으며이 기간 동안에는 재사용할 수 없습니다. 이 기간 동안는

PendingDeletion 상태와 함께 정책 스토어 별칭을 GetPolicyStoreAlias 반환합니다. 이는를 지정하지 않거나를 지정할 deletionMode때의 기본 동작입니다SoftDelete.

- 하드 삭제: 정책 스토어 별칭이 즉시 삭제됩니다. 정책 스토어 별칭 이름은 즉시 재사용할 수 있습니다. 하드 삭제를 수행하려면를 HardDelete로 지정합니다deletionMode.

정책 스토어 별칭 소프트 삭제

기본적으로 정책 스토어 별칭을 삭제하면 소프트 삭제가 수행됩니다. 정책 스토어 별칭은 PendingDeletion 상태로 전환되고 정책 스토어 별칭 이름은 24시간 동안 예약됩니다.

AWS CLI

정책 스토어 별칭을 소프트 삭제하려면

DeletePolicyStoreAlias 작업을 사용하여 정책 스토어 별칭을 소프트 삭제할 수 있습니다. 다음 예제 소프트는 이름이 인 정책 스토어 별칭을 삭제합니다example-policy-store.

```
$ aws verifiedpermissions delete-policy-store-alias \
    --alias-name policy-store-alias/example-policy-store
```

소프트 삭제 모드를 명시적으로 지정할 수도 있습니다.

```
$ aws verifiedpermissions delete-policy-store-alias \
    --alias-name policy-store-alias/example-policy-store \
    --deletion-mode SoftDelete
```

정책 스토어 별칭 하드 삭제

정책 스토어 별칭을 즉시 삭제하려면를 HardDelete로 지정합니다deletionMode. 하드 삭제된 정책 스토어 별칭은 PendingDeletion 상태가 되지 않으며 정책 스토어 별칭 이름을 즉시 재사용할 수 있게 됩니다.

이전에 소프트 삭제된 정책 스토어 별칭을 하드 삭제하면 정책 스토어 별칭이 즉시 삭제됩니다.

> **Important**
>
> Amazon Verified Permissions는 최종적으로 일관됩니다. 정책 스토어 별칭을 하드 삭제하고 즉시 다시 생성하여 다른 정책 스토어를 가리키는 경우 정책 스토어 별칭을 참조하는 요청은

잠시 동안 이전에 연결된 정책 스토어로 계속 확인될 수 있습니다. 예기치 않은 권한 부여 결과를 방지하려면 이름이 같은 정책 스토어 별칭을 다시 생성하기 전에 삭제가 전파될 때까지 기다리세요.

AWS CLI

정책 스토어 별칭을 하드 삭제하려면

`deletion-mode` 파라미터가 로 설정된 DeletePolicyStoreAlias 작업을 사용하여 정책 스토어 별칭을 하드 삭제할 수 있습니다`HardDelete`. 다음 예제에서는 이름이 인 정책 스토어 별칭을 즉시 삭제합니다`example-policy-store`.

```
$ aws verifiedpermissions delete-policy-store-alias \
    --alias-name policy-store-alias/example-policy-store \
    --deletion-mode HardDelete
```

API 작업에서 Amazon Verified Permissions 정책 스토어 별칭 사용

IsAuthorized, IsAuthorizedWithTokenIsAuthorizedWithToken, GetPolicyStore와 같은 `policyStoreId` 파라미터를 수락하는 모든 Amazon Verified Permissions 작업은 정책 스토어 ID 대신 정책 스토어 별칭 이름을 수락할 수 있습니다.

Important

정책 스토어 별칭을 `policyStoreId` 파라미터 값으로 사용하는 경우 `policy-store-alias/` 접두사를 포함해야 합니다. 예를 들어가 `policy-store-alias/example-policy-store`아닌를 사용합니다`example-policy-store`.

작업에서 정책 스토어 별칭 사용

다음 `IsAuthorized` 명령은 이름이 인 정책 스토어 별칭`example-policy-store`을 사용하여 정책 스토어를 식별합니다.

AWS CLI

```
$ aws verifiedpermissions is-authorized \
    --policy-store-id policy-store-alias/example-policy-store \
```

```
    --principal entityType=User,entityId=alice \
    --action actionType=Action,actionId=view \
    --resource entityType=Photo,entityId=photo123
```

> **Note**
>
> DeletePolicyStore 작업에는 `policyStoreId` 필드 대신 정책 스토어 별칭을 사용할 수 없습니다.

에서 정책 스토어 별칭 사용 AWS 리전

별칭의 가장 강력한 용도 중 하나는 여러 AWS 리전에서 실행되는 애플리케이션에서 사용하는 것입니다. 예를 들어 각 리전에 서로 다른 정책 스토어를 사용하는 글로벌 애플리케이션이 있을 수 있습니다.

- us-east-1에서는를 사용하려고 합니다`PSEXAMPLEabcdefg111111`.
- eu-west-1에서는를 사용하려고 합니다`PSEXAMPLEabcdefg222222`.

각 리전에서 다른 버전의 애플리케이션을 생성하거나 사전 또는 스위치 문을 사용하여 각 리전에 적합한 정책 스토어를 선택할 수 있습니다. 그러나 각 리전에서 동일한 정책 스토어 별칭 이름으로 정책 스토어 별칭을 생성하는 것이 훨씬 쉽습니다. 정책 스토어 별칭 이름은 대/소문자를 구분합니다.

AWS CLI

```
$ aws --region us-east-1 verifiedpermissions create-policy-store-alias \
    --alias-name policy-store-alias/my-app \
    --policy-store-id PSEXAMPLEabcdefg111111

$ aws --region eu-west-1 verifiedpermissions create-policy-store-alias \
    --alias-name policy-store-alias/my-app \
    --policy-store-id PSEXAMPLEabcdefg222222
```

그런 다음 코드에서 정책 스토어 별칭을 사용합니다. 코드가 각 리전에서 실행되면 정책 스토어 별칭은 해당 리전의 연결된 정책 스토어를 참조합니다.

AWS CLI

```
$ aws verifiedpermissions is-authorized \
```

```
    --policy-store-id policy-store-alias/my-app \
    --principal entityType=User,entityId=alice \
    --action actionType=Action,actionId=view \
    --resource entityType=Photo,entityId=photo123
```

그러나 정책 스토어 별칭이 삭제될 위험이 있습니다. 이 경우 애플리케이션의 정책 스토어 별칭 이름 사용 시도가 실패하고 정책 스토어 별칭을 다시 생성하거나 업데이트해야 할 수 있습니다. 이 위험을 완화하려면 보안 주체에게 애플리케이션에서 사용하는 정책 스토어 별칭을 관리할 수 있는 권한을 부여해야 합니다.

정책 스토어 별칭은 트래픽 제어 메커니즘이 아닙니다.

정책 스토어 별칭은 정책 스토어의 안정적이고 친숙한 이름입니다. 정책 스토어 간에 권한 부여 트래픽을 이동, 분할 또는 가중치를 부여하는 메커니즘은 아닙니다. Lambda 별칭과 같이 리소스 버전 간에 트래픽을 라우팅하는 기능에 익숙한 경우 정책 스토어 별칭은 동일한 방식으로 작동하지 않습니다. 설계상 정책 스토어 별칭은 AWS KMS 키 별칭에 더 가깝습니다. 즉, 앞에 있는 라우팅 계층이 아니라 리소스에 대한 내구성 있는 이름을 제공합니다.

이러한 이유로 의도적으로 `UpdatePolicyStoreAlias` 작업을 제공하지 않습니다. 정책 스토어 별칭이 가리키는 정책 스토어를 변경하려면 정책 스토어 별칭을 삭제하고 이름이 같고 다른 정책 스토어를 대상으로 하는 새 정책 스토어 별칭을 생성합니다. 이는 원자성 작업이 아니며 트래픽 제어 메커니즘이 보장하지 않습니다.

정책 스토어 별칭이 확인되는 정책 스토어를 변경하면 변경 사항이 동일한 시점에 모든 곳에서 적용되지 않습니다.

- 업데이트된 매핑은 컨트롤 플레인에서 권한 부여 요청을 평가하는 데이터 플레인으로 전파되어야 합니다. 매핑은 즉시 전파되지 않습니다.
- 정책 스토어 별칭 확인은 최종적으로 일관되고 캐시될 수 있으므로 변경 후 전환 기간이 존재합니다. 이 기간 동안 이전에 연결된 정책 스토어는 일부 요청을 처리하고 새로 연결된 정책 스토어는 다른 요청을 처리합니다. 이 창의 길이는 불확정입니다.

이 기간 동안 애플리케이션은 두 정책 스토어 중 하나로부터 권한 부여 결정을 받을 수 있습니다. 정책 스토어마다 정책, 엔터티 및 스키마가 다를 수 있으므로 동일한 요청은 어떤 정책 스토어가 이를 제공하는지에 따라 다른 결정을 내릴 수 있습니다. 따라서 정책 스토어 별칭은 정책 스토어 간에 원자성 전환을 제공할 수 없으며 배포 또는 트래픽 관리 전략을 대체하지 않습니다.

⚠ 별칭을 전환 메커니즘으로 사용하지 마세요.

한 정책 스토어에서 다른 정책 스토어로 권한 부여 트래픽을 전환하기 위해 정책 스토어 별칭을 다시 가리키는 상위 수준의 infrastructure-as-code 또는 배포 프리미티브를 구축하지 마세요. 변경 사항은 원자성이 아니므로 불확정 기간 동안 두 정책 스토어에 대해 동시에 요청을 승인할 수 있습니다. 이로 인해 권한 부여 결정이 일관되지 않을 수 있습니다. 정책 스토어를 안전하게 전환하려면 섹션을 참조하세요가동 중지 없는 정책 스토어 변경 수행.

가동 중지 없는 정책 스토어 변경 수행

정책 스토어 별칭은 원자성 전환을 제공하지 않으므로(참조정책 스토어 별칭은 트래픽 제어 메커니즘이 아닙니다.) 정책 스토어 별칭을 다시 지정하여 한 정책 스토어에서 다른 정책 스토어로 마이그레이션하지 않는 것이 좋습니다. 대신 새 정책 스토어를 검증하고 필요한 경우 즉시 롤백할 수 있도록 애플리케이션 내에서 마이그레이션을 제어합니다. 다음 접근 방식을 사용하면 가동 중지 없이 정책 스토어를 전환할 수 있습니다.

1. 새 정책 저장소를 생성하고 고유한 정책 저장소 별칭을 지정하여 현재 정책 저장소와 새 정책 저장소에 각각 고유하고 안정적인 이름이 지정되도록 합니다. 예를 들어 현재 정책 스토어를 `policy-store-alias/example-policy-store` 계속 가리키고 새 정책 스토어에 `policy-store-alias/example-policy-store-2` 대해를 생성합니다. 전환을 수행하기 위해 단일 정책 스토어 별칭을 재사용하거나 다시 지정하지 마십시오.
2. 정책, 스키마 및 기타 구성을 새 정책 스토어에 복제하고 현재 정책 스토어와 병렬로 실행합니다.
3. 권한 부여 결정에 권한이 있는 정책 스토어 별칭을 결정하는 구성 값 또는 기능 플래그를 애플리케이션에 추가합니다(예: AppConfig 사용). 트래픽을 전환하기 위해 별칭 자체를 변경하는 데 의존하지 마십시오.
4. 섀도우 모드에서 새 정책 스토어를 실행합니다. 두 정책 스토어에 동일한 `IsAuthorized` 또는 `IsAuthorizedWithToken` 요청을 보내고 결정을 비교합니다. 새 정책 스토어가 예상한 결정을 반환할 때까지 불일치를 기록하고 조사합니다.
5. 기능 플래그를 사용하여 애플리케이션 호스트 또는 사용자 세그먼트 간에 단계별 롤아웃에서 권한 부여 결정을 새 정책 스토어 별칭으로 점진적으로 이동합니다. 진행하면서 권한 부여 결정 및 오류 발생률을 모니터링합니다.
6. 문제가 감지되면 기능 플래그를 사용하여 즉시 원래 정책 스토어 별칭으로 다시 전환합니다. 애플리케이션이 스위치를 제어하기 때문에 롤백은 즉시 이루어지며 별칭 전파에 의존하지 않습니다.
7. 새 정책 스토어가 완전히 검증되고 모든 트래픽을 처리한 후 이전 정책 스토어와 해당 정책 스토어 별칭을 폐기합니다.

이 패턴은 별칭 전파 대신 애플리케이션에 각 요청을 처리하는 정책 스토어를 제어합니다. 이러한 제어는 마이그레이션을 안전하고 되돌릴 수 있게 하며 정책 스토어 별칭을 다시 가리키는 전환 기간을 방지합니다.

정책 스토어 별칭에 대한 액세스 제어

정책 스토어 별칭을 관리하는 보안 주체는 해당 정책 스토어 별칭과 상호 작용할 수 있는 권한이 있어야 하며, 일부 작업의 경우 정책 스토어 별칭이 연결된 정책 스토어가 있어야 합니다. 정책을 사용하여 IAM 이러한 권한을 제공할 수 있습니다.

다음 섹션에서는 정책 스토어 별칭을 생성하고 관리하는 데 필요한 권한을 설명합니다.

verifiedpermissions:CreatePolicyStoreAlias

정책 스토어 별칭을 생성하려면 보안 주체는 정책 스토어 별칭과 연결된 정책 스토어 모두에 대해 다음 권한이 필요합니다.

- `verifiedpermissions:CreatePolicyStoreAlias` 정책 스토어 별칭에 대한 입니다. 정책 스토어 IAM 별칭을 생성할 수 있는 보안 주체에 연결된 정책에이 권한을 제공합니다.

  다음 예제 정책 문은 `Resource` 요소의 특정 정책 스토어 별칭을 지정합니다. 그러나 여러 정책 스토어 별칭 ARNs을 나열하거나와 같은 정책 스토어 별칭 패턴을 지정할 수 있습니다`"sample*"`. 보안 주체가 AWS 계정 및 리전에서 정책 스토어 별칭을 생성할 수 `"*"` 있도록 `Resource` 값을 지정할 수도 있습니다.

  ```
  {
    "Sid": "IAMPolicyForCreateAlias",
    "Effect": "Allow",
    "Action": "verifiedpermissions:CreatePolicyStoreAlias",
    "Resource": "arn:aws:verifiedpermissions:us-east-1:123456789012:policy-store-alias/
  example-policy-store"
  }
  ```

- `verifiedpermissions:CreatePolicyStoreAlias` 연결된 정책 스토어에 대한 입니다. 이 권한은 IAM 정책에서 제공해야 합니다.

  ```
  {
    "Sid": "PolicyStorePermissionForAlias",
    "Effect": "Allow",
    "Action": "verifiedpermissions:CreatePolicyStoreAlias",
  ```

```
  "Resource": "arn:aws:verifiedpermissions::123456789012:policy-store/
PSEXAMPLEabcdefg111111"
}
```

verifiedpermissions:GetPolicyStoreAlias

특정 정책 스토어 별칭에 대한 세부 정보를 가져오려면 보안 주체에게 IAM 정책의 정책 스토어 별칭에 대한 `verifiedpermissions:GetPolicyStoreAlias` 권한이 있어야 합니다.

다음 예제 정책 문은 보안 주체에게 특정 정책 스토어 별칭을 가져올 수 있는 권한을 부여합니다.

```
{
  "Sid": "IAMPolicyForGetAlias",
  "Effect": "Allow",
  "Action": "verifiedpermissions:GetPolicyStoreAlias",
  "Resource": "arn:aws:verifiedpermissions:us-east-1:123456789012:policy-store-alias/
example-policy-store"
}
```

verifiedpermissions:ListPolicyStoreAliases

AWS 계정 및 리전에서 정책 스토어 별칭을 나열하려면 보안 주체에게 IAM 정책에 대한 `verifiedpermissions:ListPolicyStoreAliases` 권한이 있어야 합니다. 이 정책은 특정 정책 스토어 또는 정책 스토어 별칭 리소스와 관련이 없으므로 정책의 리소스 요소 값은 여야 합니다"*".

예를 들어 다음 IAM 정책 문은 보안 주체에게의 모든 정책 스토어 별칭을 나열할 수 있는 권한을 부여합니다 AWS 계정.

```
{
  "Sid": "IAMPolicyForListingAliases",
  "Effect": "Allow",
  "Action": "verifiedpermissions:ListPolicyStoreAliases",
  "Resource": "*"
}
```

verifiedpermissions:DeletePolicyStoreAlias

정책 스토어 별칭을 삭제하려면 보안 주체에게 정책 스토어 별칭에 대한 권한만 필요합니다.

> **Note**
>
> 정책 스토어 별칭을 참조하는 애플리케이션은 오류를 수신하지만 정책 스토어 별칭을 삭제해도 연결된 정책 스토어에는 영향을 미치지 않습니다. 실수로 정책 스토어 별칭을 삭제한 경우 24시간 예약 기간 후에 다시 생성할 수 있습니다.

보안 주체는 정책 스토어 별칭에 대한 `verifiedpermissions:DeletePolicyStoreAlias` 권한이 필요합니다. 정책 스토어 IAM 별칭을 삭제할 수 있는 보안 주체에 연결된 정책에이 권한을 제공합니다.

다음 예제 정책 문은 `Resource` 요소의 정책 스토어 별칭을 지정합니다. 그러나 여러 정책 스토어 별칭 ARNs을 나열하거나와 같은 정책 스토어 별칭 패턴을 지정할 수 있습니다`"sample*"`. 보안 주체가 AWS 계정 및 리전에서 정책 스토어 별칭을 삭제할 수 `"*"` 있도록 `Resource` 값을 지정할 수도 있습니다.

```
{
  "Sid": "IAMPolicyForDeleteAlias",
  "Effect": "Allow",
  "Action": "verifiedpermissions:DeletePolicyStoreAlias",
  "Resource": "arn:aws:verifiedpermissions:us-east-1:123456789012:policy-store-alias/
example-policy-store"
}
```

정책 스토어 별칭 권한 제한

정책 스토어 별칭을 사용하여 `policyStoreId` 필드를 입력으로 수락하는 모든 작업에서 정책 스토어를 참조할 수 있습니다. 이렇게 하면 Amazon Verified Permissions는 정책 스토어 별칭에 `verifiedpermissions:GetPolicyStoreAlias` 대해 권한을 부여하고 연결된 정책 스토어에 대해 요청된 작업을 승인합니다.

예를 들어 정책 스토어 별칭을 사용하여 `IsAuthorized` 작업을 수행하는 경우 보안 주체는 다음 둘 다 필요합니다.

- `verifiedpermissions:GetPolicyStoreAlias` 정책 스토어 별칭에 대한 권한
- `verifiedpermissions:IsAuthorized` 연결된 정책 스토어에 대한 권한

다음 예제 정책은 특정 정책 스토어 별칭을 `IsAuthorized` 사용하여를 호출할 수 있는 권한을 부여합니다.

```
{
  "Sid": "IAMPolicyForAliasUsage",
  "Effect": "Allow",
  "Action": "verifiedpermissions:GetPolicyStoreAlias",
  "Resource": "arn:aws:verifiedpermissions:us-east-1:123456789012:policy-store-alias/
example-policy-store"
},
{
  "Sid": "IAMPolicyForPolicyStoreOperation",
  "Effect": "Allow",
  "Action": "verifiedpermissions:IsAuthorized",
  "Resource": "arn:aws:verifiedpermissions::123456789012:policy-store/
PSEXAMPLEabcdefg111111"
}
```

보안 주체가 사용할 수 있는 정책 스토어 별칭을 제한하려면 `verifiedpermissions:GetPolicyStoreAlias` 권한을 제한합니다. 예를 들어 다음 정책은 보안 주체가 로 시작하는 별칭을 제외한 모든 정책 스토어 별칭을 사용하도록 허용합니다`Restricted`.

```
{
  "Sid": "IAMPolicyForAliasAllow",
  "Effect": "Allow",
  "Action": "verifiedpermissions:GetPolicyStoreAlias",
  "Resource": "arn:aws:verifiedpermissions:us-east-1:123456789012:policy-store-alias/*"
},
{
  "Sid": "IAMPolicyForAliasDeny",
  "Effect": "Deny",
  "Action": "verifiedpermissions:GetPolicyStoreAlias",
  "Resource": "arn:aws:verifiedpermissions:us-east-1:123456789012:policy-store-alias/
Restricted*"
}
```

Amazon Verified Permissions 정책 스토어 스키마

스키마는 애플리케이션이 지원하는 엔터티 유형의 구조와 애플리케이션이 권한 부여 요청에서 제공할 수 있는 작업을 선언한 것입니다. Verified Permissions와 Cedar가 스키마를 처리하는 방법의 차이점은 섹션을 참조하세요스키마 지원.

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. 정책 검증을 활성화하면 모든 새 정책이 스키마를 준수해야 합니다.

AWS Management Console

스키마를 생성하려면

1. Verified Permissions 콘솔을 엽니다. 정책 스토어를 선택합니다.
2. 왼쪽에 있는 탐색 창에서 스키마를 선택합니다.
3. 스키마 생성을 선택합니다.

AWS CLI

AWS CLI를 사용하여 새 스키마를 제출하거나 기존 스키마를 덮어쓰려면

다음 예제와 유사한 AWS CLI 명령을 실행하여 정책 저장소를 생성할 수 있습니다.

다음과 같은 Cedar 콘텐츠가 포함된 스키마를 생각해 보십시오.

```
{
    "MySampleNamespace": {
        "actions": {
            "remoteAccess": {
                "appliesTo": {
                    "principalTypes": [ "Employee" ]
```

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

먼저 JSON을 한 줄 문자열로 이스케이프하고 앞에 해당 데이터 유형 `cedarJson`을 선언해야 합니다. 다음 예제에서는 JSON 스키마의 이스케이프된 버전을 포함하는 `schema.json` 파일의 다음 내용을 사용합니다.

> **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 PSEXAMPLEabcdefg111111
{
    "policyStoreId": "PSEXAMPLEabcdefg111111",
    "namespaces": [
        "MySampleNamespace"
```

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

AWS SDKs

PutSchema API를 사용하여 정책 스토어를 생성할 수 있습니다. 자세한 내용은 Amazon Verified Permissions API 참조 가이드의 PutSchema를 참조하세요.

정책 스토어 스키마 편집

Amazon Verified Permissions 콘솔에서 스키마를 선택하면 스키마를 구성하는 개체 유형 및 작업이 표시됩니다. 시각적 모드 또는 JSON 모드에서 스키마 편집을 볼 수 있습니다. 시각적 모드를 사용하면 다양한 마법사를 사용하여 새 유형과 작업을 추가하여 스키마를 업데이트할 수 있습니다. JSON 모드를 사용하면 JSON 편집기에서 직접 스키마의 JSON 코드 업데이트를 시작할 수 있습니다.

Visual Mode

시각적 스키마 편집기는 스키마의 개체 간 관계를 보여주는 일련의 다이어그램으로 시작합니다. 확장을 선택하여 다이어그램 보기를 극대화합니다. 두 가지 다이어그램을 사용할 수 있습니다.

- 작업 다이어그램 - 작업 다이어그램 보기에는 정책 스토어에서 구성한 보안 주체 유형, 수행할 수 있는 작업 및 작업을 수행할 수 있는 리소스가 나열됩니다. 개체 사이의 줄은 보안 주체가 리소스에 대해 작업을 수행할 수 있도록 허용하는 정책을 생성하는 기능을 나타냅니다. 작업 다이어그램이 두 개체 간의 관계를 나타내지 않는 경우 정책에서 허용하거나 거부하려면 먼저 두 개체 간의 관계를 생성해야 합니다. 속성을 보려면 개체를 선택하고 전체 세부 정보를 보려면 드릴다운합니다. 고유한 연결만 있는 뷰에서 개체를 보려면 이 [작업 | 리소스 유형 | 보안 주체 유형]을 기준으로 필터링을 선택합니다.
- 개체 유형 다이어그램 - 개체 유형 다이어그램은 보안 주체와 리소스 간의 관계에 중점을 둡니다. 스키마의 복잡한 중첩 상위 관계를 이해하려면이 다이어그램을 검토하세요. 개체 위에 마우스를 올려 놓으면 개체가 있는 상위 관계로 드릴다운할 수 있습니다.

  다이어그램 아래에는 스키마의 개체 유형 및 작업의 목록 보기가 있습니다. 목록 보기는 특정 작업 또는 엔터티 유형의 세부 정보를 즉시 보려는 경우에 유용합니다. 세부 정보를 볼 개체를 선택합니다.

비주얼 모드에서 Verified Permissions 스키마를 편집하려면

1. Verified Permissions 콘솔을 엽니다. 정책 스토어를 선택합니다.
2. 왼쪽 탐색 창에서 스키마를 선택합니다.
3. 비주얼 모드를 선택합니다. 개체 관계 다이어그램을 검토하고 스키마에 적용하려는 변경 사항을 계획합니다. 선택적으로 한 엔터티를 기준으로 필터링하여 다른 엔터티에 대한 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. 속성 이름을 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.

JSON mode

업데이트하는 동안 JSON 편집기는 JSON 구문을 기준으로 코드를 검증하고 편집 시 오류와 경고를 식별하므로 문제를 더 쉽게 찾을 수 있습니다. 또한 JSON 형식에 대해 걱정할 필요가 없습니다. 업데이트한 후 JSON 형식을 선택하면 예상되는 JSON 형식과 일치하도록 형식이 업데이트됩니다.

JSON 모드에서 Verified Permissions 스키마를 편집하려면

1. Verified Permissions 콘솔을 엽니다. 정책 스토어를 선택합니다.
2. 왼쪽 탐색 창에서 스키마를 선택합니다.
3. JSON 모드를 선택한 다음 스키마 편집을 선택합니다.
4. 콘텐츠 필드에 JSON 스키마의 콘텐츠를 입력합니다. 모든 구문 오류를 해결할 때까지는 스키마 업데이트 사항을 저장할 수 없습니다. JSON 형식 지정을 선택하여 스키마 JSON 구문의 형식을 권장 간격과 들여쓰기로 지정할 수 있습니다.
5. 변경 사항 저장을 선택합니다.

Amazon Verified Permissions 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. 검증에 실패하면 Verified Permissions가 요청 시도를 거부합니다. 따라서 애플리케이션을 개발하는 동안 검증을 끄고 테스트를 위해 켠 다음 애플리케이션이 프로덕션 상태일 때 켜는 것이 좋습니다.

AWS Management Console

정책 스토어의 정책 검증 모드를 설정하려면

1. Verified Permissions 콘솔을 엽니다. 정책 스토어를 선택합니다.
2. 설정을 선택합니다.
3. 정책 검증 모드 섹션에서 수정을 선택합니다.
4. 다음 중 하나를 수행합니다.
   - 정책 검증을 활성화하고 모든 정책 변경 사항을 스키마에 대해 검증하도록 하려면 엄격(권장) 라디오 버튼을 선택합니다.
   - 정책 변경에 대한 정책 검증을 해제하려면 끄기 라디오 버튼을 선택합니다. `confirm`을 입력하여 정책 업데이트가 더 이상 스키마에 대해 검증되지 않는지 확인합니다.
5. 변경 사항 저장을 선택합니다.

AWS CLI

정책 스토어의 검증 모드를 설정하려면

UpdatePolicyStore 작업을 사용하고 ValidationSettings 매개 변수에 다른 값을 지정하여 정책 스토어의 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 정책 템플릿을 참조하세요.

Amazon Verified Permissions 정책

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

선택적으로 정책에 정책 이름을 할당할 수 있습니다. 정책 이름은 정책 스토어 내의 모든 정책에 대해 고유해야 하며 접두사가 붙어야 합니다name/. policyId 파라미터를 수락하는 컨트롤 플레인 작업에서 정책 ID 대신 정책 이름을 사용할 수 있습니다. 다음 예제에서는 정책 이름을 사용하여를 사용하여 정책을 검색합니다GetPolicy.

```
$ aws verifiedpermissions get-policy \
    --policy-id name/example-policy \
    --policy-store-id PSEXAMPLEabcdefg111111
```

> **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 보안 주체 및 리소스 식별자에 대해 범용 고유 식별자(UUIDs)를 사용합니다.
>
>   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 자동으로 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.
>
>   엔터티에 UUID를 사용할 때는 주석 지정자(//)와 엔터티의 '친숙한' 이름을 사용하는 것이 좋습니다. 이렇게 하면 정책을 더 쉽게 이해할 수 있습니다. 예: principal == Role::"a1b2c3d4-e5f6-a1b2-c3d4-EXAMPLE11111, // 관리자
> - 개인 식별 정보, 기밀 정보 또는 민감한 정보를 보안 주체 또는 리소스에 대한 고유 식별자의 일부로 포함하지 마십시오. 이러한 식별자는 AWS CloudTrail 추적에서 공유되는 로그 항목에 포함됩니다.

주제

- Amazon Verified Permissions 정적 정책 생성
- Amazon Verified Permissions 정적 정책 편집
- 컨텍스트 추가
- Amazon Verified Permissions 테스트 벤치 사용
- 리소스당 정책 크기
- Amazon Verified Permissions 정책 예제

Amazon Verified Permissions 정적 정책 생성

보안 주체가 애플리케이션의 지정된 리소스에 대해 지정된 작업을 수행하도록 허용하거나 금지하는 정적 정책을 생성할 수 있습니다. 정적 정책에는 `principal` 및에 대한 특정 값이 포함되어 `resource` 있으며 권한 부여 결정에 사용할 준비가 되어 있습니다.

AWS Management Console

정적 정책을 생성하려면

1. Verified Permissions 콘솔을 엽니다. 정책 스토어를 선택합니다.
2. 왼쪽의 탐색 창에서 정책을 선택합니다.
3. 정책 생성과 정적 정책 생성을 차례로 선택합니다.

> **Note**
>
> 사용하려는 정책 설명이 있는 경우 8단계로 건너뛰고 정책을 다음 페이지의 정책 섹션에 붙여 넣습니다.

4. 정책 효과 섹션에서 요청이 정책과 일치하는 경우 정책을 허용하거나 금지하도록 선택합니다. 허용을 선택하면 정책은 보안 주체가 리소스에 대한 작업을 수행하도록 허용합니다. 반대로 금지를 선택하면 정책은 보안 주체가 리소스에 대한 작업을 수행하도록 허용하지 않습니다.
5. 보안 주체 범위 필드에서 정책을 적용할 보안 주체의 범위를 선택합니다.
   - 정책을 특정 보안 주체에 적용하려면 특정 보안 주체를 선택합니다. 정책에 지정된 작업을 수행하도록 허용되거나 금지될 보안 주체의 엔터티 유형 및 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 엔터티 유형과 식별자를 지정합니다.
   - 리소스 그룹에 정책을 적용하려면 리소스 그룹을 선택합니다. 리소스 그룹 필드에 리소스 그룹 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 Cedar 기본 정책 구성을 참조하세요.
10. 세부 정보 섹션에 정책에 대한 선택적 설명을 입력합니다.
11. 정책 생성을 선택합니다.

AWS CLI

정적 정책을 생성하려면

CreatePolicy 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 PSEXAMPLEabcdefg111111
{
"Arn": "arn:aws:verifiedpermissions::123456789012:policy/PSEXAMPLEabcdefg111111/
SPEXAMPLEabcdefg111111",
    "createdDate": "2023-05-16T20:33:01.730817+00:00",
    "lastUpdatedDate": "2023-05-16T20:33:01.730817+00:00",
    "policyId": "SPEXAMPLEabcdefg111111",
    "policyStoreId": "PSEXAMPLEabcdefg111111",
```

```
    "policyType": "STATIC"
}
```

정책 이름으로 정책을 생성하려면

정책을 생성할 때 선택적으로 정책 이름을 지정할 수 있습니다. 이름은 정책 스토어 내의 모든 정책에 대해 고유해야 하며 접두사가 붙어야 합니다name/. 정책 ID 대신 이름을 사용할 수 있습니다.

```
$ aws verifiedpermissions create-policy \
    --definition "{ \"static\": { \"Statement\": \"permit(principal, action,
 resource in Album::\\\"public_folder\\\");\"}}" \
    --policy-store-id PSEXAMPLEabcdefg111111 \
    --name name/example-policy
{
    "createdDate": "2023-06-12T20:33:37.382907+00:00",
    "lastUpdatedDate": "2023-06-12T20:33:37.382907+00:00",
    "policyId": "SPEXAMPLEabcdefg111111",
    "policyStoreId": "PSEXAMPLEabcdefg111111",
    "policyType": "STATIC",
    "resource": {
        "entityId": "public_folder",
        "entityType": "Album"
    }
}
```

> **Note**
>
> 정책 스토어의 다른 정책과 이미 연결된 이름을 지정하면 ConflictException 오류가 발생합니다.

Amazon Verified Permissions 정적 정책 편집

정책 스토어에서 기존 정적 정책을 편집할 수 있습니다. 정적 정책만 직접 업데이트할 수 있습니다. 템플릿 연결 정책을 변경하려면 정책 템플릿을 업데이트해야 합니다. 자세한 내용은 Amazon Verified Permissions 정책 템플릿 편집 섹션을 참조하세요.

정적 정책의 다음 요소를 변경할 수 있습니다.

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

정적 정책의 다음 요소는 변경할 수 없습니다. 이러한 요소를 변경하려면 정책을 삭제하고 다시 생성해야 합니다.

- 정적 정책에서 템플릿 연결 정책으로의 정책입니다.
- `permit` 또는의 정적 정책의 영향입니다`forbid`.
- 정적 정책에서 참조하는 `principal`.
- 정적 정책에서 참조하는 `resource`.

AWS Management Console

정적 정책을 편집하려면

1. Verified Permissions 콘솔을 엽니다. 정책 스토어를 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 `resource`는 업데이트할 수 없습니다.
5. 정책 업데이트를 선택합니다.

> **Note**
>
> 정책 스토어에서 정책 검증을 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

UpdatePolicy 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\" );"
    }
}
```

다음 명령은 해당 파일을 참조합니다.

```
$ aws verifiedpermissions create-policy \
    --definition file://definition.txt \
    --policy-store-id PSEXAMPLEabcdefg111111

{
    "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"
    }
}
```

정책의 이름을 업데이트하려면

정책을 업데이트할 때 정책 이름을 설정하거나 업데이트할 수 있습니다. 이름은 정책 스토어 내의 모든 정책에 대해 고유해야 하며 접두사가 붙어야 합니다name/. 업데이트 요청에 이름 필드를 포함하지 않으면 기존 이름은 변경되지 않습니다. 이름을 제거하려면 빈 문자열로 설정합니다.

```
$ aws verifiedpermissions update-policy \
    --policy-id SPEXAMPLEabcdefg111111 \
    --policy-store-id PSEXAMPLEabcdefg111111 \
    --definition file://definition.txt \
    --name name/example-policy
{
    "createdDate": "2023-06-12T20:33:37.382907+00:00",
```

```
    "lastUpdatedDate": "2023-06-12T20:47:42.804511+00:00",
    "policyId": "SPEXAMPLEabcdefg111111",
    "policyStoreId": "PSEXAMPLEabcdefg111111",
    "policyType": "STATIC",
    "principal": {
        "entityId": "janeFriends",
        "entityType": "UserGroup"
    },
    "resource": {
        "entityId": "vacationFolder",
        "entityType": "Album"
    }
}
```

컨텍스트 추가

컨텍스트는 정책 결정과 관련이 있지만 보안 주체, 작업 또는 리소스의 자격 증명의 일부가 아닌 정보입니다. 액세스 토큰 클레임은 컨텍스트입니다. 소스 IP 주소 집합에서만 또는 사용자가 MFA로 로그인한 경우에만 작업을 허용할 수 있습니다. 애플리케이션은이 컨텍스트 세션 데이터에 액세스할 수 있으며 권한 부여 요청에 채워야 합니다. Verified Permissions 권한 부여 요청의 컨텍스트 데이터는 `contextMap` 요소에서 JSON 형식이어야 합니다.

이 콘텐츠를 설명하는 예제는 샘플 정책 스토어에서 가져온 것입니다. 다음을 수행하려면 테스트 환경에 DigitalPetStore 샘플 정책 스토어를 생성합니다.

다음 컨텍스트 객체는 샘플 DigitalPetStore 정책 스토어를 기반으로 애플리케이션의 각 Cedar 데이터 유형 중 하나를 선언합니다.

```
"context": {
  "contextMap": {
    "AccountCodes": {
      "set": [
        {
          "long": 111122223333
        },
        {
          "long": 444455556666
        },
        {
          "long": 123456789012
        }
```

```
      ]
    },
    "approvedBy": {
    "entityIdentifier": {
      "entityId": "Bob",
      "entityType": "DigitalPetStore::User"
    }
    },
    "MfaAuthorized": {
      "boolean": true
    },
    "NetworkInfo": {
      "record": {
        "IPAddress": {
          "string": "192.0.2.178"
        },
        "Country": {
          "string": "United States of America"
        },
        "SSL": {
          "boolean": true
        }
    }
    },
    "RequestedOrderCount": {
      "long": 4
    },
    "UserAgent": {
      "string": "My UserAgent 1.12"
    }
  }
}
```

권한 부여 컨텍스트의 데이터 형식

부울

바이너리 `true` 또는 `false` 값입니다. 이 예제에서에 `true` 대한 부울 값은 고객이 주문 보기를 요청하기 전에 다중 인증을 수행했음을 `MfaAuthenticated` 나타냅니다.

Set

컨텍스트 요소의 모음입니다. 집합 멤버는이 예제와 같은 유형 또는 중첩 집합을 포함한 다양한 유형일 수 있습니다. 이 예에서 고객은 3개의 서로 다른 계정과 연결됩니다.

문자열

문자로 묶인 " 일련의 문자, 숫자 또는 기호입니다. 이 예에서 `UserAgent` 문자열은 고객이 주문 보기를 요청하는 데 사용한 브라우저를 나타냅니다.

Long

정수. 이 예에서는이 요청이 고객이 과거 주문 중 4개를 보도록 요청한 배치의 일부임을 `RequestedOrderCount` 나타냅니다.

레코드

속성 모음입니다. 요청 컨텍스트에서 이러한 속성을 선언해야 합니다. 스키마가 있는 정책 스토어에는 스키마에이 엔터티와 엔터티의 속성이 포함되어야 합니다. 이 예제에서 `NetworkInfo` 레코드에는 사용자의 발신 IP, 클라이언트가 결정한 해당 IP의 지리적 위치, 전송 중 암호화에 대한 정보가 포함됩니다.

EntityIdentifier

요청의 `entities` 요소에 선언된 엔터티 및 속성에 대한 참조입니다. 이 예에서는 직원가 사용자의 주문을 승인했습니다Bob.

예제 DigitalPetStore 앱에서이 예제 컨텍스트를 테스트하려면 요청 , `entities`정책 스토어 스키마 및 정적 정책을 Customer Role - Get Order 설명으로 업데이트해야 합니다.

권한 부여 컨텍스트를 수락하도록 DigitalPetStore 수정

처음에는 DigitalPetStore가 매우 복잡한 정책 저장소가 아닙니다. 제공된 컨텍스트를 지원하기 위해 미리 구성된 정책 또는 컨텍스트 속성은 포함되지 않습니다. 이 컨텍스트 정보로 예제 권한 부여 요청을 평가하려면 정책 스토어와 권한 부여 요청을 다음과 같이 수정합니다. 액세스 토큰 정보가 컨텍스트로 포함된 컨텍스트 예제는 Amazon Cognito 액세스 토큰 매핑 및 OIDC 액세스 토큰 매핑을 참조하세요.

Schema

정책 스토어 스키마에 다음 업데이트를 적용하여 새 컨텍스트 속성을 지원합니다. 다음과 `actions` 같이 `GetOrder`에서를 업데이트합니다.

```
"GetOrder": {
  "memberOf": [],
  "appliesTo": {
    "resourceTypes": [
      "Order"
```

```
    ],
    "context": {
      "type": "Record",
      "attributes": {
        "AccountCodes": {
          "type": "Set",
          "required": true,
          "element": {
            "type": "Long"
          }
        },
        "approvedBy": {
          "name": "User",
          "required": true,
          "type": "Entity"
        },
        "MfaAuthorized": {
          "type": "Boolean",
          "required": true
        },
        "NetworkInfo": {
          "type": "NetworkInfo",
          "required": true
        },
        "RequestedOrderCount": {
          "type": "Long",
          "required": true
        },
        "UserAgent": {
          "required": true,
          "type": "String"
        }
      }
    },
    "principalTypes": [
      "User"
    ]
  }
}
```

요청 컨텍스트NetworkInfo에서 이름이 인 record 데이터 유형을 참조하려면 스키마에 다음을 스키마 앞에 추가하여 스키마에 commonType 구문을 생성합니다actions. commonType 구문은 서로 다른 엔터티에 적용할 수 있는 공유 속성 집합입니다.

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

Policy

다음 정책은 제공된 각 컨텍스트 요소에서 충족해야 하는 조건을 설정합니다. 고객 역할 - 주문 가져오기라는 설명과 함께 기존 정적 정책을 기반으로 합니다. 이 정책은 처음에는 요청을 하는 보안 주체만 리소스의 소유자가 되도록 요구합니다.

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

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

1. 사용자는 MFA로 로그인해야 합니다.
2. 사용자의 웹 브라우저에는 문자열이 포함되어야 `User-Agent` 합니다`My UserAgent`.
3. 사용자가 4개 이하의 주문을 보도록 요청했어야 합니다.
4. 사용자의 계정 코드 중 하나는 여야 합니다`111122223333`.
5. 사용자의 IP 주소는 미국에서 시작되어야 하고, 암호화된 세션에 있어야 하며, IP 주소는 로 시작해야 합니다`192.0.2.`.
6. 직원은 주문을 승인해야 합니다. 권한 부여 요청의 `entities` 요소에서의 역할을 Bob 가진 사용자를 선언합니다`Employee`.

Request body

적절한 스키마 및 정책으로 정책 스토어를 구성한 후 Verified Permissions API 작업 IsAuthorized에 이 권한 부여 요청을 제공할 수 있습니다. `entities` 세그먼트에는 역할이 인 사용자Bob인의 정의가 포함되어 있습니다`Employee`.

```
{
  "principal": {
    "entityType": "DigitalPetStore::User",
    "entityId": "Alice"
  },
  "action": {
    "actionType": "DigitalPetStore::Action",
    "actionId": "GetOrder"
  },
  "resource": {
    "entityType": "DigitalPetStore::Order",
    "entityId": "1234"
  },
  "context": {
    "contextMap": {
      "AccountCodes": {
        "set": [
          {"long": 111122223333},
          {"long": 444455556666},
          {"long": 123456789012}
        ]
      },
      "approvedBy": {
```

```
        "entityIdentifier": {
          "entityId": "Bob",
          "entityType": "DigitalPetStore::User"
        }
      },
      "MfaAuthorized": {
        "boolean": true
      },
      "NetworkInfo": {
        "record": {
          "Country": {"string": "United States of America"},
          "IPAddress": {"string": "192.0.2.178"},
          "SSL": {"boolean": true}
        }
      },
      "RequestedOrderCount":{
        "long": 4
      },
      "UserAgent": {
        "string": "My UserAgent 1.12"
      }
    }
  },
  "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 Permissions 테스트 벤치 사용

Verified Permissions 테스트 벤치를 사용하여 권한 부여 요청을 실행하여 Verified Permissions 정책을 테스트하고 문제를 해결합니다. 테스트 벤치는 지정한 매개 변수를 사용하여 정책 스토어의 Cedar 정책이 요청을 승인할지 여부를 결정합니다. 권한 부여 요청을 테스트하는 동안 비주얼 모드와 JSON 모

드 사이를 전환할 수 있습니다. Cedar 정책의 구조 및 평가 방법에 대한 자세한 내용은 Cedar 정책 언어 참조 가이드의 Cedar의 기본 정책 구성을 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 데모는 AWS YouTube 채널의 Amazon Verified Permissions - Policy Creation and Testing(Primer Series #3)을 참조하세요.

Visual mode

> **Note**
>
> 테스트 벤치의 비주얼 모드를 사용하려면 정책 스토어에 스키마가 정의되어 있어야 합니다.

비주얼 모드에서 정책을 테스트하려면

1. Verified Permissions 콘솔을 엽니다. 정책 스토어를 선택합니다.
2. 왼쪽에 있는 탐색 창에서 테스트 벤치를 선택합니다.
3. 비주얼 모드를 선택합니다.
4. 보안 주체 섹션에서 스키마의 보안 주체 유형 중에서 작업을 수행하는 보안 주체를 선택합니다. 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.
6. 지정된 보안 주체의 각 속성에 대해 속성 값을 지정합니다. 테스트 벤치는 시뮬레이션된 권한 부여 요청에서 지정된 속성 값을 사용합니다.
7. 리소스 섹션에서 보안 주체가 작업을 수행하는 리소스를 선택합니다. 텍스트 상자에 리소스 식별자를 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 부여 요청에서 지정된 속성 값을 사용합니다.
10. 작업 섹션에서 지정된 보안 주체 및 리소스에 대한 유효한 작업 목록에서 보안 주체가 수행하는 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 엔터티를 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.
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

JSON 모드에서 정책을 테스트하려면

1. Verified Permissions 콘솔을 엽니다. 정책 스토어를 선택합니다.
2. 왼쪽에 있는 탐색 창에서 테스트 벤치를 선택합니다.
3. JSON 모드를 선택합니다.
4. 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 시뮬레이션합니다. 테스트 벤치에는 평가 중에 충족된 정책이나 발생한 오류에 대한 정보와 함께 요청 허용 또는 거부 결정이 표시됩니다.

리소스당 정책 크기

Verified Permissions는 동일한 리소스를 참조하는 모든 정책의 총 크기를 제한합니다. 이 제한은 리소스 할당량당 정책 크기이며 기본값은 1MB입니다. `CreatePolicy` 또는를 호출하면 `UpdatePolicy`Verified Permissions가 참조된 리소스의 새 합계를 계산합니다. 새 합계가 할당량을 초과하면 요청이와 함께 실패합니다`ServiceQuotaExceededException`.

또한 Verified Permissions는 보안 주체와 리소스의 동일한 조합을 참조하는 모든 정책의 총 크기를 제한합니다. 이 제한은 리소스 보안 주체 조합 할당량당 정책 크기이며 값은 200,000바이트입니다. 이 제한은 조정할 수 없습니다.

리소스를 지정하지 않는 정책은 모든 리소스에 적용됩니다. 이러한 정책은 `"unspecified"` 리소스에 대해 별도의 합계를 공유하며, 리소스 할당량당 동일한 정책 크기가 적용됩니다.

모든 Verified Permissions 할당량 목록은 섹션을 참조하세요Amazon Verified Permissions 할당량.

주제

- 정책 크기 계산 방법
- 할당량 내에서 유지
- 템플릿 연결 정책 크기 예제

정책 크기 계산 방법

각 정책은 다음과 같이 리소스의 합계에 기여합니다.

- 정적 정책 - 정적 정책은 전체 크기를 제공합니다. 이는 정책 크기 할당량에 대해 측정되는 값과 동일합니다.
- 템플릿 연결 정책 - 템플릿 연결 정책은 인스턴스화에 사용되는 보안 주체 및 리소스의 총 길이와 데이터베이스 항목에 저장된 일부 고정 오버헤드를 제공합니다. 템플릿에 보안 주체 또는 리소스 슬롯이 없는 경우 지정되지 않은이 개체의 길이는 0으로 계산됩니다. 정책 템플릿 본문 자체의 크기는이 할당량에 포함되지 않습니다.

또한 각 정책은 총계에 최대 수백 바이트의 고정 오버헤드를 추가합니다. 템플릿 연결 정책은 정적 정책보다 더 많은 오버헤드를 추가합니다.

할당량 내에서 유지

애플리케이션이 리소스 할당량당 정책 크기에 가까워지면 다음 전략을 고려하세요.

- 적절한 정책 유형을 사용합니다. 많은 정책이 동일한 리소스에 대해 유사한 권한을 부여하는 경우 정책 템플릿은 정책 스토어의 스토리지 소비를 줄일 수 있습니다. 각 템플릿 연결 정책은 보안 주체와 리소스만 저장합니다. 정책 템플릿은 공유 문을 한 번 저장하지만 연결된 각 정책에는 고정된 스토리지 오버헤드도 포함됩니다. 템플릿은 공유 문이 해당 오버헤드를 상쇄할 만큼 충분히 크거나 약 200바이트 이상인 경우에만 비용을 지불합니다. 이 임계값 미만에서는 개별 정적 정책이 공간을 적게 사용합니다. 자세한 내용은 Amazon Verified Permissions 정책 템플릿 및 템플릿 연결 정책 섹션을 참조하세요.
- 정책 범위에서 보안 주체와 리소스를 지정합니다. 리소스를 지정하지 않는 정책은 정책 스토어 전체에서 `"unspecified"` 리소스에 대한 단일 합계를 공유합니다. 각 정책을 특정 리소스로 범위 지정하면 각 리소스에 대해 하나씩 정책을 별도의 합계로 분산합니다.
- 보안 주체와 리소스를 그룹화합니다. 각 보안 주체에 대한 정책을 작성하는 대신 그룹에 대한 정책을 작성하고 보안 주체를 그룹에 추가합니다. 리소스를 컨테이너로 그룹화하면 단일 리소스를 참조하는 정책 수를 비슷하게 줄일 수 있습니다.
- 할당량 증가를 요청하십시오. 정책 설계가 특정 제약 조건을 충족하는 경우 리소스 할당량당 정책 크기 증가를 요청할 수 있습니다. 자격을 갖추려면 정책 범위에서 보안 주체 또는 리소스를 지정해야 합니다. 또한 증가하면 보안 주체와 리소스가 가질 수 있는 계층 구조 상위 항목 수를 줄일 수 있습니다. 리소스 보안 주체 조합당 정책 크기 할당량은 보안 주체와 리소스의 동일한 조합을 참조하는 정책에 적용되는 별도의 200,000바이트 제한입니다. 이 제한은 조정할 수 없습니다.

템플릿 연결 정책 크기 예제

템플릿 연결 정책이 리소스 할당량당 정책 크기 및 리소스 보안 주체 조합 할당량당 정책 크기에 어떻게 기여하는지 확인하려면 보안 주체의 길이와 리소스의 길이를 추가합니다. 리소스를 지정하지 않으면 Verified Permissions가 해당 크기를 "unspecified" 리소스에 가산합니다.

다음 템플릿을 검토합니다.

```
permit (
  principal in ?principal,
  action in
    [MyApplication::Action::"ContentManagement",
     MyApplication::Action::"AccountAdministration",
     MyApplication::Action::"BillingOperations"],
  resource in ?resource
)
when {
  resource has authorizedTenantIds &&
  principal has tenantId &&
  resource.authorizedTenantIds.contains(principal.tenantId)
};
```

해당 템플릿에서 다음 정책을 생성합니다.

```
TemplateLinkedPolicy {
  policyId: "policy1",
  templateId: "template1",
  principal: User::"alice",
  resource: Photo::"car.jpg"
}

TemplateLinkedPolicy {
  policyId: "policy2",
  templateId: "template1",
  principal: User::"bob",
  resource: Photo::"boat.jpg"
}

TemplateLinkedPolicy {
  policyId: "policy3",
  templateId: "template1",
  principal: User::"jane",
```

```
  resource: Photo::"car.jpg"
}

TemplateLinkedPolicy {
  policyId: "policy4",
  templateId: "template1",
  principal: User::"jane",
  resource: Photo::"boat.jpg"
}

TemplateLinkedPolicy {
  policyId: "policy5",
  templateId: "template1",
  principal: User::"jane",
  resource
}
```

각각에 resource 대해 principal 및의 바이트 수를 계산하여 해당 정책의 크기를 계산합니다. Verified Permissions는 UTF-8 바이트 단위로 크기를 측정합니다. 인쇄 가능한 ASCII 범위의 문자는 각각 1바이트로 계산되지만 다른 문자는 더 많이 사용할 수 있습니다. 다음 예제에서는 ASCII 문자만 사용하므로 각 문자는 1바이트로 계산됩니다. 이러한 계산은 고정 오버헤드를 제외합니다. 실제 합계는 더 높습니다.

의 크기는 보안 주체의 policy1 길이User::"alice"(13)와 리소스의 길이Photo::"car.jpg"(16)입니다. 합계는 13 + 16 = 29바이트입니다.

의 크기는 보안 주체User::"bob"(11)의 policy2 길이와 리소스Photo::"boat.jpg"(17)의 길이입니다. 합계는 11 + 17 = 28바이트입니다.

의 크기는 보안 주체의 policy3 길이User::"jane"(12)와 리소스의 길이Photo::"car.jpg"(16)입니다. 합계는 12 + 16 = 28바이트입니다.

의 크기는 보안 주체의 policy4 길이User::"jane"(12)와 리소스의 길이Photo::"boat.jpg"(17)입니다. 합계는 12 + 17 = 29바이트입니다.

의 크기는 보안 주체의 길이User::"jane"(12)에 리소스의 길이(0)를 더한 값policy5입니다. 합계는 12 + 0 = 12바이트입니다.

다음 합계는 리소스 할당량당 정책 크기에 적용됩니다. policy5는 리소스를 지정하지 않으므로 해당 크기는 "unspecified" 리소스에 포함됩니다.

- "unspecified" 리소스의 합계는 policy5 (12)이며 12바이트입니다.

- 의 합계Photo::"car.jpg"는 policy1 57바이트인 (29)와 policy3 (28)의 합계입니다.
- 의 합계Photo::"boat.jpg"는 policy2 57바이트인 (28)과 policy4 (29)의 합계입니다.

다음 합계는 리소스 보안 주체 조합 할당량당 정책 크기에 적용됩니다. policy5는 리소스를 지정하지 않으므로 크기는 보안 주체, User::"jane"및 "unspecified" 리소스의 조합에 포함됩니다.

- User::"alice" 및의 합계Photo::"car.jpg"는 policy1 (29)이며 29바이트입니다.
- User::"bob" 및의 합계Photo::"boat.jpg"는 policy2 (28)이며 28바이트입니다.
- User::"jane" 및의 합계Photo::"car.jpg"는 policy3 (28)이며 28바이트입니다.
- User::"jane" 및의 합계Photo::"boat.jpg"는 policy4 (29)이며 29바이트입니다.
- User::"jane" 및 "unspecified" 리소스의 합계는 policy5 (12)이며, 이는 12바이트입니다.

Amazon Verified Permissions 정책 예제

여기에 포함된 정책 예제 중 일부는 기본 Cedar 정책 예제이고 일부는 Verified Permissions에 따라 다릅니다. 기본 항목은 Cedar 정책 언어 참조 가이드에 연결되며 여기에 포함되어 있습니다. Cedar 정책 구문에 대한 자세한 내용은 Cedar 정책 언어 참조 가이드의 Cedar 기본 정책 구성을 참조하세요.

정책 예제

- 개별 엔터티에 대한 액세스 허용
- 개체 그룹에 대한 액세스 허용
- 모든 엔터티에 대한 액세스 허용
- 개체의 속성에 대한 액세스 허용(ABAC)
- 액세스 거부
- 대괄호 표기법을 사용하여 토큰 속성 참조
- 점 표기법을 사용하여 속성 참조
- Amazon Cognito ID 토큰 속성을 반영합니다.
- OIDC ID 토큰 속성을 반영합니다.
- Amazon Cognito 액세스 토큰 속성을 반영합니다.
- OIDC 액세스 토큰 속성을 반영합니다.

대괄호 표기법을 사용하여 토큰 속성 참조

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

Verified Permissions의 정책에서 토큰 속성을 사용하는 방법에 대한 자세한 내용은 스키마에 Amazon Cognito 토큰 매핑 및 스키마에 OIDC 토큰 매핑을 참조하세요.

```
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 속성 참조

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

Verified Permissions의 정책에서 토큰 속성을 사용하는 방법에 대한 자세한 내용은 스키마에 Amazon Cognito 토큰 매핑 및 스키마에 OIDC 토큰 매핑을 참조하세요.

```
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"
};
```

Amazon Cognito ID 토큰 속성을 반영합니다.

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

Verified Permissions의 정책에서 토큰 속성을 사용하는 방법에 대한 자세한 내용은 스키마에 Amazon Cognito 토큰 매핑 및 스키마에 OIDC 토큰 매핑을 참조하세요.

```
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"
};
```

OIDC ID 토큰 속성을 반영합니다.

다음 예제에서는 정책 참조를 생성하여 OIDC 공급자의 ID 토큰 속성을 참조하는 방법을 보여줍니다.

Verified Permissions의 정책에서 토큰 속성을 사용하는 방법에 대한 자세한 내용은 스키마에 Amazon Cognito 토큰 매핑 및 스키마에 OIDC 토큰 매핑을 참조하세요.

```
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 예제에서는 정책 참조 액세스 토큰 속성을 생성하는 방법을 보여줍니다 Amazon Cognito.

Verified Permissions의 정책에서 토큰 속성을 사용하는 방법에 대한 자세한 내용은 스키마에 Amazon Cognito 토큰 매핑 및 스키마에 OIDC 토큰 매핑을 참조하세요.

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

OIDC 액세스 토큰 속성을 반영합니다.

다음 예제에서는 정책 참조를 생성하여 OIDC 공급자의 액세스 토큰 속성을 참조하는 방법을 보여줍니다.

Verified Permissions의 정책에서 토큰 속성을 사용하는 방법에 대한 자세한 내용은 스키마에 Amazon Cognito 토큰 매핑 및 스키마에 OIDC 토큰 매핑을 참조하세요.

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

Amazon Verified Permissions 정책 템플릿 및 템플릿 연결 정책

Verified Permissions에서 정책 템플릿은 , principal resource또는 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 보안 주체 및 리소스와 함께 사용할 수 있습니다. 정책 템플릿 업데이트는 템플릿을 사용하는 모든 정책에 반영됩니다. 자세한 내용은 Cedar 정책 언어 참조 가이드의 Cedar 정책 템플릿을 참조하세요.

선택적으로 정책 템플릿 이름을 정책 템플릿에 할당할 수 있습니다. 정책 템플릿 이름은 정책 스토어 내에서 고유해야 하며 접두사가 붙어야 합니다name/. policyTemplateId 파라미터를 수락하는 컨트롤 플레인 작업에서 정책 템플릿 ID 대신 정책 템플릿 이름을 사용할 수 있습니다. GetPolicyTemplate 및 만 출력에 이름을 ListPolicyTemplates 반환합니다. 다음 예제에서는 정책 템플릿 이름을 사용하여를 사용하여 정책 템플릿을 검색합니다GetPolicyTemplate.

```
$ aws verifiedpermissions get-policy-template \
    --policy-template-id name/example-policy-template \
    --policy-store-id PSEXAMPLEabcdefg111111
```

예를 들어 다음 정책 템플릿은 정책 템플릿을 사용하는 보안 주체 및 리소스에 대한 ReadEdit, 및 Comment 권한을 제공합니다.

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

이 템플릿을 Editor 기반으로 라는 정책을 생성하려는 경우 보안 주체가 특정 리소스의 편집기로 지정되면 애플리케이션에서 보안 주체가 리소스를 읽고 편집하고 댓글을 달 수 있는 권한을 제공하는 정책을 생성합니다.

정적 정책과 달리 템플릿 연결 정책은 동적입니다. 정책 템플릿에서 Comment 작업을 제거하려는 경우 해당 템플릿에 연결되거나 이를 기반으로 하는 모든 정책이 그에 따라 업데이트되고 정책에 지정된 보안 주체가 더 이상 해당 리소스에 주석을 달 수 없게 됩니다.

템플릿 연결 정책 예제에 대한 자세한 내용은 섹션을 참조하세요Amazon Verified Permissions 예제 템플릿 연결 정책.

Amazon Verified Permissions 정책 템플릿 생성

AWS Management Console AWS CLI, 또는 AWS SDKs. 정책 템플릿을 사용하면 정책을 한 번 정의한 다음 여러 보안 주체 및 리소스와 함께 사용할 수 있습니다. 정책 템플릿을 생성한 후 템플릿 연결 정책을 생성하여 정책 템플릿을 특정 보안 주체 및 리소스와 함께 사용할 수 있습니다. 자세한 내용은 Amazon Verified Permissions 템플릿 연결 정책 생성 단원을 참조하십시오.

AWS Management Console

정책 템플릿을 생성하려면

1. Verified Permissions 콘솔을 엽니다. 정책 스토어를 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

CreatePolicyTemplate 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 PSEXAMPLEabcdefg111111
{
    "createdDate": "2023-05-18T21:17:47.284268+00:00",
    "lastUpdatedDate": "2023-05-18T21:17:47.284268+00:00",
    "policyStoreId": "PSEXAMPLEabcdefg111111",
    "policyTemplateId": "PTEXAMPLEabcdefg111111"
}
```

정책 템플릿 이름을 사용하여 정책 템플릿을 생성하려면

정책 템플릿을 생성할 때 선택적으로 정책 템플릿 이름을 지정할 수 있습니다. 이름은 정책 스토어 내의 모든 정책 템플릿에 대해 고유해야 하며 접두사가 붙어야 합니다name/. 정책 템플릿 ID 대신 이름을 사용할 수 있습니다.

```
$ aws verifiedpermissions create-policy-template \
    --description "Template for vacation picture access" \
    --statement file://template1.txt \
    --policy-store-id PSEXAMPLEabcdefg111111 \
    --name name/example-policy-template
{
    "createdDate": "2023-06-12T20:47:42.804511+00:00",
    "lastUpdatedDate": "2023-06-12T20:47:42.804511+00:00",
    "policyStoreId": "PSEXAMPLEabcdefg111111",
    "policyTemplateId": "PTEXAMPLEabcdefg111111"
}
```

> **Note**
>
> 정책 스토어의 다른 정책 템플릿과 이미 연결된 이름을 지정하면 `ConflictException` 오류가 발생합니다.

Amazon Verified Permissions 템플릿 연결 정책 생성

, AWS Management Console AWS CLI또는 AWS SDKs. 템플릿 연결 정책은 해당 정책 템플릿에 연결된 상태로 유지됩니다. 정책 템플릿에서 정책 문을 변경하면 해당 템플릿에 연결된 모든 정책은 해당 시점부터 이루어진 모든 권한 부여 결정에 대해 새 문을 자동으로 사용합니다.

템플릿 연결 정책 예제는 섹션을 참조하세요Amazon Verified Permissions 예제 템플릿 연결 정책.

AWS Management Console

정책 템플릿을 인스턴스화하여 템플릿 연결 정책을 생성하려면

1. Verified Permissions 콘솔을 엽니다. 정책 스토어를 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. `AdminUsers`를 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

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"
    }
}
```

Amazon Verified Permissions 정책 템플릿 편집

, 또는 SDK를 사용하여 Verified Permissions에서 정책 템플릿을 편집 AWS Management Console AWS CLI하거나 업데이트할 수 있습니다. AWS SDKs 정책 템플릿을 편집하면 템플릿에 연결되거나 이를 기반으로 하는 정책이 자동으로 업데이트되므로 정책 템플릿을 편집할 때 주의해야 하며 애플리케이션을 손상시키는 변경 사항이 실수로 도입되지 않도록 해야 합니다.

정책 템플릿의 다음 요소를 변경할 수 있습니다.

- 정책 템플릿에서 `action` 참조하는
- `when` 및와 같은 조건 절 `unless`

정책 템플릿의 다음 요소는 변경할 수 없습니다. 이러한 요소를 변경하려면 정책 템플릿을 삭제하고 다시 생성해야 합니다.

- `permit` 또는에서 정책 템플릿의 영향 `forbid`
- 정책 템플릿에서 `principal` 참조하는
- 정책 템플릿에서 `resource` 참조하는

AWS Management Console

정책 템플릿을 편집하려면

1. Verified Permissions 콘솔을 엽니다. 정책 스토어를 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 업데이트한 다음 정책 템플릿 업데이트를 선택합니다.
5. 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 편집하려면

UpdatePolicy 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",
```

```
    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"
}
```

정책 템플릿의 이름을 업데이트하려면

정책 템플릿을 업데이트할 때 정책 템플릿 이름을 설정하거나 업데이트할 수 있습니다. 이름은 정책 스토어 내의 모든 정책 템플릿에 대해 고유해야 하며 접두사가 붙어야 합니다name/. 업데이트 요청에 이름 필드를 포함하지 않으면 기존 이름은 변경되지 않습니다. 이름을 제거하려면 빈 문자열로 설정합니다.

```
$ aws verifiedpermissions update-policy-template \
    --policy-template-id PTEXAMPLEabcdefg111111 \
    --statement file://template1.txt \
    --policy-store-id PSEXAMPLEabcdefg111111 \
    --name name/example-policy-template
{
    "createdDate": "2023-05-17T18:58:48.795411+00:00",
    "lastUpdatedDate": "2023-05-17T19:18:48.870209+00:00",
    "policyStoreId": "PSEXAMPLEabcdefg111111",
    "policyTemplateId": "PTEXAMPLEabcdefg111111"
}
```

Amazon Verified Permissions 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. 다음

과 같은 Verified Permissions 템플릿 연결 정책 예제를 샘플 정책 스토어 및 해당 정책, 정책 템플릿 및 스키마와 함께 사용할 수 있습니다.

PhotoFlash 예제

다음 예제에서는 정책 템플릿을 사용하는 템플릿 연결 정책을 생성하는 방법을 보여줍니다. 개별 사용자 및 사진과 비공개 공유 사진에 대한 제한된 액세스 권한을 부여합니다.

> **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"
 );
```

다음 예제에서는 정책 템플릿을 사용하는 템플릿 연결 정책을 생성하는 방법을 보여줍니다. 개별 사용자 및 앨범과 비공개 공유 사진에 대한 제한된 액세스 권한을 부여합니다.

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

다음 예제에서는 정책 템플릿을 사용하는 템플릿 연결 정책을 생성하는 방법을 보여줍니다. 친구 그룹 및 개별 사진과 비공개 공유 사진에 대한 제한된 액세스 권한을 부여합니다.

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

다음 예제에서는 정책 템플릿을 사용하는 템플릿 연결 정책을 생성하는 방법을 보여줍니다. 친구 그룹 및 앨범과 비공개 공유 사진에 대한 제한된 액세스 권한을 부여합니다.

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

다음 예제에서는 정책 템플릿을 사용하는 템플릿 연결 정책을 생성하는 방법을 보여줍니다. 친구 그룹 및 개별 사진과 비공개 공유 사진에 대한 전체 액세스 권한을 부여합니다.

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

다음 예제에서는 정책 템플릿 계정에서 사용자 차단을 사용하는 템플릿 연결 정책을 생성하는 방법을 보여줍니다.

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

DigitalPetStore 예제

DigitalPetStore 샘플 정책 스토어에는 정책 템플릿이 포함되어 있지 않습니다. DigitalPetStore 샘플 정책 스토어를 만든 후 왼쪽의 탐색 창에서 정책을 선택하여 정책 스토어에 포함된 정책을 볼 수 있습니다.

TinyToDo 예제

다음 예제에서는 최종 사용자에게 개별 사용자 및 작업 목록에 대한 액세스 권한을 부여하는 정책 템플릿을 사용하는 템플릿 연결 정책을 생성하는 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 예제에서는 개별 사용자 및 작업 목록에 대한 편집기 액세스 권한을 부여하는 정책 템플릿을 사용하는 템플릿 연결 정책을 생성하는 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"
);
```

자격 증명 소스 및 토큰으로 애플리케이션 보호

Amazon Verified Permissions에서 외부 ID 제공업체(IdP)를 나타내는 ID 소스를 생성하여 애플리케이션을 빠르게 보호합니다. 자격 증명 소스는 정책 스토어와 신뢰 관계가 있는 IdP로 인증된 사용자의 정보를 제공합니다. 애플리케이션이 자격 증명 소스의 토큰으로 권한 부여를 요청할 때 정책 스토어는 사용자 속성 및 액세스 권한에서 권한 부여를 결정할 수 있습니다. Amazon Cognito 사용자 풀 또는 사용자 지정 OpenID Connect(OIDC) IdP를 ID 소스로 추가할 수 있습니다.

Verified Permissions와 함께 OpenID Connect(OIDC) ID 제공업체(IdPs)를 사용할 수 있습니다. 애플리케이션은 OIDC 준수 자격 증명 공급자가 생성한 JSON 웹 토큰(JWTs)을 사용하여 권한 부여 요청을 생성할 수 있습니다. 토큰의 사용자 자격 증명은 보안 주체 ID에 매핑됩니다. ID 토큰을 사용하면 Verified Permissions는 속성 클레임을 보안 주체 속성에 매핑합니다. 액세스 토큰을 사용하면 이러한 클레임이 컨텍스트에 매핑됩니다. 두 토큰 유형 모두와 같은 클레임을 보안 주체 그룹에 매핑하고 역할 기반 액세스 제어(RBAC)`groups`를 평가하는 정책을 구축할 수 있습니다.

> **Note**
>
> Verified Permissions는 IdP 토큰의 정보를 기반으로 권한 부여를 결정하지만 어떤 식으로든 IdP와 직접 상호 작용하지 않습니다.

Amazon Cognito 사용자 풀 또는 OIDC 자격 증명 공급자를 사용하여 Amazon API Gateway REST APIs에 대한 권한 부여 로직을 구축하는 step-by-step 연습은 AWS 보안 블로그의 Authorize API Gateway APIs using Amazon Verified Permissions with Amazon Cognito or bring your own identity provider를 참조하세요.

주제

- 올바른 자격 증명 공급자 선택
- Amazon Cognito 자격 증명 소스 작업
- OIDC 자격 증명 소스 작업

올바른 자격 증명 공급자 선택

Verified Permissions는 다양한 IdPs에서 작동하지만 애플리케이션에서 사용할 IdP를 결정할 때는 다음 사항을 고려하세요.

다음과 같은 Amazon Cognito 경우에 사용합니다.

- 기존 자격 증명 인프라 없이 새 애플리케이션을 구축하는 경우
- 기본 보안 기능이 있는 원하는 AWS관리형 사용자 풀
- 소셜 자격 증명 공급자 통합이 필요합니다.
- 간소화된 토큰 관리를 원하는 경우

다음과 같은 경우 OIDC 공급자를 사용합니다.

- 기존 자격 증명 인프라(Auth0, Okta, Azure AD)가 있는 경우
- 중앙 집중식 사용자 관리를 유지해야 함
- 특정 IdPs

Amazon Cognito 자격 증명 소스 작업

Verified Permissions는 Amazon Cognito 사용자 풀과 밀접하게 작동합니다. Amazon Cognito JWTs 예측 가능한 구조를 갖습니다. Verified Permissions는이 구조를 인식하고 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.

- AWS 리전.
- 사용자 풀 ID입니다.
- 자격 증명 소스와 연결할 보안 주체 엔터티 유형입니다. 예: `MyCorp::User`.
- 자격 증명 소스와 연결할 보안 주체 그룹 엔터티 유형입니다. 예: `MyCorp::UserGroup`.
- 가 정책 스토어에 요청할 수 있도록 승인하려는 사용자 풀의 클라이언트 IDs입니다.

Verified Permissions는 동일한의 Amazon Cognito 사용자 풀에서만 작동하므로 다른 계정에서 자격 증명 소스를 지정할 AWS 계정수 없습니다. Verified Permissions는 사용자 풀 보안 주체를 기반으로 하는 정책에서 참조해야 하는 자격 증명 소스 식별자인 엔터티 접두사를와 같은 사용자 풀의 ID로 설정합니다`us-west-2_EXAMPLE`. 이 경우 ID가 인 사용자 풀의 사용자를 참조`a1b2c3d4-5678-90ab-cdef-EXAMPLE22222`합니다. `us-west-2_EXAMPLE|a1b2c3d4-5678-90ab-cdef-EXAMPLE22222`

사용자 풀 토큰 클레임에는 속성, 범위, 그룹, 클라이언트 IDs 및 사용자 지정 데이터가 포함될 수 있습니다. Amazon Cognito JWTs는 Verified Permissions에서 권한 부여 결정에 기여할 수 있는 다양한 정보를 포함할 수 있습니다. 다음이 포함됩니다.

1. `cognito:` 접두사가 있는 사용자 이름 및 그룹 클레임
2. 를 사용한 사용자 지정 사용자 속성 `custom: prefix`
3. 런타임에 추가된 사용자 지정 클레임
4. `sub` 및와 같은 OIDC 표준 클레임 `email`

이러한 클레임과 이를 관리하는 방법은의 Verified Permissions 정책에서 자세히 다룹니다스키마에 Amazon Cognito 토큰 매핑.

> **Important**
>
> 만료되기 전에 Amazon Cognito 토큰을 취소할 수 있지만 JWTs는 서명 및 유효성이 포함된 상태 비저장 리소스로 간주됩니다. JSON 웹 토큰 RFC 7519를 준수하는 서비스는 원격으로 토큰을 검증할 것으로 예상되며 발행자를 통해 토큰을 검증할 필요는 없습니다. 즉, Verified Permissions가 나중에 삭제된 사용자에 대해 취소되거나 발급된 토큰을 기반으로 액세스 권한을 부여할 수 있습니다. 이러한 위험을 줄이려면 유효 기간이 가장 짧은 토큰을 생성하고 사용자 세션을 계속할 수 있는 권한을 제거하려는 경우 새로 고침 토큰을 취소하는 것이 좋습니다. 자세한 내용은 토큰 해지로 사용자 세션 종료를 참조하세요.

다음 예제에서는 보안 주체와 연결된 일부 Amazon Cognito 사용자 풀 클레임을 참조하는 정책을 생성하는 방법을 보여줍니다.

```
permit(
     principal,
     action,
     resource == ExampleCo::Photo::"VacationPhoto94.jpg"
)
when {
     principal["cognito:username"]) == "alice" &&
     principal["custom:department"]) == "Finance"
};
```

다음 예제에서는 Cognito 사용자 풀의 사용자인 보안 주체를 참조하는 정책을 생성하는 방법을 보여줍니다. 보안 주체 ID는 형식을 취합니다`"<userpool-id>|<sub>"`.

```
permit(
     principal == ExampleCo::User::"us-east-1_example|a1b2c3d4-5678-90ab-cdef-
EXAMPLE11111",
```

```
    action,
    resource == ExampleCo::Photo::"VacationPhoto94.jpg"
);
```

Verified Permissions의 사용자 풀 자격 증명 소스에 대한 Cedar 정책은 영숫자 및 밑줄() 이외의 문자가 포함된 클레임 이름에 특수 구문을 사용합니다_. 여기에는 `cognito:username` 및와 같은 : 문자가 포함된 사용자 풀 접두사 클레임이 포함됩니다`custom:department`. `cognito:username` 또는 `custom:department` 클레임을 참조하는 정책 조건을 작성하려면 `principal["custom:department"]`각각 `principal["cognito:username"]` 및 로 작성합니다.

> **Note**
>
> 토큰에 `custom:` 접두사가 `cognito:` 또는 이고 클레임 이름이 리터럴 값 `cognito` 또는 인 클레임이 포함된 경우 IsAuthorizedWithToken을 사용한 `custom`권한 부여 요청은와 함께 실패합니다`ValidationException`.

클레임 매핑에 대한 자세한 내용은 섹션을 참조하세요스키마에 Amazon Cognito 토큰 매핑. Amazon Cognito 사용자 권한 부여에 대한 자세한 내용은 Amazon Cognito 개발자 안내서의 Amazon Verified Permissions 권한 부여를 참조하세요. Amazon Cognito

주제

- Amazon Verified Permissions Amazon Cognito 자격 증명 소스 생성
- Amazon Verified Permissions 자격 Amazon Cognito 증명 소스 편집
- 스키마에 Amazon Cognito 토큰 매핑
- 에 대한 클라이언트 및 대상 검증 Amazon Cognito

Amazon Verified Permissions Amazon Cognito 자격 증명 소스 생성

다음 절차에서는 기존 정책 스토어에 자격 증명 소스를 추가합니다.

Verified Permissions 콘솔에서 새 정책 스토어를 생성할 때 자격 증명 소스를 생성할 수도 있습니다. 이 프로세스에서는 자격 증명 소스 토큰의 클레임을 개체 속성으로 자동으로 가져올 수 있습니다. 안내 설정 또는 API Gateway 설정 및 자격 증명 공급자 옵션을 선택합니다. 이러한 옵션은 초기 정책도 생성합니다.

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

에서 create-identity-source를 사용하여 자격 증명 소스를 생성 AWS CLI 하거나 Verified Permissions API에서 CreateIdentitySource를 사용하여 자격 증명 소스를 생성할 때 보안 주체 엔터티 유형을 제외할 수 있습니다. 그러나 빈 엔터티 유형은 엔터티 유형이 인 ID 소스를 생성합니다`AWS::Cognito`. 이 엔터티 이름은 정책 스토어 스키마와 호환되지 않습니다. 자격 증명을 정책 스토어 스키마와 통합하려면 보안 주체 Amazon Cognito 엔터티 유형을 지원되는 정책 스토어 엔터티로 설정해야 합니다.

AWS Management Console

Amazon Cognito 사용자 풀 자격 증명 소스를 생성하려면

1. Verified Permissions 콘솔을 엽니다. 정책 스토어를 선택합니다.
2. 왼쪽 탐색 창에서 자격 증명 소스를 선택합니다.
3. 자격 증명 소스 생성을 선택합니다.
4. Cognito 사용자 풀 세부 정보에서를 선택하고 자격 증명 소스의 사용자 풀 ID를 AWS 리전 입력합니다.
5. 보안 주체 구성의 보안 주체 유형에서이 소스의 보안 주체에 대한 엔터티 유형을 선택합니다. 연결된 Amazon Cognito 사용자 풀의 자격 증명은 선택한 보안 주체 유형에 매핑됩니다.
6. 사용자 풀 `cognito:groups` 클레임을 매핑하려면 그룹 구성에서 Cognito 그룹 사용을 선택합니다. 보안 주체 유형의 상위인 엔터티 유형을 선택합니다.
7. 클라이언트 애플리케이션 검증에서 클라이언트 애플리케이션 IDs 검증할지 여부를 선택합니다.
   - 클라이언트 애플리케이션 ID를 검증하려면 클라이언트 애플리케이션 ID가 일치하는 토큰만 수락을 선택합니다. 검증할 각 클라이언트 애플리케이션 ID에 대해 새 클라이언트 애플리케이션 ID 추가를 선택합니다. 추가된 클라이언트 애플리케이션 ID를 제거하려면 클라이언트 애플리케이션 ID 옆의 제거를 선택합니다.
   - 클라이언트 애플리케이션 ID를 검증하지 않으려면 클라이언트 애플리케이션 ID 검증 안 함을 선택합니다.
8. 자격 증명 소스 생성을 선택합니다.

9. (선택 사항) 정책 스토어에 스키마가 있는 경우 Cedar 정책의 자격 증명 또는 액세스 토큰에서 추출한 속성을 참조하려면 먼저 Cedar가 자격 증명 소스가 생성하는 보안 주체 유형을 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 섹션을 참조하세요스키마에 Amazon Cognito 토큰 매핑.

   > **Note**
   >
   > API 연결 정책 저장소를 생성하거나 정책 저장소를 생성할 때 API Gateway 및 자격 증명 공급자로 설정을 사용하는 경우 Verified Permissions는 사용자 풀에서 사용자 속성을 쿼리하고 보안 주체 유형이 사용자 풀 속성으로 채워지는 스키마를 생성합니다.

10. 토큰의 정보를 사용하여 권한 부여 결정을 내리는 정책을 생성합니다. 자세한 내용은 Amazon Verified Permissions 정적 정책 생성 단원을 참조하십시오.

이제 자격 증명 소스를 생성하고, 스키마를 업데이트하고, 정책을 생성했으므로 `IsAuthorizedWithToken`를 사용하여 Verified Permissions가 권한 부여 결정을 내리도록 합니다. 자세한 내용은 Amazon Verified Permissions API 참조 안내서의 IsAuthorizedWithToken을 참조하세요.

AWS CLI

Amazon Cognito 사용자 풀 자격 증명 소스를 생성하려면

CreateIdentitySource 작업을 사용하여 자격 증명 소스를 생성할 수 있습니다. 다음 예제에서는 Amazon Cognito 사용자 풀에서 인증된 자격 증명에 액세스할 수 있는 자격 증명 소스를 생성합니다.

1. `create-identity-source` 명령의 `--configuration` 파라미터에서 사용할 Amazon Cognito 사용자 풀에 대한 다음 세부 정보가 포함된 `config.txt` 파일을 생성합니다.

```
{
    "cognitoUserPoolConfiguration": {
        "userPoolArn": "arn:aws:cognito-idp:us-west-2:123456789012:userpool/us-
west-2_1a2b3c4d5",
        "clientIds":["a1b2c3d4e5f6g7h8i9j0kalbmc"],
        "groupConfiguration": {
              "groupEntityType": "MyCorp::UserGroup"
        }
    }
```

```
}
```

2. 다음 명령을 실행하여 Amazon Cognito 자격 증명 소스를 생성합니다.

```
$ 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"
}
```

3. (선택 사항) 정책 스토어에 스키마가 있는 경우 Cedar 정책의 자격 증명 또는 액세스 토큰에서 추출한 속성을 참조하려면 먼저 Cedar가 자격 증명 소스가 생성하는 보안 주체 유형을 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 섹션을 참조하세요스키마에 Amazon Cognito 토큰 매핑.

> **Note**
>
> API 연결 정책 저장소를 생성하거나 정책 저장소를 생성할 때 API Gateway 및 자격 증명 공급자로 설정을 사용하는 경우 Verified Permissions는 사용자 풀에서 사용자 속성을 쿼리하고 보안 주체 유형이 사용자 풀 속성으로 채워지는 스키마를 생성합니다.

4. 토큰의 정보를 사용하여 권한 부여 결정을 내리는 정책을 생성합니다. 자세한 내용은 Amazon Verified Permissions 정적 정책 생성 단원을 참조하십시오.

이제 자격 증명 소스를 생성하고, 스키마를 업데이트하고, 정책을 생성했으므로 `IsAuthorizedWithToken`를 사용하여 Verified Permissions가 권한 부여 결정을 내리도록 합니다. 자세한 내용은 Amazon Verified Permissions API 참조 안내서의 IsAuthorizedWithToken을 참조하세요.

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

Amazon Verified Permissions 자격 Amazon Cognito 증명 소스 편집

자격 증명 소스를 생성한 후 일부 파라미터를 편집할 수 있습니다. 자격 증명 소스 유형은 변경할 수 없으며 자격 증명 소스를 삭제하고에서 OIDC 또는 OIDC Amazon Cognito 로 전환할 새 소스를 생성해야 합니다 Amazon Cognito. 정책 스토어 스키마가 자격 증명 소스 속성과 일치하는 경우 자격 증명 소스에 대한 변경 사항을 반영하도록 스키마를 별도로 업데이트해야 합니다.

AWS Management Console

자격 Amazon Cognito 증명 소스를 업데이트하려면

1. Verified Permissions 콘솔을 엽니다. 정책 스토어를 선택합니다.
2. 왼쪽 탐색 창에서 자격 증명 소스를 선택합니다.
3. 편집할 자격 증명 소스의 ID를 선택합니다.
4. 편집을 선택합니다.
5. Cognito 사용자 풀 세부 정보에서를 선택하고 자격 증명 소스의 사용자 풀 ID를 AWS 리전 입력합니다.
6. 보안 주체 세부 정보에서 자격 증명 소스의 보안 주체 유형을 업데이트할 수 있습니다. 연결된 Amazon Cognito 사용자 풀의 자격 증명은 선택한 보안 주체 유형에 매핑됩니다.
7. 사용자 풀 `cognito:groups` 클레임을 매핑하려면 그룹 구성에서 Cognito 그룹 사용을 선택합니다. 보안 주체 유형의 상위인 엔터티 유형을 선택합니다.
8. 클라이언트 애플리케이션 검증에서 클라이언트 애플리케이션 IDs 검증할지 여부를 선택합니다.
   - 클라이언트 애플리케이션 ID를 검증하려면 클라이언트 애플리케이션 ID가 일치하는 토큰만 수락을 선택합니다. 검증할 각 클라이언트 애플리케이션 ID에 대해 새 클라이언트 애플리케이션 ID 추가를 선택합니다. 추가된 클라이언트 애플리케이션 ID를 제거하려면 클라이언트 애플리케이션 ID 옆의 제거를 선택합니다.
   - 클라이언트 애플리케이션 ID를 검증하지 않으려면 클라이언트 애플리케이션 ID 검증 안 함을 선택합니다.
9. 변경 사항 저장을 선택합니다.
10. 자격 증명 소스의 보안 주체 유형을 변경한 경우 업데이트된 보안 주체 유형을 올바르게 반영하도록 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

자격 Amazon Cognito 증명 소스를 업데이트하려면

UpdateIdentitySource 작업을 사용하여 자격 증명 소스를 업데이트할 수 있습니다. 다음 예시에서는 다른 Amazon Cognito 사용자 풀을 사용하도록 지정된 자격 증명 소스를 업데이트합니다.

1. update-identity-source 명령의 --configuration 파라미터에서 사용할 Amazon Cognito 사용자 풀에 대한 다음 세부 정보가 포함된 config.txt 파일을 생성합니다.

```
{
    "cognitoUserPoolConfiguration": {
        "userPoolArn": "arn:aws:cognito-idp:us-west-2:123456789012:userpool/us-
west-2_1a2b3c4d5",
        "clientIds":["a1b2c3d4e5f6g7h8i9j0kalbmc"],
        "groupConfiguration": {
              "groupEntityType": "MyCorp::UserGroup"
        }
    }
}
```

2. 다음 명령을 실행하여 Amazon Cognito 자격 증명 소스를 업데이트합니다.

```
$ 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"
}
```

> **Note**
>
> 자격 증명 소스의 보안 주체 유형을 변경하는 경우 업데이트된 보안 주체 유형을 올바르게 반영하도록 스키마를 업데이트해야 합니다.

스키마에 Amazon Cognito 토큰 매핑

정책 스토어에 자격 증명 소스를 추가하고 정책 스토어 스키마에 공급자 클레임 또는 토큰을 매핑하려고 할 수 있습니다. 가이드 설정을 사용하여 자격 증명 소스로 정책 스토어를 생성하거나 정책 스토어가 생성된 후 스키마를 수동으로 업데이트하여이 프로세스를 자동화할 수 있습니다. 토큰을 스키마에 매핑한 후에는 토큰을 참조하는 정책을 생성할 수 있습니다.

사용 설명서의이 섹션에는 다음 정보가 포함되어 있습니다.

- 정책 스토어 스키마에 속성을 자동으로 채울 수 있는 경우
- Verified Permissions 정책에서 Amazon Cognito 토큰 클레임을 사용하는 방법
- 자격 증명 소스에 대한 스키마를 수동으로 빌드하는 방법

API 연결 정책 스토어 및 안내형 설정을 통해 생성된 자격 증명 소스가 있는 정책 스토어는 스키마에 자격 증명(ID) 토큰 속성을 수동으로 매핑할 필요가 없습니다. Verified Permissions에 사용자 풀의 속성을 제공하고 사용자 속성으로 채워진 스키마를 생성할 수 있습니다. ID 토큰 권한 부여에서 Verified Permissions는 클레임을 보안 주체 엔터티의 속성에 매핑합니다. 다음 조건에서 Amazon Cognito 토큰을 스키마에 수동으로 매핑해야 할 수 있습니다.

- 샘플에서 빈 정책 저장소 또는 정책 저장소를 생성했습니다.
- 액세스 토큰 사용을 역할 기반 액세스 제어(RBAC) 이상으로 확장하려고 합니다.
- Verified Permissions REST API, AWS SDK 또는를 사용하여 정책 스토어를 생성합니다 AWS CDK.

Verified Permissions 정책 스토어에서를 자격 증명 소스 Amazon Cognito 로 사용하려면 스키마에 공급자 속성이 있어야 합니다. 스키마는 고정되며 공급자 토큰이 IsAuthorizedWithToken 또는 BatchIsAuthorizedWithToken API 요청에서 생성하는 엔터티와 일치해야 합니다. ID 토큰의 공급자 정보에서 스키마를 자동으로 채우는 방식으로 정책 스토어를 생성한 경우 정책을 작성할 준비가 된 것입니다. 자격 증명 소스에 대한 스키마 없이 정책 스토어를 생성하는 경우 API 요청을 사용하여 생성된 엔터티와 일치하는 스키마에 공급자 속성을 추가해야 합니다. 그런 다음 공급자 토큰의 속성을 사용하여 정책을 작성할 수 있습니다.

Verified Permissions에서 인증된 사용자의 Amazon Cognito ID 및 액세스 토큰 사용에 대한 자세한 내용은 Amazon Amazon Cognito 개발자 안내서의 Amazon Verified Permissions 권한 부여를 참조하세요.

주제

- 스키마에 ID 토큰 매핑

- 액세스 토큰 매핑
- Amazon Cognito 콜론으로 구분된 클레임에 대한 대체 표기법
- 스키마 매핑에 대해 알아야 할 사항

스키마에 ID 토큰 매핑

Verified Permissions는 ID 토큰 클레임을 사용자의 이름 및 제목, 그룹 멤버십, 연락처 정보 등 사용자의 속성으로 처리합니다. ID 토큰은 속성 기반 액세스 제어(ABAC) 권한 부여 모델에서 가장 유용합니다. Verified Permissions가 요청자를 기반으로 리소스에 대한 액세스를 분석하도록 하려면 자격 증명 소스의 ID 토큰을 선택합니다.

Amazon Cognito ID 토큰은 대부분의 OIDC 신뢰 당사자 라이브러리에서 작동합니다. 추가 클레임으로 OIDC의 기능을 확장합니다. 애플리케이션은 Amazon Cognito 사용자 풀 인증 API 작업 또는 사용자 풀 호스팅 UI를 사용하여 사용자를 인증할 수 있습니다. 자세한 내용은 Amazon Cognito 개발자 안내서의 API 및 엔드포인트 사용을 참조하세요.

Amazon Cognito ID 토큰의 유용한 클레임

cognito:username 및 *preferred_username*

사용자 이름의 변형입니다.

sub

사용자의 고유 사용자 식별자(UUID)

custom: 접두사가 있는 클레임

와 같은 사용자 지정 사용자 풀 속성의 접두사입니다`custom:employmentStoreCode`.

표준 클레임

`email` 및와 같은 표준 OIDC 클레임`phone_number`. 자세한 내용은 오류 세트 2를 통합하는 OpenID Connect Core 1.0의 표준 클레임을 참조하세요.

cognito:groups

사용자의 그룹 멤버십입니다. 역할 기반 액세스 제어(RBAC)를 기반으로 하는 권한 부여 모델에서 이 클레임은 정책에서 평가할 수 있는 역할을 제공합니다.

일시적 클레임

사용자의 속성은 아니지만 사용자 풀 사전 토큰 생성 Lambda 트리거에 의해 런타임에 추가되는 클레임입니다. 일시적 클레임은 표준 클레임과 유사하지만 tenant 또는와 같이 표준을 벗어납니다department.

: 구분자가 있는 Amazon Cognito 속성을 참조하는 정책에서 형식의 속성을 참조합니다principal["*cognito:username*"]. 역할 클레임cognito:groups은이 규칙에 대한 예외입니다. Verified Permissions는이 클레임의 내용을 사용자 엔터티의 상위 엔터티에 매핑합니다.

Amazon Cognito 사용자 풀의 ID 토큰 구조에 대한 자세한 내용은 Amazon Cognito 개발자 안내서의 ID 토큰 사용을 참조하세요.

다음 예제 ID 토큰에는 네 가지 유형의 속성이 각각 있습니다. 여기에는 Amazon Cognito특정 클레임 cognito:username, 사용자 지정 클레임 custom:employmentStoreCode, 표준 클레임 email, 임시 클레임가 포함됩니다tenant.

```
{
    "sub": "91eb4550-XXX",
    "cognito:groups": [
        "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-TOKEN1111111",
    "email": "alice@example.com"
}
```

Amazon Cognito 사용자 풀을 사용하여 자격 증명 소스를 생성할 때 Verified Permissions가를 사용하여 권한 부여 요청에서 생성하는 보안 주체 엔터티의 유형을 지정합니다`IsAuthorizedWithToken`. 그러면 정책을 통해 해당 요청 평가의 일환으로 해당 보안 주체의 속성을 테스트할 수 있습니다. 스키마는 자격 증명 소스의 보안 주체 유형과 속성을 정의한 다음 Cedar 정책에서 참조할 수 있습니다.

ID 토큰 그룹 클레임에서 파생하려는 그룹 엔터티의 유형도 지정합니다. 권한 부여 요청에서 Verified Permissions는 그룹 클레임의 각 구성원을 해당 그룹 엔터티 유형에 매핑합니다. 정책에서 해당 그룹 개체를 보안 주체로 참조할 수 있습니다.

다음 예는 Verified Permissions 스키마에 자격 증명 토큰 예제의 속성을 반영하는 방법을 보여줍니다. 스키마 편집에 대한 자세한 내용은 정책 스토어 스키마 편집을(를) 참조하세요. 자격 증명 소스 구성에서 보안 주체 유형 `User`를 지정하면 다음 예제와 유사한 내용을 포함하여 Cedar에서 해당 속성을 사용할 수 있도록 할 수 있습니다.

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

이 스키마에 대해 검증할 정책 예제는 섹션을 참조하세요Amazon Cognito ID 토큰 속성을 반영합니다..

액세스 토큰 매핑

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

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

Amazon Cognito 액세스 토큰의 유용한 클레임

client_id

OIDC 신뢰 당사자에 대한 클라이언트 애플리케이션의 ID입니다. 클라이언트 ID를 사용하여 Verified Permissions는 권한 부여 요청이 정책 스토어에 허용된 클라이언트에서 온 것인지 확인할 수 있습니다. M2M(Machinemachine-to-machine) 권한 부여에서 요청 시스템은 클라이언트 보안 암호로 요청을 승인하고 클라이언트 ID와 범위를 권한 부여의 증거로 제공합니다.

scope

토큰 보유자의 액세스 권한을 나타내는 OAuth 2.0 범위입니다.

cognito:groups

사용자의 그룹 멤버십입니다. 역할 기반 액세스 제어(RBAC)를 기반으로 하는 권한 부여 모델에서 이 클레임은 정책에서 평가할 수 있는 역할을 제공합니다.

일시적 클레임

액세스 권한은 아니지만 사용자 풀 사전 토큰 생성 Lambda 트리거에 의해 런타임에 추가되는 클레임입니다. 일시적 클레임은 표준 클레임과 유사하지만 `tenant` 또는와 같이 표준을 벗어납니다`department`. 액세스 토큰을 사용자 지정하면 AWS 청구서에 비용이 추가됩니다.

Amazon Cognito 사용자 풀의 액세스 토큰 구조에 대한 자세한 내용은 Amazon Cognito 개발자 안내서의 액세스 토큰 사용을 참조하세요.

Amazon Cognito 액세스 토큰은 Verified Permissions에 전달될 때 컨텍스트 객체에 매핑됩니다. `context.token.`*`attribute_name`*을 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-TOKEN1111111",
    "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 정책 스토어 스키마 편집을(를) 참조하세요.

```
{
   "MyApplication": {
      "actions": {
         "Read": {
            "appliesTo": {
               "context": {
                  "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 스키마에 대해 검증할 정책 예제는 섹션을 참조하세요Amazon Cognito 액세스 토큰 속성을 반영합니다..

Amazon Cognito 콜론으로 구분된 클레임에 대한 대체 표기법

Verified Permissions가 시작된 시점에 `cognito:groups`와 같은 Amazon Cognito 토큰 클레임에 권장되는 스키마는 `.` 문자를 계층 구분 기호로 사용하도록 콜론으로 구분된 문자열을 `custom:store` 변환했습니다. 이 형식을 점 표기법이라고 합니다. 예를 들어에 대한 참조가 정책에 `cognito:groups` `principal.cognito.groups` 추가되었습니다. 이 형식을 계속 사용할 수 있지만 대괄호 표기법으로 스키마와 정책을 빌드하는 것이 좋습니다. 이 형식에서는에 대한 참조가 정책에 `cognito:groups` `principal["cognito:groups"]` 포함됩니다. Verified Permissions 콘솔에서 사용자 풀 ID 토큰에 대해 자동으로 생성된 스키마는 대괄호 표기법을 사용합니다.

자격 증명 소스에 대해 Amazon Cognito 수동으로 빌드된 스키마 및 정책에서 점 표기법을 계속 사용할 수 있습니다. 다른 유형의 OIDC IdP에 대해 스키마 `:` 또는 정책에 또는 다른 영숫자가 아닌 문자와 함께 점 표기법을 사용할 수 없습니다.

점 표기법에 대한 스키마는 다음 예제와 같이 `:` 캐릭터의 각 인스턴스를 `cognito` 또는 `custom` 초기 구문의 하위 항목으로 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 스키마에 대해 검증하고 점 표기법을 사용하는 정책 예제는 섹션을 참조하세요점 표기법을 사용하여 속성 참조.

스키마 매핑에 대해 알아야 할 사항

속성 매핑은 토큰 유형마다 다릅니다.

액세스 토큰 권한 부여에서 Verified Permissions는 클레임을 컨텍스트에 매핑합니다. ID 토큰 권한 부여에서 Verified Permissions는 클레임을 보안 주체 속성에 매핑합니다. Verified Permissions 콘솔에서

생성하는 정책 스토어의 경우 빈 정책 스토어와 샘플 정책 스토어만 자격 증명 소스가 없어지고 ID 토큰 권한 부여를 위해 사용자 풀 속성으로 스키마를 채워야 합니다. 액세스 토큰 권한 부여는 그룹 멤버십 클레임이 있는 역할 기반 액세스 제어(RBAC)를 기반으로 하며 다른 클레임을 정책 스토어 스키마에 자동으로 매핑하지 않습니다.

자격 증명 소스 속성은 필요하지 않습니다.

Verified Permissions 콘솔에서 자격 증명 소스를 생성하면 속성이 필수로 표시되지 않습니다. 이렇게 하면 누락된 클레임으로 인해 권한 부여 요청에 검증 오류가 발생하지 않습니다. 필요에 따라 속성을 필수로 설정할 수 있지만 모든 권한 부여 요청에 있어야 합니다.

RBAC는 스키마에 속성이 필요하지 않습니다.

자격 증명 소스의 스키마는 자격 증명 소스를 추가할 때 수행하는 엔터티 연결에 따라 달라집니다. 자격 증명 소스는 하나의 클레임을 사용자 엔터티 유형에 매핑하고 하나의 클레임을 그룹 엔터티 유형에 매핑합니다. 이러한 엔터티 매핑은 자격 증명 소스 구성의 핵심입니다. 이 최소 정보를 사용하여 역할 기반 액세스 제어(RBAC) 모델에서 특정 사용자 및 사용자가 구성원일 수 있는 특정 그룹에 대한 권한 부여 작업을 수행하는 정책을 작성할 수 있습니다. 스키마에 토큰 클레임을 추가하면 정책 스토어의 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 공급자의 추가 액세스 제어 정보를 제공할 수 있지만 추가 스키마 수정이 필요한 OAuth 2.0 범위와 같은 정보가 있습니다.

Verified Permissions 콘솔의 API Gateway 및 자격 증명 공급자로 설정 및 안내 설정 옵션은 스키마에 ID 토큰 클레임을 할당합니다. 액세스 토큰 클레임의 경우에는 그렇지 않습니다. 스키마에 비그룹 액세스 토큰 클레임을 추가하려면 JSON 모드에서 스키마를 편집하고 commonTypes 속성을 추가해야 합니다. 자세한 내용은 액세스 토큰 매핑 단원을 참조하십시오.

토큰 유형 선택

정책 스토어가 자격 증명 소스와 함께 작동하는 방식은 ID 또는 액세스 토큰을 처리할지 여부와 같은 자격 증명 소스 구성의 주요 결정에 따라 달라집니다. 자격 Amazon Cognito 증명 공급자를 사용하면 API 연결 정책 스토어를 생성할 때 토큰 유형을 선택할 수 있습니다. API 연결 정책 스토어를 생성할 때 ID 또는 액세스 토큰에 대한 권한 부여를 설정할지 여부를 선택해야 합니다. 이 정보는 Verified Permissions가 정책 스토어에 적용하는 스키마 속성과 API Gateway API에 대한 Lambda 권한 부여자의 구문에 영향을 줍니다. 특히 ID 토큰 클레임을 Verified Permissions 콘솔의 속성에 자동으로 매핑하여 혜택을 받으려면 ID 소스를 생성하기 전에 처리하려는 토큰 유형을 조기에 결정해야 합니다. 토큰 유형을 변경하려면 정책과 스키마를 리팩터링하는 데 상당한 노력이 필요합니다. 다음 주제에서는 정책 스토어에서 ID 및 액세스 토큰을 사용하는 방법을 설명합니다.

Cedar 구문 분석기에는 일부 문자에 대해 대괄호가 필요합니다.

정책은 일반적으로와 같은 형식의 스키마 속성을 참조합니다`principal.username`. 토큰 클레임 이름에 나타날 수 / 있는 `:`, `.`또는와 같은 영숫자가 아닌 대부분의 문자의 경우 Verified Permissions는 `principal.cognito:username` 또는와 같은 조건 값을 구문 분석할 수 없습니다`context.ip-address`. 대신 대괄호 표기법으로 이러한 조건의 형식을 `context["ip-address"]`각각 `principal["cognito:username"]` 또는 형식으로 지정해야 합니다. 밑줄 문자`_`는 클레임 이름에 유효한 문자이며이 요구 사항에 대한 유일한 영숫자가 아닌 예외입니다.

이 유형의 보안 주체 속성에 대한 부분 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 부분 예제 스키마는 다음과 같습니다.

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

```
                "type": "String"
            }
    }
    },
       "principalTypes": [
          "User"
       ]
    }
}
```

이 스키마에 대해 검증할 정책 예제는 섹션을 참조하세요대괄호 표기법을 사용하여 토큰 속성 참조.

에 대한 클라이언트 및 대상 검증 Amazon Cognito

정책 스토어에 자격 증명 소스를 추가하면 Verified Permissions에는 ID 및 액세스 토큰의 클라이언트 ID와 대상을 검증하고 `token_use` 클레임을 확인하는 구성 옵션이 있습니다. 이 검증은 `IsAuthorizedWithToken` 및 `BatchIsAuthorizedWithToken` API 요청을 처리할 때 발생합니다. 동작은 ID와 액세스 토큰, 및 Amazon Cognito OIDC 자격 증명 소스 간에 다릅니다. Amazon Cognito 사용자 풀 공급자를 사용하면 Verified Permissions가 ID 및 액세스 토큰 모두에서 클라이언트 ID를 검증할 수 있습니다. OIDC 공급자를 통해 Verified Permissions는 ID 토큰의 클라이언트 ID와 액세스 토큰의 대상을 검증할 수 있습니다.

클라이언트 ID는 애플리케이션이 사용하는 자격 증명 공급자 인스턴스와 연결된 식별자입니다. 예: `1example23456789`. 대상은와 같은 액세스 토큰의 의도된 신뢰 당사자 또는 대상과 연결된 URL 경로입니다`https://mytoken.example.com`. 액세스 토큰을 사용할 때 `aud` 클레임은 항상 대상과 연결됩니다.

Amazon Cognito ID 토큰에는 앱 클라이언트 ID가 포함된 `aud` 클레임이 있습니다. 액세스 토큰에는 앱 클라이언트 ID도 포함된 `client_id` 클레임이 있습니다.

자격 증명 소스에 클라이언트 애플리케이션 검증을 위한 값을 하나 이상 입력하면 Verified Permissions는이 앱 클라이언트 IDs 목록을 ID 토큰 `aud` 클레임 또는 액세스 토큰 `client_id` 클레임과 비교합니다. Verified Permissions는 자격 Amazon Cognito 증명 소스에 대한 신뢰 당사자 대상 URL을 검증하지 않습니다.

JWTs 대한 클라이언트 측 권한 부여

정책 스토어 자격 증명 소스를 사용하지 않고 애플리케이션에서 JSON 웹 토큰을 처리하고 해당 클레임을 Verified Permissions에 전달할 수 있습니다. JSON 웹 토큰(JWT)에서 개체 속성을 추출하여 Verified Permissions로 구문 분석할 수 있습니다.

이 예제는 JWT.1을 사용하여 애플리케이션에서 Verified Permissions를 호출하는 방법을 보여줍니다.

```
async function authorizeUsingJwtToken(jwtToken) {

    const payload = await verifier.verify(jwtToken);

    let 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
    };
    let 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
    };
    let 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
    };
    let entities = {
        entityList: [],
    };
    entities.entityList.push(...getUserEntitiesFromToken(payload));
    let policyStoreId = "PSEXAMPLEabcdefg111111"; // set your own policy store id

    const authResult = await client
        .isAuthorized({
        policyStoreId: policyStoreId,
        principal: principalEntity,
        resource: resourceEntity,
        action: action,
        entities,
        })
        .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",
      entityId: payload["sub"], // the application needs to use the claim that
 represents the user-id
    }
  };
  return [entityItem];
}
```

[1] 이 코드 예제는 aws-jwt-verify 라이브러리를 사용하여 OIDC 호환 IdP가 서명한 JWT를 검증합니다.

OIDC 자격 증명 소스 작업

규정 준수 OpenID Connect(OIDC) IdP를 정책 스토어의 자격 증명 소스로 구성할 수도 있습니다. OIDC 공급자는 Amazon Cognito 사용자 풀과 유사합니다. 인증의 결과로 JWTs를 생성합니다. OIDC 공급자를 추가하려면 발급자 URL을 제공해야 합니다.

새 OIDC 자격 증명 소스에는 다음 정보가 필요합니다.

- 발급자 URL입니다. Verified Permissions는이 URL에서 `.well-known/openid-configuration` 엔드포인트를 검색할 수 있어야 합니다.
- 와일드카드가 포함되지 않은 CNAME 레코드입니다. 예를 들어는에 매핑할 `a.example.com` 수 없습니다`*.example.net`. 반대로는에 매핑할 `*.example.com` 수 없습니다`a.example.net`.
- 권한 부여 요청에 사용할 토큰 유형입니다. 이 경우 자격 증명 토큰을 선택합니다.
- 와 같이 자격 증명 소스와 연결할 사용자 엔터티 유형입니다`MyCorp::User`.
- 와 같이 자격 증명 소스와 연결할 그룹 엔터티 유형입니다`MyCorp::UserGroup`.
- 예제 ID 토큰 또는 ID 토큰의 클레임 정의입니다.
- 사용자 및 그룹 개체 IDs. CLI 및 API에서이 접두사를 선택할 수 있습니다. API Gateway 설정 및 자격 증명 공급자 또는 안내 설정 옵션으로 생성한 정책 스토어에서 Verified Permissions는 발급자 이름의 접두사에서를 뺀 접두사를 할당합니다. 예를 들면 `https://`입니다`MyCorp::User::"auth.example.com|a1b2c3d4-5678-90ab-cdef-EXAMPLE11111"`.

API 작업을 사용하여 OIDC 소스의 요청을 승인하는 방법에 대한 자세한 내용은 섹션을 참조하세요권한 부여에 사용 가능한 API 작업.

다음 예제에서는 회계 부서의 직원에 대한 연말 보고서에 대한 액세스를 허용하고 기밀 분류가 있으며 위성 사무실에 있지 않은 정책을 생성하는 방법을 보여줍니다. Verified Permissions는 보안 주체의 ID 토큰에 있는 클레임에서 이러한 속성을 파생합니다.

보안 주체의 그룹을 참조할 때 정책을 올바르게 평가하려면 `in` 연산자를 사용해야 합니다.

```
permit(
     principal in MyCorp::UserGroup::"MyOIDCProvider|Accounting",
     action,
     resource in MyCorp::Folder::"YearEnd2024"
) when {
```

```
    principal.jobClassification == "Confidential" &&
    !(principal.location like "SatelliteOffice*")
};
```

주제

- Amazon Verified Permissions OIDC 자격 증명 소스 생성
- Amazon Verified Permissions OIDC 자격 증명 소스 편집
- 스키마에 OIDC 토큰 매핑
- OIDC 공급자를 위한 클라이언트 및 대상 검증

Amazon Verified Permissions OIDC 자격 증명 소스 생성

다음 절차에서는 기존 정책 스토어에 자격 증명 소스를 추가합니다.

Verified Permissions 콘솔에서 새 정책 스토어를 생성할 때 자격 증명 소스를 생성할 수도 있습니다. 이 프로세스에서는 자격 증명 소스 토큰의 클레임을 개체 속성으로 자동으로 가져올 수 있습니다. 안내 설정 또는 API Gateway 설정 및 자격 증명 공급자 옵션을 선택합니다. 이러한 옵션은 초기 정책도 생성합니다.

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

에서 create-identity-source를 사용하여 자격 증명 소스를 생성 AWS CLI 하거나 Verified Permissions API에서 CreateIdentitySource를 사용하여 자격 증명 소스를 생성할 때 보안 주체 엔터티 유형을 제외할 수 있습니다. 그러나 빈 엔터티 유형은 엔터티 유형이 인 ID 소스를 생성합니다`AWS::Cognito`. 이 엔터티 이름은 정책 스토어 스키마와 호환되지 않습니다. 자격 증명을 정책 스토어 스키마와 통합하려면 보안 주체 Amazon Cognito 엔터티 유형을 지원되는 정책 스토어 엔터티로 설정해야 합니다.

AWS Management Console

OpenID Connect(OIDC) 자격 증명 소스를 생성하려면

1. Verified Permissions 콘솔을 엽니다. 정책 스토어를 선택합니다.
2. 왼쪽 탐색 창에서 자격 증명 소스를 선택합니다.

3. 자격 증명 소스 생성을 선택합니다.
4. 외부 OIDC 공급자를 선택합니다.
5. 발급자 URL에 OIDC 발급자의 URL을 입력합니다. 이는 권한 부여 서버, 서명 키 및와 같은 공급자에 대한 기타 정보를 제공하는 서비스 엔드포인트입니다`https://auth.example.com`. 발급자 URL은에서 OIDC 검색 문서를 호스팅해야 합니다`/.well-known/openid-configuration`.
6. 토큰 유형에서 애플리케이션이 권한 부여를 위해 제출할 OIDC JWT 유형을 선택합니다. 자세한 내용은 스키마에 OIDC 토큰 매핑 단원을 참조하십시오.
7. 스키마 엔터티에 토큰 클레임 매핑에서 자격 증명 소스에 대한 사용자 엔터티 및 사용자 클레임을 선택합니다. 사용자 엔터티는 OIDC 공급자의 사용자를 참조하려는 정책 스토어의 엔터티입니다. 사용자 클레임은 일반적으로 평가할 엔터티의 고유 식별자를 포함하는 ID 또는 액세스 토큰`sub`의 클레임입니다. 연결된 OIDC IdP의 ID는 선택한 보안 주체 유형에 매핑됩니다.
8. (선택 사항) 스키마 엔터티에 토큰 클레임 매핑에서 자격 증명 소스에 대한 그룹 엔터티 및 그룹 클레임을 선택합니다. 그룹 개체는 사용자 개체의 상위 개체입니다. 그룹 클레임이이 엔터티에 매핑됩니다. 그룹 클레임은 일반적으로 평가할 엔터티`groups`에 대한 사용자 그룹 이름의 문자열, JSON 또는 공백으로 구분된 문자열이 포함된 ID 또는 액세스 토큰의 클레임입니다. 연결된 OIDC IdP의 ID는 선택한 보안 주체 유형에 매핑됩니다.
9. 검증 - 선택 사항에서 권한 부여 요청에 정책 스토어가 수락할 클라이언트 IDs 또는 대상 URLs 있는 경우 입력합니다.
10. 자격 증명 소스 생성을 선택합니다.
11. (선택 사항) 정책 스토어에 스키마가 있는 경우 Cedar 정책의 자격 증명 또는 액세스 토큰에서 추출한 속성을 참조하려면 먼저 Cedar가 자격 증명 소스가 생성하는 보안 주체 유형을 인식하도록 스키마를 업데이트해야 합니다. 스키마에 추가되는 항목에는 Cedar 정책에서 참조하려는 속성이 포함되어야 합니다. OIDC 토큰 속성을 Cedar 보안 주체 속성에 매핑하는 방법에 대한 자세한 내용은 섹션을 참조하세요스키마에 OIDC 토큰 매핑.
12. 토큰의 정보를 사용하여 권한 부여 결정을 내리는 정책을 생성합니다. 자세한 내용은 Amazon Verified Permissions 정적 정책 생성 단원을 참조하십시오.

이제 자격 증명 소스를 생성하고, 스키마를 업데이트하고, 정책을 생성했으므로 `IsAuthorizedWithToken`를 사용하여 Verified Permissions가 권한 부여 결정을 내리도록 합니다. 자세한 내용은 Amazon Verified Permissions API 참조 안내서의 IsAuthorizedWithToken을 참조하세요.

AWS CLI

OIDC 자격 증명 소스를 생성하려면

CreateIdentitySource 작업을 사용하여 자격 증명 소스를 생성할 수 있습니다. 다음 예제에서는 OIDC 자격 증명 공급자(IdP)의 인증된 자격 증명에 액세스할 수 있는 자격 증명 소스를 생성합니다.

1. `create-identity-source` 명령의 `--configuration` 파라미터에서 사용할 OIDC IdP에 대한 다음 세부 정보가 포함된 `config.txt` 파일을 생성합니다.

```
{
    "openIdConnectConfiguration": {
        "issuer": "https://auth.example.com",
        "tokenSelection": {
                "identityTokenOnly": {
                        "clientIds":["1example23456789"],
                        "principalIdClaim": "sub"
                },
        },
        "entityIdPrefix": "MyOIDCProvider",
        "groupConfiguration": {
              "groupClaim": "groups",
              "groupEntityType": "MyCorp::UserGroup"
        }
    }
}
```

2. 다음 명령을 실행하여 OIDC 자격 증명 소스를 생성합니다.

```
$ 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"
}
```

3. (선택 사항) 정책 스토어에 스키마가 있는 경우 Cedar 정책의 자격 증명 또는 액세스 토큰에서 추출한 속성을 참조하려면 먼저 Cedar가 자격 증명 소스가 생성하는 보안 주체 유형을 인식하도록 스키마를 업데이트해야 합니다. 스키마에 추가되는 항목에는 Cedar 정책에서 참조하려는 속성이 포함되어야 합니다. OIDC 토큰 속성을 Cedar 보안 주체 속성에 매핑하는 방법에 대한 자세한 내용은 섹션을 참조하세요스키마에 OIDC 토큰 매핑.

4. 토큰의 정보를 사용하여 권한 부여 결정을 내리는 정책을 생성합니다. 자세한 내용은 Amazon Verified Permissions 정적 정책 생성 단원을 참조하십시오.

이제 자격 증명 소스를 생성하고, 스키마를 업데이트하고, 정책을 생성했으므로 `IsAuthorizedWithToken`를 사용하여 Verified Permissions가 권한 부여 결정을 내리도록 합니다. 자세한 내용은 Amazon Verified Permissions API 참조 안내서의 IsAuthorizedWithToken을 참조하세요.

Amazon Verified Permissions OIDC 자격 증명 소스 편집

자격 증명 소스를 생성한 후 일부 파라미터를 편집할 수 있습니다. 자격 증명 소스 유형은 변경할 수 없으며 자격 증명 소스를 삭제하고에서 OIDC 또는 OIDC Amazon Cognito 로 전환할 새 소스를 생성해야 합니다 Amazon Cognito. 정책 스토어 스키마가 자격 증명 소스 속성과 일치하는 경우 자격 증명 소스에 대한 변경 사항을 반영하도록 스키마를 별도로 업데이트해야 합니다.

AWS Management Console

OIDC 자격 증명 소스를 업데이트하려면

1. Verified Permissions 콘솔을 엽니다. 정책 스토어를 선택합니다.
2. 왼쪽 탐색 창에서 자격 증명 소스를 선택합니다.
3. 편집할 자격 증명 소스의 ID를 선택합니다.
4. 편집을 선택합니다.
5. OIDC 공급자 세부 정보에서 필요에 따라 발급자 URL을 변경합니다.
6. 토큰 클레임을 스키마 속성에 매핑에서 필요에 따라 사용자 및 그룹 클레임과 정책 스토어 엔터티 유형 간의 연결을 변경합니다. 개체 유형을 변경한 후에는 정책 및 스키마 속성을 업데이트하여 새 개체 유형에 적용해야 합니다.
7. 대상 검증에서 적용하려는 대상 값을 추가하거나 제거합니다.
8. 변경 사항 저장을 선택합니다.

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

AWS CLI

OIDC 자격 증명 소스를 업데이트하려면

UpdateIdentitySource 작업을 사용하여 자격 증명 소스를 업데이트할 수 있습니다. 다음 예시에서는 다른 OIDC 공급자를 사용하도록 지정된 자격 증명 소스를 업데이트합니다.

1. update-identity-source 명령의 --configuration 파라미터에서 사용할 OIDC IdP에 대한 다음 세부 정보가 포함된 config.txt 파일을 생성합니다.

```
{
    "openIdConnectConfiguration": {
        "issuer": "https://auth2.example.com",
        "tokenSelection": {
                "identityTokenOnly": {
                        "clientIds":["2example10111213"],
                        "principalIdClaim": "sub"
                },
        },
        "entityIdPrefix": "MyOIDCProvider",
        "groupConfiguration": {
              "groupClaim": "groups",
              "groupEntityType": "MyCorp::UserGroup"
        }
    }
}
```

2. 다음 명령을 실행하여 OIDC 자격 증명 소스를 업데이트합니다.

```
$ 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"
}
```

> **Note**
>
> 자격 증명 소스의 보안 주체 유형을 변경하는 경우 업데이트된 보안 주체 유형을 올바르게 반영하도록 스키마를 업데이트해야 합니다.

스키마에 OIDC 토큰 매핑

정책 스토어에 자격 증명 소스를 추가하고 정책 스토어 스키마에 공급자 클레임 또는 토큰을 매핑하려고 할 수 있습니다. 가이드 설정을 사용하여 자격 증명 소스로 정책 스토어를 생성하거나 정책 스토어가 생성된 후 스키마를 수동으로 업데이트하여이 프로세스를 자동화할 수 있습니다. 토큰을 스키마에 매핑한 후에는 토큰을 참조하는 정책을 생성할 수 있습니다.

사용 설명서의이 섹션에는 다음 정보가 포함되어 있습니다.

- 정책 스토어 스키마에 속성을 자동으로 채울 수 있는 경우
- 자격 증명 소스에 대한 스키마를 수동으로 빌드하는 방법

API 연결 정책 저장소 및 안내형 설정을 통해 생성된 자격 증명 소스가 있는 정책 저장소는 스키마에 자격 증명(ID) 토큰 속성을 수동으로 매핑할 필요가 없습니다. Verified Permissions에 사용자 풀의 속성을 제공하고 사용자 속성으로 채워진 스키마를 생성할 수 있습니다. ID 토큰 권한 부여에서 Verified Permissions는 클레임을 보안 주체 엔터티의 속성에 매핑합니다.

Verified Permissions 정책 스토어에서 OIDC ID 제공업체(IdP)를 ID 소스로 사용하려면 스키마에 제공업체 속성이 있어야 합니다. 스키마는 고정되며 공급자 토큰이 IsAuthorizedWithToken 또는 BatchIsAuthorizedWithToken API 요청에서 생성하는 엔터티와 일치해야 합니다. ID 토큰의 공급자 정보에서 스키마를 자동으로 채우는 방식으로 정책 스토어를 생성한 경우 정책을 작성할 준비가 된 것입니다. 자격 증명 소스에 대한 스키마 없이 정책 스토어를 생성하는 경우 API 요청을 사용하여 생성된 엔터티와 일치하는 스키마에 공급자 속성을 추가해야 합니다. 그런 다음 공급자 토큰의 속성을 사용하여 정책을 작성할 수 있습니다.

주제

- 스키마에 ID 토큰 매핑
- 액세스 토큰 매핑
- 스키마 매핑에 대해 알아야 할 사항

스키마에 ID 토큰 매핑

Verified Permissions는 ID 토큰 클레임을 사용자의 이름 및 제목, 그룹 멤버십, 연락처 정보 등 사용자의 속성으로 처리합니다. ID 토큰은 속성 기반 액세스 제어(ABAC) 권한 부여 모델에서 가장 유용합니다. Verified Permissions가 요청자를 기반으로 리소스에 대한 액세스를 분석하도록 하려면 자격 증명 소스의 ID 토큰을 선택합니다.

OIDC 공급자의 ID 토큰 작업은 Amazon Cognito ID 토큰 작업과 거의 동일합니다. 차이점은 클레임에 있습니다. IdP는 표준 OIDC 속성을 제공하거나 사용자 지정 스키마를 가질 수 있습니다. Verified Permissions 콘솔에서 새 정책 스토어를 생성할 때 예제 ID 토큰을 사용하여 OIDC 자격 증명 소스를 추가하거나 토큰 클레임을 사용자 속성에 수동으로 매핑할 수 있습니다. Verified Permissions는 IdP의 속성 스키마를 인식하지 못하므로이 정보를 제공해야 합니다.

자세한 내용은 Verified Permissions 정책 스토어 생성 단원을 참조하십시오.

다음은 OIDC 자격 증명 소스가 있는 정책 스토어의 스키마 예제입니다.

```
"User": {
   "shape": {
      "type": "Record",
      "attributes": {
         "email": {
            "type": "String"
         },
         "email_verified": {
            "type": "Boolean"
         },
         "name": {
            "type": "String",
            "required": true
         },
         "phone_number": {
            "type": "String"
         },
         "phone_number_verified": {
            "type": "Boolean"
         }
      }
   }
}
```

이 스키마에 대해 검증할 정책 예제는 섹션을 참조하세요OIDC ID 토큰 속성을 반영합니다..

액세스 토큰 매핑

Verified Permissions는 그룹 클레임 이외의 액세스 토큰 클레임을 작업의 속성 또는 컨텍스트 속성으로 처리합니다. 그룹 멤버십과 함께 IdP의 액세스 토큰에는 API 액세스에 대한 정보가 포함될 수 있습니다. 액세스 토큰은 역할 기반 액세스 제어(RBAC)를 사용하는 권한 부여 모델에서 유용합니다. 그룹 멤버십 이외의 액세스 토큰 클레임에 의존하는 권한 부여 모델은 스키마 구성에 추가 노력이 필요합니다.

외부 OIDC 공급자의 대부분의 액세스 토큰은 Amazon Cognito 액세스 토큰과 밀접하게 일치합니다. OIDC 액세스 토큰은 Verified Permissions에 전달될 때 컨텍스트 객체에 매핑됩니다. `context.token.`*`attribute_name`*을 사용하여 액세스 토큰의 속성을 참조할 수 있습니다. 다음 예제 OIDC 액세스 토큰에는 기본 클레임 예제가 포함되어 있습니다.

```
{
    "sub": "91eb4550-9091-708c-a7a6-9758ef8b6b1e",
    "groups": [
        "Store-Owner-Role",
        "Customer"
    ],
    "iss": "https://auth.example.com",
    "client_id": "1example23456789",
    "aud": "https://myapplication.example.com"
    "scope": "MyAPI-Read",
    "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 정책 스토어 스키마 편집을(를) 참조하세요.

```
{
   "MyApplication": {
      "actions": {
         "Read": {
            "appliesTo": {
               "context": {
                  "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 스키마에 대해 검증할 정책 예제는 섹션을 참조하세요OIDC 액세스 토큰 속성을 반영합니다..

스키마 매핑에 대해 알아야 할 사항

속성 매핑은 토큰 유형마다 다릅니다.

액세스 토큰 권한 부여에서 Verified Permissions는 클레임을 컨텍스트에 매핑합니다. ID 토큰 권한 부여에서 Verified Permissions는 클레임을 보안 주체 속성에 매핑합니다. Verified Permissions 콘솔에서 생성하는 정책 스토어의 경우 빈 정책 스토어와 샘플 정책 스토어만 자격 증명 소스가 없어지고 ID 토큰 권한 부여를 위해 사용자 풀 속성으로 스키마를 채워야 합니다. 액세스 토큰 권한 부여는 그룹 멤버

십 클레임이 있는 역할 기반 액세스 제어(RBAC)를 기반으로 하며 다른 클레임을 정책 스토어 스키마에 자동으로 매핑하지 않습니다.

자격 증명 소스 속성은 필요하지 않습니다.

Verified Permissions 콘솔에서 자격 증명 소스를 생성하면 속성이 필수로 표시되지 않습니다. 이렇게 하면 누락된 클레임으로 인해 권한 부여 요청에 검증 오류가 발생하지 않습니다. 필요에 따라 속성을 필수로 설정할 수 있지만 모든 권한 부여 요청에 있어야 합니다.

RBAC는 스키마에 속성이 필요하지 않습니다.

자격 증명 소스의 스키마는 자격 증명 소스를 추가할 때 수행하는 엔터티 연결에 따라 달라집니다. 자격 증명 소스는 하나의 클레임을 사용자 엔터티 유형에 매핑하고 하나의 클레임을 그룹 엔터티 유형에 매핑합니다. 이러한 엔터티 매핑은 자격 증명 소스 구성의 핵심입니다. 이 최소 정보를 사용하여 역할 기반 액세스 제어(RBAC) 모델에서 특정 사용자 및 사용자가 구성원일 수 있는 특정 그룹에 대한 권한 부여 작업을 수행하는 정책을 작성할 수 있습니다. 스키마에 토큰 클레임을 추가하면 정책 스토어의 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 공급자의 추가 액세스 제어 정보를 제공할 수 있지만 추가 스키마 수정이 필요한 OAuth 2.0 범위와 같은 정보가 있습니다.

Verified Permissions 콘솔의 API Gateway 설정 및 자격 증명 공급자 및 안내 설정 옵션은 스키마에 ID 토큰 클레임을 할당합니다. 액세스 토큰 클레임의 경우에는 그렇지 않습니다. 스키마에 비 그룹 액세스 토큰 클레임을 추가하려면 JSON 모드에서 스키마를 편집하고 commonTypes 속성을 추가해야 합니다. 자세한 내용은 액세스 토큰 매핑 단원을 참조하십시오.

OIDC 그룹 클레임은 여러 형식을 지원합니다.

OIDC 공급자를 추가할 때 정책 스토어의 사용자 그룹 멤버십에 매핑하려는 ID 또는 액세스 토큰에서 그룹 클레임의 이름을 선택할 수 있습니다. 확인된 권한은 그룹 클레임을 다음 형식으로 인식합니다.

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

> **Note**
>
> Verified Permissions는 공백으로 구분된 그룹 클레임의 각 문자열을 별도의 그룹으로 해석합니다. 공백 문자가 있는 그룹 이름을 단일 그룹으로 해석하려면 클레임에서 공백을 바꾸거나 제거합니다. 예를 들어 라는 이름의 그룹을 `My Group`로 포맷합니다`MyGroup`.

토큰 유형 선택

정책 스토어가 자격 증명 소스와 함께 작동하는 방법은 ID 또는 액세스 토큰을 처리할지 여부와 같은 자격 증명 소스 구성의 주요 결정에 따라 달라집니다. OIDC 공급자를 사용하면 자격 증명 소스를 추가할 때 토큰 유형을 선택해야 합니다. ID 또는 액세스 토큰을 선택할 수 있으며, 선택하지 않은 토큰 유형은 정책 스토어에서 처리되지 않습니다. 특히 ID 토큰 클레임을 Verified Permissions 콘솔의 속성에 자동으로 매핑하여 혜택을 받으려면 자격 증명 소스를 생성하기 전에 처리하려는 토큰 유형을 조기에 결정해야 합니다. 토큰 유형을 변경하려면 정책과 스키마를 리팩터링하는 데 상당한 노력이 필요합니다. 다음 주제에서는 정책 스토어에서 ID 및 액세스 토큰을 사용하는 방법을 설명합니다.

Cedar 구문 분석기에는 일부 문자에 대해 대괄호가 필요합니다.

정책은 일반적으로와 같은 형식의 스키마 속성을 참조합니다`principal.username`. 토큰 클레임 이름에 나타날 수 `/` 있는 `.`, 또는 `:`와 같은 영숫자가 아닌 대부분의 문자의 경우 Verified Permissions는 `principal.cognito:username` 또는와 같은 조건 값을 구문 분석할 수 없습니다`context.ip-address`. 대신 대괄호 표기법으로 이러한 조건의 형식을 `context["ip-address"]`각각 `principal["cognito:username"]` 또는 형식으로 지정해야 합니다. 밑줄 문자 `_`는 클레임 이름에 유효한 문자이며이 요구 사항에 대한 유일한 영숫자가 아닌 예외입니다.

이 유형의 보안 주체 속성에 대한 부분 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 부분 예제 스키마는 다음과 같습니다.

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

이 스키마에 대해 검증할 정책 예제는 섹션을 참조하세요대괄호 표기법을 사용하여 토큰 속성 참조.

OIDC 공급자를 위한 클라이언트 및 대상 검증

정책 스토어에 자격 증명 소스를 추가하면 Verified Permissions에는 ID 및 액세스 토큰의 클라이언트 ID와 대상을 검증하는 구성 옵션이 있습니다. 이 검증은 `IsAuthorizedWithToken` 및 `BatchIsAuthorizedWithToken` API 요청을 처리할 때 발생합니다. 동작은 ID와 액세스 토큰, 및 Amazon Cognito OIDC 자격 증명 소스 간에 다릅니다. Amazon Cognito 사용자 풀 공급자를 사용하면 Verified Permissions가 ID 및 액세스 토큰 모두에서 클라이언트 ID를 검증할 수 있습니다. OIDC 공급자를 통해 Verified Permissions는 ID 토큰의 클라이언트 ID와 액세스 토큰의 대상을 검증할 수 있습니다.

Amazon Cognito 토큰과 달리 OIDC 토큰에는 토큰이 ID 토큰인지 액세스 토큰인지를 나타내는 표준 클레임이 없습니다. 따라서 Verified Permissions는 OIDC 자격 증명 소스의 토큰 유형을 확인하지 않습니다. 정책 스토어가 각 `IsAuthorizedWithToken` 또는 `BatchIsAuthorizedWithToken` 요청에서 처리하도록 구성된 토큰 유형을 전달하는 것은 사용자의 책임입니다.

클라이언트 ID는 애플리케이션이 사용하는 자격 증명 공급자 인스턴스와 연결된 식별자입니다. 예: `1example23456789`. 대상은와 같은 액세스 토큰의 의도된 신뢰 당사자 또는 대상과 연결된 URL 경로입니다`https://mytoken.example.com`. 액세스 토큰을 사용할 때 `aud` 클레임은 항상 대상과 연결됩니다.

OIDC ID 토큰에는와 같은 클라이언트 ID가 포함된 `aud` 클레임이 있습니다`1example23456789`. IDs

OIDC 액세스 토큰에는와 같은 토큰의 대상 URL이 포함된 `aud` 클레임`https://myapplication.example.com`과와 같은 클라이언트 ID가 포함된 `client_id` 클레임이 있습니다`1example23456789`. IDs

정책 스토어를 설정할 때 정책 스토어가 토큰의 대상을 검증하는 데 사용하는 대상 검증 값을 하나 이상 입력합니다.

- ID 토큰 - Verified Permissions는 `aud` 클레임의 클라이언트 ID 중 하나 이상이 대상 검증 값과 일치하는지 확인하여 클라이언트 IDs를 검증합니다.
- 액세스 토큰 - Verified Permissions는 `aud` 클레임의 URL이 대상 검증 값과 일치하는지 확인하여 대상을 검증합니다. `aud` 클레임이 없는 경우 `cid` 또는 `client_id` 클레임을 사용하여 대상을 검증할 수 있습니다. 자격 증명 공급자에게 문의하여 올바른 대상 클레임 및 형식을 확인하세요.

JWTs 대한 클라이언트 측 권한 부여

정책 스토어 자격 증명 소스를 사용하지 않고 애플리케이션에서 JSON 웹 토큰을 처리하고 해당 클레임을 Verified Permissions에 전달할 수 있습니다. JSON 웹 토큰(JWT)에서 개체 속성을 추출하여 Verified Permissions로 구문 분석할 수 있습니다.

이 예제는 JWT.1을 사용하여 애플리케이션에서 Verified Permissions를 호출하는 방법을 보여줍니다.

```
async function authorizeUsingJwtToken(jwtToken) {

    const payload = await verifier.verify(jwtToken);

    let 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
    };
    let 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
    };
    let 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
    };
    let entities = {
        entityList: [],
    };
    entities.entityList.push(...getUserEntitiesFromToken(payload));
    let policyStoreId = "PSEXAMPLEabcdefg111111"; // set your own policy store id

    const authResult = await client
        .isAuthorized({
        policyStoreId: policyStoreId,
        principal: principalEntity,
        resource: resourceEntity,
        action: action,
        entities,
        })
        .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",
      entityId: payload["sub"], // the application needs to use the claim that
 represents the user-id
    }
  };
  return [entityItem];
}
```

[1] 이 코드 예제는 aws-jwt-verify 라이브러리를 사용하여 OIDC 호환 IdP가 서명한 JWT를 검증합니다.

Amazon Verified Permissions 통합

Amazon Verified Permissions 통합을 사용하면 코드와 프레임워크별 모범 사례를 최소화하면서 애플리케이션에서 세분화된 권한 부여를 구현할 수 있습니다. 이러한 통합은 애플리케이션을 Verified Permissions와 원활하게 연결하는 미들웨어 구성 요소와 유틸리티를 제공합니다.

통합을 통해 다음을 수행할 수 있습니다.

- 몇 분 안에 권한 부여 구현
- 프레임워크별 패턴 및 규칙 준수
- 유지 관리 오버헤드 감소
- 잠재적 보안 구현 오류 최소화
- 권한 부여 코드보다는 비즈니스 로직에 집중

애플리케이션에 추가되면 통합은 다음을 수행합니다.

1. 프레임워크별 미들웨어를 통해 수신 요청 차단
2. 요청에서 관련 권한 부여 컨텍스트 추출
3. Verified Permissions를 사용하여 권한 부여 결정 결정
4. 권한 부여 결과를 기반으로 액세스 제어 적용

Verified Permissions는 현재 다음 프레임워크를 지원합니다.

- Node.js 애플리케이션용 Express.js

Express와 Amazon Verified Permissions 통합

Verified Permissions Express 통합은 Express.js 애플리케이션에서 권한 부여를 구현하는 미들웨어 기반 접근 방식을 제공합니다. 이 통합을 통해 기존 라우팅 핸들러를 수정하지 않고도 세분화된 권한 부여 정책을 사용하여 API 엔드포인트를 보호할 수 있습니다. 통합은 요청을 가로채고, 정의된 정책에 따라 요청을 평가하고, 승인된 사용자만 보호된 리소스에 액세스할 수 있도록 하여 권한 부여 검사를 자동으로 처리합니다.

이 주제에서는 정책 스토어 생성부터 권한 부여 미들웨어 구현 및 테스트에 이르기까지 Express 통합을 설정하는 방법을 안내합니다. 이 단계를 따르면 코드 변경을 최소화하면서 Express 애플리케이션에 강력한 권한 부여 제어를 추가할 수 있습니다.

이 주제 전체에서 다음 GitHub리포지토리가 참조됩니다.

- cedar-policy/authorization-for-expressjs - Express.js용 Cedar 권한 부여 미들웨어
- verifiedpermissions/authorization-clients-js - JavaScript에 대한 Verified Permissions 권한 부여 클라이언트
- verifiedpermissions/examples/express-petstore - Express.js 미들웨어를 사용한 구현 예제

사전 조건

Express 통합을 구현하기 전에 다음을 갖추어야 합니다.

- Verified Permissions에 액세스할 수 있는 AWS 계정
- Node.js 및 npm 설치됨
- Express.js 애플리케이션
- OpenID Connect(OIDC) 자격 증명 공급자(예: Amazon Cognito)
- AWS CLI 적절한 권한으로 구성됨

통합 설정

1단계: 정책 스토어 생성

AWS CLI다음을 사용하여 정책 스토어를 생성합니다.

```
aws verifiedpermissions create-policy-store --validation-settings "mode=STRICT"
```

> **Note**
>
> 후속 단계에서 사용할 수 있도록 응답에 반환된 정책 스토어 ID를 저장합니다.

2단계: 종속성 설치

Express 애플리케이션에 필요한 패키지를 설치합니다.

```
npm i --save @verifiedpermissions/authorization-clients-js
npm i --save @cedar-policy/authorization-for-expressjs
```

권한 부여 구성

1단계: Cedar 스키마 생성 및 업로드

스키마는 애플리케이션의 엔터티 유형과 사용자가 수행할 수 있는 작업을 포함하여 애플리케이션에 대한 권한 부여 모델을 정의합니다. 스키마의 네임스페이스를 정의하는 것이 좋습니다. 이 예제에서는 `YourNamespace`을 사용합니다. 스키마를 Verified Permissions 정책 스토어에 연결하면 정책이 추가되거나 수정되면 서비스가 스키마에 대해 정책을 자동으로 검증합니다.

`@cedar-policy/authorization-for-expressjs` 패키지는 애플리케이션의 OpenAPI 사양을 분석하고 Cedar 스키마를 생성할 수 있습니다. 특히 경로 객체는 사양에 필요합니다.

OpenAPI 사양이 없는 경우 express-openapi-generator 패키지의 빠른 지침에 따라 OpenAPI 사양을 생성할 수 있습니다.

OpenAPI 사양에서 스키마를 생성합니다.

```
npx @cedar-policy/authorization-for-expressjs generate-schema --api-spec schemas/
openapi.json --namespace YourNamespace --mapping-type SimpleRest
```

그런 다음에 사용할 Cedar 스키마의 형식을 지정합니다 AWS CLI. 필요한 특정 형식에 대한 자세한 내용은 섹션을 참조하세요정책 스토어 스키마. 스키마 형식을 지정하는 데 도움이 필요한 경우 verifiedpermissions/examples GitHub리포지토리`prepare-cedar-schema.sh`에 라는 스크립트가 있습니다. 다음은 `v2.cedarschema.forAVP.json` 파일에서 Verified Permissions 형식의 스키마를 출력하는 해당 스크립트에 대한 호출 예제입니다.

```
./scripts/prepare-cedar-schema.sh v2.cedarschema.json v2.cedarschema.forAVP.json
```

형식이 지정된 스키마를 정책 스토어에 업로드하고를 정책 스토어 ID`policy-store-id`로 바꿉니다.

```
aws verifiedpermissions put-schema \
  --definition file://v2.cedarschema.forAVP.json \
  --policy-store-id policy-store-id
```

2단계: 권한 부여 정책 생성

정책이 구성되지 않은 경우 Cedar는 모든 권한 부여 요청을 거부합니다. Express 프레임워크 통합은 이전에 생성된 스키마를 기반으로 예제 정책을 생성하여이 프로세스를 부트스트랩하는 데 도움이 됩니다.

프로덕션 애플리케이션에서이 통합을 사용하는 경우 코드형 인프라(IaaC) 도구를 사용하여 새 정책을 생성하는 것이 좋습니다. 자세한 내용은 작업 AWS CloudFormation 단원을 참조하십시오.

샘플 Cedar 정책 생성:

```
npx @cedar-policy/authorization-for-expressjs generate-policies --schema
 v2.cedarschema.json
```

그러면 /policies 디렉터리에 샘플 정책이 생성됩니다. 그런 다음 사용 사례에 따라 이러한 정책을 사용자 지정할 수 있습니다. 예제:

```
// Defines permitted administrator user group actions
permit (
    principal in YourNamespace::UserGroup::"<userPoolId>|administrator",
    action,
    resource
);

// Defines permitted employee user group actions
permit (
    principal in YourNamespace::UserGroup::"<userPoolId>|employee",
    action in
        [YourNamespace::Action::"GET /resources",
         YourNamespace::Action::"POST /resources",
         YourNamespace::Action::"GET /resources/{resourceId}",
         YourNamespace::Action::"PUT /resources/{resourceId}"],
    resource
);
```

와 함께 사용할 정책의 형식을 지정합니다 AWS CLI. 필요한 형식에 대한 자세한 내용은 AWS CLI 참조의 create-policy를 참조하세요. 정책 형식 지정에 도움이 필요한 경우 verifiedpermissions/examples GitHub리포지토리convert_cedar_policies.sh에 라는 스크립트가 있습니다. 다음은 해당 스크립트에 대한 호출입니다.

```
./scripts/convert_cedar_policies.sh
```

형식이 지정된 정책을 Verified Permissions에 업로드하여 `policy_1.json`를 정책 파일의 경로 및 이름으로 바꾸고를 정책 스토어 ID`policy-store-id`로 바꿉니다.

```
aws verifiedpermissions create-policy \
  --definition file://policies/json/policy_1.json \
  --policy-store-id policy-store-id
```

3단계: 자격 증명 공급자 연결

기본적으로 Verified Permissions 권한 부여자 미들웨어는 사용자 정보를 가져오기 위해 API 요청의 권한 부여 헤더 내에 제공된 JSON 웹 토큰(JWT)을 읽습니다. Verified Permissions는 권한 부여 정책 평가를 수행하는 것 외에도 토큰을 검증할 수 있습니다.

`userPoolArn` 및를 사용하여 `clientId`다음과 `identity-source-configuration.txt` 같은 이름의 ID 소스 구성 파일을 생성합니다.

```
{
    "cognitoUserPoolConfiguration": {
        "userPoolArn": "arn:aws:cognito-idp:region:account:userpool/pool-id",
        "clientIds": ["client-id"],
        "groupConfiguration": {
            "groupEntityType": "YourNamespace::UserGroup"
        }
    }
}
```

다음 AWS CLI 명령을 실행하여 자격 증명 소스를 생성하고를 정책 스토어 ID`policy-store-id`로 바꿉니다.

```
aws verifiedpermissions create-identity-source \
  --configuration file://identity-source-configuration.txt \
  --policy-store-id policy-store-id \
  --principal-entity-type YourNamespace::User
```

권한 부여 미들웨어 구현

권한 부여 미들웨어를 포함하도록 Express 애플리케이션을 업데이트합니다. 이 예제에서는 자격 증명 토큰을 사용하지만 액세스 토큰도 사용할 수 있습니다. 자세한 내용은의 authorization-for-expressjs를 참조하세요GitHub.

```
const { ExpressAuthorizationMiddleware } = require('@cedar-policy/authorization-for-
expressjs');

const { AVPAuthorizationEngine } = require('@verifiedpermissions/authorization-
clients');

const avpAuthorizationEngine = new AVPAuthorizationEngine({
    policyStoreId: 'policy-store-id',
    callType: 'identityToken'
});

const expressAuthorization = new ExpressAuthorizationMiddleware({
    schema: {
        type: 'jsonString',
        schema: fs.readFileSync(path.join(__dirname, '../v4.cedarschema.json'),
 'utf8'),
    },
    authorizationEngine: avpAuthorizationEngine,
    principalConfiguration: { type: 'identityToken' },
    skippedEndpoints: [],
    logger: {
        debug: (s) => console.log(s),
        log: (s) => console.log(s),
    }
});

// Add the middleware to your Express application
app.use(expressAuthorization.middleware);
```

통합 테스트

다른 사용자 토큰을 사용하여 API 엔드포인트에 요청하여 권한 부여 구현을 테스트할 수 있습니다. 권한 부여 미들웨어는 정의된 정책을 기준으로 각 요청을 자동으로 평가합니다.

예를 들어 권한이 다른 다른 사용자 그룹을 설정한 경우:

- 관리자: 모든 리소스 및 관리 기능에 대한 전체 액세스
- 직원: 리소스를 보고, 생성하고, 업데이트할 수 있습니다.
- 고객: 리소스만 볼 수 있음

다른 사용자로 로그인하고 다양한 작업을 시도하여 권한 정책이 예상대로 작동하는지 확인할 수 있습니다. Express 애플리케이션의 터미널에서 권한 부여 결정에 대한 추가 세부 정보를 제공하는 로그 출력을 볼 수 있습니다.

문제 해결

권한 부여에 실패한 경우 다음을 시도하세요.

- 정책 스토어 ID가 올바른지 확인
- 자격 증명 소스가 올바르게 구성되었는지 확인
- 정책 형식이 올바른지 확인
- JWT 토큰이 유효한지 확인

다음 단계

기본 통합을 구현한 후 다음 사항을 고려하세요.

- 특정 권한 부여 시나리오에 대한 사용자 지정 매퍼 구현
- 권한 부여 결정을 위한 모니터링 및 로깅 설정
- 다양한 사용자 역할에 대한 추가 정책 생성

Amazon Verified Permissions에서 권한 부여 구현

정책 스토어, 정책, 템플릿, 스키마 및 권한 부여 모델을 빌드한 후에는 Amazon Verified Permissions를 사용하여 요청 권한 부여를 시작할 준비가 된 것입니다. Verified Permissions 권한 부여를 구현하려면 의 권한 부여 정책 구성을 애플리케이션의 통합 AWS 과 결합해야 합니다. Verified Permissions를 애플리케이션과 통합하려면 AWS SDK를 추가하고 Verified Permissions API를 호출하고 정책 스토어에 대한 권한 부여 결정을 생성하는 메서드를 구현합니다.

Verified Permissions를 사용한 권한 부여는 애플리케이션의 UX 권한 및 API 권한에 유용합니다.

UX 권한

> 애플리케이션 UX에 대한 사용자 액세스를 제어합니다. 사용자가 액세스해야 하는 정확한 양식, 버튼, 그래픽 및 기타 리소스만 볼 수 있도록 허용할 수 있습니다. 예를 들어 사용자가 로그인할 때 계정에 "자금 이체" 버튼이 표시되는지 확인할 수 있습니다. 사용자가 수행할 수 있는 작업을 제어할 수도 있습니다. 예를 들어 동일한 뱅킹 앱에서 사용자가 트랜잭션 범주를 변경할 수 있는지 여부를 확인할 수 있습니다.

API 권한

> 데이터에 대한 사용자 액세스를 제어합니다. 애플리케이션은 종종 분산 시스템의 일부이며 외부 APIs에서 정보를 가져옵니다. Verified Permissions가 "자금 이체" 버튼 표시를 허용한 뱅킹 앱의 예제에서는 사용자가 이체를 시작할 때 보다 복잡한 권한 부여 결정을 내려야 합니다. Verified Permissions는 적격 이전 대상인 대상 계정을 나열하는 API 요청을 승인한 다음 이전을 다른 계정으로 푸시하는 요청을 승인할 수 있습니다.

이 콘텐츠를 설명하는 예제는 샘플 정책 스토어에서 가져온 것입니다. 다음을 수행하려면 테스트 환경에 DigitalPetStore 샘플 정책 스토어를 생성합니다.

배치 권한 부여를 사용하여 UX 권한을 구현하는 엔드 투 엔드 샘플 애플리케이션은 AWS 보안 블로그의 대규모 세분화된 권한 부여를 위한 Amazon Verified Permissions 사용을 참조하세요.

주제

- 권한 부여에 사용 가능한 API 작업
- 권한 부여 모델 테스트
- 권한 부여 모델을 애플리케이션과 통합

권한 부여에 사용 가능한 API 작업

Verified Permissions API에는 다음과 같은 권한 부여 작업이 있습니다.

IsAuthorized

`IsAuthorized` API 작업은 Verified Permissions가 있는 권한 부여 요청의 진입점입니다. 보안 주체, 작업, 리소스, 컨텍스트 및 엔터티 요소를 제출해야 합니다. Verified Permissions는 요청의 엔터티에 적용되는 요청된 정책 스토어의 모든 정책에 대해 요청을 평가합니다.

IsAuthorizedWithToken

`IsAuthorizedWithToken` 작업은 JSON 웹 토큰(JWTs)의 사용자 데이터에서 권한 부여 요청을 생성합니다. Verified Permissions는 정책 스토어의 자격 증명 소스 Amazon Cognito 와 같은 OIDC 공급자와 직접 작동합니다. Verified Permissions는 사용자의 ID 또는 액세스 토큰의 클레임에서 요청의 보안 주체에 모든 속성을 채웁니다. 자격 증명 소스의 사용자 속성 또는 그룹 멤버십에서 작업 및 리소스를 승인할 수 있습니다.

`IsAuthorizedWithToken` 요청에 그룹 또는 사용자 보안 주체 유형에 대한 정보를 포함할 수 없습니다. 제공하는 JWT에 모든 보안 주체 데이터를 채워야 합니다.

BatchIsAuthorized

`BatchIsAuthorized` 작업은 단일 API 요청에서 단일 보안 주체 또는 리소스에 대한 여러 권한 부여 결정을 처리합니다. 이 작업은 할당량 사용을 최소화하고 최대 30개의 복잡한 중첩 작업 각각에 대한 권한 부여 결정을 반환하는 단일 배치 작업으로 요청을 그룹화합니다. 단일 리소스에 대한 배치 권한 부여를 사용하면 사용자가 리소스에 대해 수행할 수 있는 작업을 필터링할 수 있습니다. 단일 보안 주체에 대한 배치 권한 부여를 사용하면 사용자가 조치를 취할 수 있는 리소스를 필터링할 수 있습니다.

BatchIsAuthorizedWithToken

`BatchIsAuthorizedWithToken` 작업은 하나의 API 요청으로 단일 보안 주체에 대한 여러 권한 부여 결정을 처리합니다. 보안 주체는 정책 스토어 ID 소스에서 ID 또는 액세스 토큰으로 제공됩니다. 이 작업은 할당량 사용을 최소화하고 작업 및 리소스에 대한 최대 30개의 요청 각각에 대한 권한 부여 결정을 반환하는 단일 배치 작업으로 요청을 그룹화합니다. 정책에서 사용자 디렉터리의 속성 또는 그룹 멤버십에서 액세스 권한을 부여할 수 있습니다.

와 마찬가지로 `BatchIsAuthorizedWithToken` 요청에 그룹 또는 사용자 보안 주체 유형에 대한 정보를 포함할 `IsAuthorizedWithToken`수 없습니다. 제공하는 JWT에 모든 보안 주체 데이터를 채워야 합니다.

권한 부여 모델 테스트

애플리케이션을 배포할 때 Amazon Verified Permissions 권한 부여 결정의 영향을 이해하려면 Amazon Verified Permissions 테스트 벤치 사용 및 HTTPS REST API 요청을 사용하여 정책을 개발할 때 정책을 평가할 수 있습니다. 테스트 벤치는 정책 스토어의 권한 부여 요청 및 응답을 AWS Management Console 평가하는의 도구입니다.

Verified Permissions REST API는 개념적 이해에서 애플리케이션 설계로 전환함에 따라 개발의 다음 단계입니다. Verified Permissions API는 IsAuthorized, IsAuthorizedWithToken 및 BatchIsAuthorized를 사용하여 리전 서비스 엔드포인트에 대한 서명된 AWS API 요청으로 권한 부여 요청을 수락합니다. 권한 부여 모델을 테스트하기 위해 API 클라이언트로 요청을 생성하고 정책이 예상대로 권한 부여 결정을 반환하는지 확인할 수 있습니다.

예를 들어 다음 절차를 사용하여 샘플 정책 스토어`IsAuthorized`에서 테스트할 수 있습니다.

Test bench

1. Verified Permissions 콘솔에서 Verified Permissions 콘솔을 엽니다. 샘플 정책 스토어에서 DigitalPetStore라는 이름으로 정책 스토어를 생성합니다.
2. 새 정책 스토어에서 테스트 벤치를 선택합니다.
3. Verified Permissions API 참조의 IsAuthorized에서 테스트 벤치 요청을 채웁니다. 다음 세부 정보는 DigitalPetStore 샘플을 참조하는 예제 4의 조건을 복제합니다.

    a. Alice를 보안 주체로 설정합니다. 작업을 수행하는 보안 주체에서 `DigitalPetStore::User`를 선택하고를 입력합니다`Alice`.

    b. Alice의 역할을 고객으로 설정합니다. 상위 항목 추가를 선택하고 `DigitalPetStore::Role`를 선택한 다음 고객을 입력합니다.

    c. 리소스를 "1234" 순서로 설정합니다. 보안 주체가 작업하는 리소스에서 `DigitalPetStore::Order`를 선택하고를 입력합니다`1234`.

    d. `DigitalPetStore::Order` 리소스에는 `owner` 속성이 필요합니다. Alice를 주문 소유자로 설정합니다. 선택 `DigitalPetStore::User` 및 입력 `Alice`

    e. Alice가 주문을 보도록 요청했습니다. 보안 주체가 수행하는 작업에서를 선택합니다`DigitalPetStore::Action::"GetOrder"`.
4. 권한 부여 요청 실행을 선택합니다. 수정되지 않은 정책 스토어에서이 요청은 `ALLOW` 결정을 내리게 됩니다. 결정을 반환한 충족 정책을 기록해 둡니다.
5. 왼쪽 탐색 메뉴에서 정책을 선택합니다. 고객 역할 - 주문 받기 설명과 함께 정적 정책을 검토합니다.

6. 보안 주체가 고객 역할에 있고 리소스의 소유자였기 때문에 Verified Permissions가 요청을 허용했는지 관찰합니다.

REST API

1. Verified Permissions 콘솔에서 Verified Permissions 콘솔을 엽니다. 샘플 정책 스토어에서 DigitalPetStore라는 이름으로 정책 스토어를 생성합니다.
2. 새 정책 스토어의 정책 스토어 ID를 기록해 둡니다.
3. Verified Permissions API 참조의 IsAuthorized에서 DigitalPetStore 샘플을 참조하는 예제 4의 요청 본문을 복사합니다.
4. API 클라이언트를 열고 정책 스토어의 리전 서비스 엔드포인트에 대한 요청을 생성합니다. 예제와 같이 헤더를 채웁니다.
5. 샘플 요청 본문에 붙여넣고의 값을 앞서 기록한 정책 스토어 IDpolicyStoreId로 변경합니다.
6. 요청을 제출하고 결과를 검토합니다. 기본 DigitalPetStore 정책 스토어에서이 요청은 ALLOW 결정을 반환합니다.

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

1. Verified Permissions에서 결정을 변경하는 방식으로 요청을 변경합니다. 예를 들어 Alice의 역할을 로 변경하거나 순서 1234의 owner 속성을 로 Employee 변경합니다Bob.
2. 권한 부여 결정에 영향을 미치는 방식으로 정책을 변경합니다. 예를 들어 고객 역할 - 주문 가져오기라는 설명으로 정책을 수정하여가의 소유자User여야 하는 조건을 제거하고가 주문을 보Bob도록 요청을 Resource 수정합니다.
3. 스키마를 변경하여 정책이 더 복잡한 결정을 내릴 수 있도록 합니다. Alice가 새 요구 사항을 충족할 수 있도록 요청 엔터티를 업데이트합니다. 예를 들어가 ActiveUsers 또는의 멤버가 되도록 스키마를 편집User합니다InactiveUsers. 활성 사용자만 자신의 주문을 볼 수 있도록 정책을 업데이트합니다. Alice가 활성 또는 비활성 사용자가 되도록 요청 엔터티를 업데이트합니다.

권한 부여 모델을 애플리케이션과 통합

애플리케이션에서 Amazon Verified Permissions를 구현하려면 앱이 적용할 정책과 스키마를 정의해야 합니다. 권한 부여 모델을 마련하고 테스트한 상태에서 다음 단계는 적용 지점에서 API 요청 생성을 시

작하는 것입니다. 이렇게 하려면 사용자 데이터를 수집하여 권한 부여 요청에 채우도록 애플리케이션 로직을 설정해야 합니다.

앱이 Verified Permissions를 사용하여 요청을 승인하는 방법

1. 현재 사용자에 대한 정보를 수집합니다. 일반적으로 사용자의 세부 정보는 JWT 또는 웹 세션 쿠키와 같은 인증된 세션의 세부 정보에 제공됩니다. 이 사용자 데이터는 정책 스토어에 연결된 Amazon Cognito ID 소스 또는 다른 OpenID Connect(OIDC) 공급자에서 비롯될 수 있습니다.
2. 사용자가 액세스하려는 리소스에 대한 정보를 수집합니다. 일반적으로 사용자가 앱이 새 자산을 로드해야 하는 항목을 선택하면 애플리케이션에 리소스에 대한 정보가 수신됩니다.
3. 사용자가 수행하려는 작업을 결정합니다.
4. 사용자가 시도한 작업에 대해 보안 주체, 작업, 리소스 및 엔터티를 사용하여 Verified Permissions에 대한 권한 부여 요청을 생성합니다. Verified Permissions는 정책 스토어의 정책을 기준으로 요청을 평가하고 권한 부여 결정을 반환합니다.
5. 애플리케이션은 Verified Permissions의 허용 또는 거부 응답을 읽고 사용자의 요청에 대한 결정을 적용합니다.

Verified Permissions API 작업은 AWS SDKs. 앱에 Verified Permissions를 포함하려면 선택한 언어의 AWS SDK를 앱 패키지에 통합합니다.

자세히 알아보고 AWS SDKs. Amazon Web Services

다음은 AWS SDKs.

- 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
- AWS SDK for Rust

에 대한 다음 AWS SDK for JavaScript 예제는 Amazon Verified Permissions 및 Amazon Cognito를 사용한 세분화된 권한 부여 간소화에서 `IsAuthorized` 비롯됩니다.

```
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 Verified Permissions 워크숍
- Amazon Verified Permissions - 리소스
- Amazon Verified Permissions를 사용하여 ASP.NET Core 앱에 대한 사용자 지정 권한 부여 정책 공급자 구현
- Amazon Verified Permissions를 사용하여 비즈니스 애플리케이션을 위한 권한 부여 서비스 구축
- Amazon Verified Permissions 및 Amazon Cognito를 사용하여 세분화된 권한 부여 간소화

API 요청 만들기

Amazon Verified Permissions에 대한 쿼리 요청은 POST 메서드를 사용하는 HTTP 또는 HTTPS 요청입니다.

Verified Permissions 엔드포인트

엔드포인트는 웹 서비스의 진입점 역할을 하는 URL입니다. 지연 시간을 줄이기 위해 요청할 때 적절한 AWS 리전 엔드포인트를 선택할 수 있습니다. Verified Permissions에서 사용하는 엔드포인트에 대한 자세한 내용은의 Amazon Verified Permissions를 참조하세요Amazon Web Services 일반 참조.

요청 파라미터

Amazon Verified Permissions는 JSON 기반 프로토콜을 사용합니다. HTTP 요청 본문에서 요청 파라미터를 JSON으로 전달합니다. 목록은 JSON 배열로 표시됩니다. 자세한 내용은 Amazon Verified Permissions API 참조의 공통 파라미터를 참조하세요.

요청 식별자

모든 응답에서는 `x-amzn-RequestId` HTTP 응답 헤더를 통해 요청 ID를 AWS 반환합니다. 이 문자열은가 추적 정보를 제공하도록 AWS 할당하는 고유 식별자입니다. 요청 ID는 모든 응답에 포함되지만 가독성을 높이고 중복성을 줄이기 위해 개별 API 설명서 페이지에 나열되지는 않습니다.

쿼리 API 인증

HTTPS를 통해 쿼리 요청을 보냅니다. 모든 쿼리 요청에 서명을 포함해야 합니다. 서명 생성 및 포함에 대한 자세한 내용은의 AWS API 요청 서명을 참조하세요Amazon Web Services 일반 참조.

사용 가능한 라이브러리

AWS 는 명령줄 도구 및 쿼리 API 대신 언어별 APIs를 사용하여 애플리케이션을 빌드하려는 소프트웨어 개발자를 위해 라이브러리, 샘플 코드, 자습서 및 기타 리소스를 제공합니다. 이러한 라이브러리는 더 쉽게 시작할 수 있도록 요청 인증, 요청 재시도 및 오류 처리와 같은 기본 함수(APIs에 포함되지 않음)를 제공합니다. Verified Permissions 라이브러리 및 리소스는 다음 언어 및 플랫폼에서 사용할 수 있습니다.

- AWS SDK for Go

- AWS SDK for Java 2.x
- AWS SDK for Java 1.x
- AWS SDK for JavaScript
- AWS SDK for .NET
- AWS SDK for PHP
- AWS SDK for Python (Boto)
- AWS SDK for Ruby

모든 언어의 라이브러리 및 샘플 코드에 대한 자세한 내용은 샘플 코드 및 라이브러리를 참조하세요.

POST 메서드를 사용하여 API 요청

AWS SDKs 중 하나를 사용하지 않는 경우 요청 방법을 사용하여 HTTPS를 통해 Verified Permissions POST 요청을 수행할 수 있습니다. POST 메서드를 사용하려면 요청의 헤더에 작업을 지정하고 요청 본문에 JSON 형식으로 작업에 대한 데이터를 제공해야 합니다.

헤더 이름	헤더 값
`Host`	Amazon Verified Permissions 엔드포인트입니다. 예: `verifiedpermissions.us-east-1.amazonaws.com`
`X-Amz-Date`	HTTP 날짜 헤더 또는 the AWS x-amz-date 헤더에 타임스탬프를 제공해야 합니다. 일부 HTTP 클라이언트 라이브러리에서는 날짜 헤더를 설정할 수 없습니다. x-amz-date 헤더가 있으면 시스템은 요청 인증 중에 날짜 헤더를 무시합니다. x-amz-date 헤더는 ISO 8601 기본 형식으로 지정해야 합니다. 예: `20130315T092054Z`
`Authorization`	요청의 유효성 및 신뢰성을 보장하기 위해가 AWS 사용하는 권한 부여 파라미터 세트입니다. 이 헤더 구성에 대한 자세한 내용은의 서명 버전 4 서명 프로세스를 참조하세요Amazon Web Services 일반 참조.
`X-Amz-Target`	수행하려는 Verified Permissions 작업을 지정합니다. `VerifiedPermissions.` *`API_Name`*

헤더 이름	헤더 값
	예를 들어 CreatePolicy 작업을 호출하려면 다음 대상 값을 사용합니다. VerifiedPermissions.CreatePolicy
Content-Type	입력 형식을 지정합니다. 다음 값을 사용합니다. application/x-amz-json-1.0
Accept	응답 형식을 지정합니다. 다음 값을 사용합니다. application/x-amz-json-1.0
Content-Length	바이트 단위의 페이로드 크기입니다.
Content-Encoding	입력 및 출력의 인코딩 형식을 지정합니다. 다음 값을 사용합니다. amz-1.0

다음은 HTTP 요청이가 명User명된를 Principal 참조하는의 지정된 정책 스토어에 AWS 계정 있는 모든 정책 목록을 반환하는 헤더의 예입니다alice. 이 예제에서는 읽기 쉽도록 Authorization 줄을 단어로 묶습니다. 실제 요청에서 단어를 묶지 마세요.

```
POST / HTTP/1.1
Host: verifiedpermissions.us-east-1.amazonaws.com
X-Amz-Date: 20230101T200059Z
Accept-Encoding: identity
Content-Type: application/x-amz-json-1.0
X-Amz-Target: VerifiedPermissions.ListPolicies
User-Agent: <UserAgentString>
Authorization: AWS4-HMAC-SHA256 Credential=<Credential>, SignedHeaders=<Headers>,
 Signature=<Signature>
Content-Length: <PayloadSizeBytes>

{
    "PolicyStoreId": "PSExAmPLE222222222",
    "Filter": {
        "Principal": {
            "Id": {
```

```
                "EntityType": "User",
                "EntityId": "alice"
            }
        }
    }
}
```

Amazon Verified Permissions의 보안

의 클라우드 보안 AWS 이 최우선 순위입니다. AWS 고객은 보안에 가장 민감한 조직의 요구 사항을 충족하도록 구축된 데이터 센터 및 네트워크 아키텍처의 이점을 누릴 수 있습니다.

보안은 AWS 와 사용자 간의 공동 책임입니다. 공동 책임 모델은 이 사항을 클라우드의 보안 및 클라우드 내 보안으로 설명합니다.

- 클라우드 보안 -에서 AWS 서비스를 실행하는 인프라를 보호할 AWS 책임이 있습니다 AWS 클라우드. AWS 또한는 안전하게 사용할 수 있는 서비스를 제공합니다. 타사 감사자는 규정 AWS 준수 프로그램 일환으로 보안의 효과를 정기적으로 테스트하고 확인합니다. Amazon Verified Permissions에 적용되는 규정 준수 프로그램에 대한 자세한 내용은 규정 준수 프로그램 AWS 제공 범위 내 서비스 규정 준수 프로그램 .
- 클라우드의 보안 - 사용자의 책임은 사용하는 AWS 서비스에 따라 결정됩니다. 또한 귀하는 귀사의 데이터 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 도움이 됩니다. 다음 주제에서는 보안 및 규정 준수 목표를 충족하도록 Verified Permissions를 구성하는 방법을 보여줍니다. 또한 Verified Permissions 리소스를 모니터링하고 보호하는 데 도움이 되는 다른 AWS 서비스를 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 공동 책임 모델 Amazon Verified Permissions의 데이터 보호에 적용됩니다. 이 모델에서 설명하는 것처럼 AWS 는 모든 AWS 클라우드를 실행하는 글로벌 인프라를 보호할 책임이 있습니다. 사용자는 이 인프라에 호스팅되는 콘텐츠에 대한 통제 권한을 유지할 책임이 있습니다. 이 콘텐츠에는 AWS 서비스 사용하는에 대한 보안 구성 및 관리 작업이 포함됩니다. 데이터 프라이버시에 관한 자세한 내용

은 데이터 프라이버시 FAQ를 참조하세요. 유럽의 데이터 보호에 대한 자세한 내용은 AWS 보안 블로그의 AWS 공동 책임 모델 및 GDPR 블로그 게시물을 참조하세요.

- 데이터 보호를 위해 자격 증명을 보호하고 AWS 계정 AWS IAM Identity Center 또는 AWS Identity and Access Management ()를 사용하여 개별 사용자를 설정하는 것이 좋습니다IAM. 이렇게 하면 개별 사용자에게 자신의 직무를 충실히 이행하는 데 필요한 권한만 부여됩니다.
- 다음과 같은 방법으로 데이터를 보호하는 것이 좋습니다.
  - 각 계정에 다중 인증(MFA)을 사용합니다.
  - SSL/TLS를 사용하여 AWS 리소스와 통신합니다. TLS 1.2가 필요합니다.
  - 를 사용하여 API 및 사용자 활동 로깅을 설정합니다 AWS CloudTrail.
  - 내부의 모든 기본 보안 제어와 함께 AWS 암호화 솔루션을 사용합니다 AWS 서비스.
  - 와 같은 고급 관리형 보안 서비스를 사용하면 Amazon Macie에 저장된 민감한 데이터를 검색하고 보호할 수 있습니다 Amazon S3.
  - 명령행 인터페이스 또는 API를 통해 AWS 에 액세스할 때 FIPS 140-2 검증된 암호화 모듈이 필요한 경우, FIPS 엔드포인트를 사용합니다. 사용 가능한 FIPS 엔드포인트에 대한 자세한 내용은 Federal Information Processing Standard(FIPS) 140-2를 참조하세요.
- 고객의 이메일 주소와 같은 기밀 정보나 중요한 정보는 태그나 이름 필드와 같은 자유 양식 필드에 입력하지 않는 것이 좋습니다. 여기에는 Verified Permissions 또는 기타에서 콘솔 AWS CLI, API 또는 AWS SDKs를 AWS 서비스 사용하여 작업하는 경우가 포함됩니다. 이름에 사용되는 태그 또는 자유 형식 텍스트 필드에 입력하는 모든 데이터는 청구 또는 진단 로그에 사용될 수 있습니다. 외부 서버에 URL을 제공할 때 해당 서버에 대한 요청을 검증하기 위해 자격 증명을 URL에 포함해서는 안 됩니다.
- 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. 그러나 프로덕션 시스템에서는 보안상의 이유로 재사용할 수 없는 고유한 값을 사용하는 것이 중요합니다. 범용 고유 식별자(UUID) 와 같은 값을 사용하는 것이 좋습니다. 회사를 떠나는 사용자 `jane`을 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.

- `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 정책이 예기치 않게 액세스를 허용하지 못하거나 금지 정책이 예기치 않게 액세스를 차단하지 못할 수 있습니다.

데이터 암호화

Amazon Verified Permissions는를 사용하여 정책과 같은 모든 고객 데이터를 자동으로 암호화합니다 AWS 관리형 키. 또한 Amazon Verified Permissions를 사용하면 고객이를 활용하여 데이터를 암호화 고객 관리형 키 할 수 있습니다.

암호화에 고객 관리형 키를 사용하는 방법에 대한 자세한 내용은 섹션을 참조하세요the section called “고객 관리형 키”.

Amazon Verified Permissions에서 리소스 암호화

Amazon Verified Permissions는 기본적으로 암호화를 제공하여 AWS 소유 암호화 키를 사용하여 저장된 민감한 고객 데이터를 보호합니다. 추가 보호 계층인 Amazon Verified Permissions를 사용하면 AWS Key Management Service (AWS KMS) 고객 관리형 키 s(CMK)를 사용하여 정책 스토어를 암호화할 수 있습니다. 이 기능은 저장 데이터 암호화를 통해 민감한 데이터를 보호하므로 다음을 수행할 수 있습니다.

- 애플리케이션 측의 운영 부담을 줄여 민감한 데이터 보호
- 자체 AWS KMS 고객 관리형 키를 통해 권한 부여 정책의 세부 정보를 볼 수 있는 사용자를 제어합니다.
- 엄격한 암호화 규정 준수 및 규제 요구 사항을 충족하는 보안에 민감한 애플리케이션 구축

다음 섹션에서는 새 정책 스토어에 대한 암호화를 구성하고 암호화 키를 관리하는 방법을 설명합니다.

AWS KMS Amazon Verified Permissions의 키 유형

Amazon Verified Permissions는와 통합되어 고객 데이터를 암호화/복호화하는 AWS KMS 데 사용되는 암호화 키를 관리합니다. 키 유형 및 상태에 대한 자세한 내용은 AWS KMS 개발자 안내서의 AWS Key Management Service concepts를 참조하세요. 새 정책 스토어를 생성할 때 다음 AWS KMS 키 유형 중에서 선택하여 데이터를 암호화할 수 있습니다.

AWS 소유 키

기본 암호화 유형입니다. Amazon Verified Permissions는 추가 비용 없이 키를 소유하고 생성 시 저장된 리소스 데이터를 암호화합니다. Verified Permissions에서 소유한 키를 사용하여 데이터를 암호화/복호화하기 위해 코드 또는 애플리케이션에 추가 구성이 필요하지 않습니다.

고객 관리형 키

AWS 계정에서 키를 생성, 소유 및 관리합니다. 에 대한 AWS KMS key. AWS KMS charges가 적용됩니다를 완전히 제어할 고객 관리형 키수 있습니다. 자세한 내용은 AWS KMS 요금 페이지를 참조하세요. 키 유형에 대한 자세한 내용은 AWS KMS 개발자 안내서의 고객 관리형 키를 참조하세요.

최상위 리소스(예: 정책 스토어) 고객 관리형 키 에 대한 암호화를 위해를 지정하면 Verified Permissions는 해당 키로 리소스와 하위 리소스를 암호화합니다. 를 사용하여 정책 스토어를 암호화하려면 키 정책의 Verified Permissions에 대한 액세스 권한을 부여 고객 관리형 키해야 합니다. 키 정책은 고객 관리형 키 에 연결하여 액세스 권한을 제어하는 리소스 기반 정책입니다. 자세한 내용은 the section called “Amazon Verified Permissions에 대한 AWS KMS 키 사용 권한 부여” 섹션을 참조하세요.

또한를 사용하여 암호화된 정책 저장소를 생성하거나 고객 관리형 키로 암호화된 정책 저장소에 대한 API 호출을 수행하려면 호출 고객 관리형 키을 수행하는 IAM 사용자 또는 역할도 키에 액세스할 수 있어야 합니다. Verified Permissions가 키에 액세스할 수 없는 경우 해당 키로 암호화된 리소스와 관련된 권한 부여 결정이 오래되었거나 부정확할 수 있습니다. 키에 액세스할 수 없는 경우 해당 키로 암호화된 리소스를 read/update/delete 수 없으며 암호화에 키를 활용하기 위한 호출 생성이 실패합니다.

> **Note**
>
> Verified Permissions 저장 시 암호화는 Verified Permissions를 사용할 수 있는 모든 AWS 리전에서 사용할 수 있습니다.

> **Important**
>
> 고객 관리형 키 를 사용하여 정책 스토어를 암호화한 후에는 암호화에 다른 키를 사용하도록 리소스를 업데이트하거나 해당 정책 스토어에서 키를 제거할 수 없습니다.

Amazon Verified Permissions에서 AWS KMS 및 데이터 키 사용

Amazon Verified Permissions 저장 시 암호화 기능은 AWS KMS 키와 데이터 키 계층 구조를 사용하여 리소스 데이터를 보호합니다.

> **Note**
>
> Amazon Verified Permissions는 대칭 AWS KMS 키만 지원합니다. 비대칭 AWS KMS 키를 사용하여 Amazon Verified Permissions 리소스를 암호화할 수 없습니다.

AWS 소유 키 사용

Amazon Verified Permissions는 기본적으로 AWS 소유 키를 사용하여 모든 리소스를 암호화합니다. 이러한 키는 무료로 사용 가능하며 계정 리소스를 보호하기 위해 매년 교체할 수 있습니다. 이러한 키를 보거나 관리하거나 사용하거나 감사할 필요가 없으므로 데이터 보호를 위해 별도로 작업할 필요가 없습니다. AWS 소유 키에 대한 자세한 내용은 AWS KMS 개발자 안내서의 AWS 소유 키를 참조하세요.

고객 관리형 키 사용

암호화를 고객 관리형 키 위해를 선택하면 다음과 같은 이점이 있습니다.

- AWS KMS 키에 대한 액세스를 제어하기 위한 키 정책 및 IAM 정책 설정을 포함하여 AWS KMS 키를 생성하고 관리합니다. AWS KMS 키를 활성화 및 비활성화하고, 자동 키 교체를 활성화 및 비활성화하고, 더 이상 사용하지 않을 때 AWS KMS 키를 삭제할 수 있습니다.
- 가져온 키 구성 요소와 고객 관리형 키 함께 또는 소유하고 관리하는 사용자 지정 키 스토어 고객 관리형 키 에서를 사용할 수 있습니다.
- AWS KMS AWS CloudTrail 로그에서에 대한 Amazon Verified Permissions API 호출을 검사하여 Verified Permissions 리소스의 암호화 및 복호화를 감사할 수 있습니다.

Amazon Verified Permissions가 암호화/복호화에 고객 관리형 키를 사용하려면 Amazon Verified Permissions가 사용자를 대신하여 리소스를 암호화/복호화할 수 있도록 특정 키 정책을 추가해야 합니다.

Amazon Verified Permissions에 대한 AWS KMS 키 사용 권한 부여

최소한 Amazon Verified Permissions에는에 대한 다음 권한이 필요합니다. 고객 관리형 키

- kms:Encrypt
- kms:GenerateDataKeyWithoutPlaintext
- kms:DescribeKey
- kms:ReEncryptTo
- kms:ReEncryptFrom
- kms:Decrypt

다음은 예제 키 정책입니다.

```
{
    "Sid": "Enable AVP to use the KMS key for encrypting project J.A.K. policy
 resources",
    "Effect": "Allow",
    "Principal": {
        "Service": "verifiedpermissions.amazonaws.com"
    },
    "Action": [
        "kms:Decrypt",
        "kms:GenerateDataKeyWithoutPlaintext",
        "kms:Encrypt",
        "kms:ReEncryptFrom",
        "kms:ReEncryptTo",
        "kms:DescribeKey"
    ],
    "Resource": "*"
}
```

소스 컨텍스트 이해

소스 컨텍스트는 지정된 키에 대해 AWS KMS 작업을 수행하려고 시도하는 소스 호출자에 대한 정보를 제공합니다. 이렇게 하면 데이터 소스에 컨텍스트를 바인딩하여 암호화된 데이터의 혼동 또는 오용을 방지할 수 있습니다.

고객은 소스 컨텍스트를 다음 키 정책 설명과 같은 키 정책의 추가 조건으로 활용할 수 있습니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Enable this account full access to this key",
```

```
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::111122223333:root"
            },
            "Action": "kms:*",
            "Resource": "*"
        },
        {
            "Sid": "Enable AVP to retrieve this key's metadata",
            "Effect": "Allow",
            "Principal": {
                "Service": "verifiedpermissions.amazonaws.com"
            },
            "Action": "kms:DescribeKey",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "aws:SourceAccount": "111122223333"
                },
                "ArnLike": {
                    "aws:SourceArn": "arn:aws:verifiedpermissions::111122223333:policy-
store/*"
                }
            }
        },
        {
            "Sid": "Enable AVP to encrypt/decrypt resources utilizing this key",
            "Effect": "Allow",
            "Principal": {
                "Service": "verifiedpermissions.amazonaws.com"
            },
            "Action": [
                "kms:Decrypt",
                "kms:ReEncryptTo",
                "kms:ReEncryptFrom",
                "kms:GenerateDataKeyWithoutPlaintext",
                "kms:Encrypt"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "aws:SourceAccount": "111122223333"
                },
                "ArnLike": {
```

```
                    "aws:SourceArn": "arn:aws:verifiedpermissions::111122223333:policy-
store/*"
                }
            }
        }
    ]
}
```

이 키 정책은 소스 계정이이 AWS KMS 키가 있는 계정과 동일한 경우 Verified Permissions가 사용자를 대신하여 AWS KMS 호출할 수 있도록 허용합니다. CMK 키에 대한 AWS CloudTrail 감사 로그를 확인할 때 이러한 값을 확인할 수 있어야 합니다. 전역 AWS 조건 키에 대한 자세한 내용은 aws:SourceArn 또는 aws:SourceAccount 조건 키 사용을 참조하세요.

암호화 컨텍스트 이해

암호화 컨텍스트는 암호화 무결성 검사를 위한 추가 인증 데이터를 포함하는 키-값 페어 세트입니다. 데이터 암호화 요청에 암호화 컨텍스트를 포함하면 암호화 컨텍스트가 암호화된 데이터에 AWS KMS 암호화 방식으로 바인딩됩니다. 데이터를 해독하려면 동일한 암호화 컨텍스트를 전달해야 합니다.

Amazon Verified Permissions는 모든 암호화 작업에서 동일한 AWS KMS 암호화 컨텍스트를 사용하며 Verified Permissions가 사용자를 대신하여 암호화/복호화 프로세스를 AWS KMS 호출할 때 로그 내에서 AWS CloudTrail 확인할 수 있습니다. 기본적으로 Verified Permissions는 리소스를 암호화할 때 다음과 같은 암호화 컨텍스트 키-값 페어를 활용합니다.

```
{
    "aws:verifiedpermissions:policy-store-arn":
 "arn:aws:verifiedpermissions::111122223333:policy-store/PSt123456789012"
}
```

또한 Amazon Verified Permissions를 사용하면 암호화/복호화 프로세스 중에 포함하려는 추가 메타데이터의 일부로 사용자 지정 암호화 컨텍스트를 추가할 수 있습니다. 즉, 키 정책이 아래 예제와 같은 권한을 부여하는 데 더 세분화될 수 있습니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Enable this account full access to this key",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::111122223333:root"
```

```
            },
            "Action": "kms:*",
            "Resource": "*"
        },
        {
            "Sid": "Enable AVP to retrieve this key's metadata",
            "Effect": "Allow",
            "Principal": {
                "Service": "verifiedpermissions.amazonaws.com"
            },
            "Action": "kms:DescribeKey",
            "Resource": "*"
        },
        {
            "Sid": "Enable AVP to encrypt/decrypt resources utilizing this key",
            "Effect": "Allow",
            "Principal": {
                "Service": "verifiedpermissions.amazonaws.com"
            },
            "Action": [
                "kms:Decrypt",
                "kms:ReEncryptTo",
                "kms:ReEncryptFrom",
                "kms:GenerateDataKeyWithoutPlaintext",
                "kms:Encrypt"
            ],
            "Resource": "*",
            "Condition": {
                "StringLike": {
                    "kms:EncryptionContext:aws:verifiedpermissions:policy-store-arn":
 "arn:aws:verifiedpermissions::111122223333:policy-store/*",
                    "kms:EncryptionContext:policy_owner": "Tim"
                }
            }
        }
    ]
}
```

이 키 정책은 암호화 컨텍스트 맵에 키가 포함되어 있고 값이 형식을 따르arn:aws:verifiedpermissions::111122223333:policy-store/*며 키-값 페어 도 포함되어 aws:verifiedpermissions:policy-store-arn있는 경우 Verified Permissions가 사용자를 대신하여 AWS KMS 호출할 수 있도록 허용합니다"policy_owner": "Tim". 사용자 지정 암호화 컨텍스트를 설정하는 방법은 the section called “암호화된 정책 저장소 생성” 섹션을 참조하세요.

> **Note**
>
> 각 키-값 페어를 확인하는 대신 암호화 컨텍스트를 기반으로 하는 조건이 있는 키 정책이 암호화 컨텍스트 맵의 하위 집합에 대한 것이 좋습니다. 서비스 및 해당 종속성 업스트림은 사용자에게 표시되지 않는 키-값 페어를 추가할 수 있으며 키 정책이 암호화 컨텍스트 맵의 정확한 모양을 기반으로 조건부로 허용하는 경우 Verified Permissions의 키 액세스에 영향을 미칠 수 있습니다.

kms:ViaService 이해

`kms:ViaService` 조건 키는 AWS KMS 키 사용을 지정된 AWS 서비스의 요청으로 제한합니다. 이 조건 키는 전달 액세스 세션(FAS)에만 적용됩니다. 에 대한 자세한 내용은 AWS KMS 개발자 안내서의 kms:ViaService를 `kms:ViaService`참조하세요.

예를 들어 다음 키 정책 문은 `kms:ViaService` 조건 키를 사용하여 요청이를 대신하여 미국 동부(버지니아 북부) 리전의 Amazon Verified Permissions에서 오는 경우에만 지정된 작업에를 사용할 고객 관리형 키 수 있도록 허용합니다`BrentRole`.

```
{
    "Sid": "Enable AVP to encrypt/decrypt resources using credentials of BrentRole",
    "Effect": "Allow",
    "Principal": {
        "AWS": "arn:aws:iam::111122223333:role/BrentRole"
    },
    "Action": [
        "kms:Decrypt",
        "kms:GenerateDataKeyWithoutPlaintext",
        "kms:Encrypt",
        "kms:ReEncryptFrom",
        "kms:ReEncryptTo",
        "kms:DescribeKey"
    ],
    "Resource": "*",
    "Condition": {
        "StringEquals": {
            "kms:ViaService": [
                "verifiedpermissions.us-east-1.amazonaws.com"
            ]
        }
    }
```

```
}
```

이는 Verified Permissions가 암호화/암호 해독을 위해 AWS KMS 사용자를 대신하여에 요청할 때 Verified Permissions가 자격 증명, 권한 및 세션 속성을 전달할 수 있도록 하는 데 필요합니다. FAS 요청에 대한 자세한 내용은 IAM 사용 설명서의 전달 액세스 세션을 참조하세요.

전체 AWS KMS 키 정책

이전 섹션의 개념을 기반으로 Amazon Verified Permissions가 암호화/복호화에 CMK를 사용하도록 허용하는 예제 키 정책입니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Enable this account full access to this key",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::111122223333:root"
            },
            "Action": "kms:*",
            "Resource": "*"
        },
        {
            "Sid": "Enable AVP to retrieve this key's metadata",
            "Effect": "Allow",
            "Principal": {
                "Service": "verifiedpermissions.amazonaws.com"
            },
            "Action": "kms:DescribeKey",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "aws:SourceAccount": "111122223333"
                },
                "ArnLike": {
                    "aws:SourceArn": "arn:aws:verifiedpermissions::111122223333:policy-
store/*"
                }
            }
        },
        {
            "Sid": "Enable AVP to encrypt/decrypt resources utilizing this key",
```

```
            "Effect": "Allow",
            "Principal": {
                "Service": "verifiedpermissions.amazonaws.com"
            },
            "Action": [
                "kms:Decrypt",
                "kms:ReEncryptTo",
                "kms:ReEncryptFrom",
                "kms:Encrypt",
                "kms:GenerateDataKeyWithoutPlaintext"
            ],
            "Resource": "*",
            "Condition": {
                "StringLike": {
                    "kms:EncryptionContext:aws:verifiedpermissions:policy-store-arn":
 "arn:aws:verifiedpermissions::111122223333:policy-store/*",
                    "kms:EncryptionContext:policy_owner": "Tim"
                },
                "ArnLike": {
                    "aws:SourceArn": "arn:aws:verifiedpermissions::111122223333:policy-
store/*"
                },
                "StringEquals": {
                    "aws:SourceAccount": "111122223333"
                }
            }
        },
        {
            "Sid": "Enable AVP to encrypt/decrypt resources using credentials of
 BrentRole",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::111122223333:role/BrentRole"
            },
            "Action": [
                "kms:Decrypt",
                "kms:GenerateDataKeyWithoutPlaintext",
                "kms:Encrypt",
                "kms:ReEncryptFrom",
                "kms:ReEncryptTo"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
```

```
                    "kms:ViaService": [
                        "verifiedpermissions.us-east-1.amazonaws.com"
                    ]
                },
                "StringLike": {
                    "kms:EncryptionContext:aws:verifiedpermissions:policy-store-arn":
 "arn:aws:verifiedpermissions::111122223333:policy-store/*",
                    "kms:EncryptionContext:policy_owner": "Tim"
                }
            }
        },
        {
            "Sid": "Enable AVP to describe this key using credentials of BrentRole",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::111122223333:role/BrentRole"
            },
            "Action": "kms:DescribeKey",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "kms:ViaService": [
                        "verifiedpermissions.us-east-1.amazonaws.com"
                    ]
                }
            }
        }
    ]
}
```

Warning

Amazon Verified Permissions에서 이미 사용 중인 AWS KMS 키에 대한 키 정책을 수정할 때는 주의해야 합니다. 최상위 리소스 생성 중에 AWS KMS 키를 처음 구성할 때 Verified Permissions는 암호화 및 복호화 권한을 검증하지만 요청 시 후속 정책 변경을 확인할 수 없습니다. 실수로 필요한 권한을 제거하면 권한 부여 결정 및 정기적인 Verified Permissions 서비스 흐름이 중단될 수 있습니다. Amazon Verified Permissions의 고객 관리형 키와 관련된 일반적인 오류를 해결하는 지침은 섹션을 참조하세요the section called “Amazon Verified Permissions의 고객 관리형 키 문제 해결”.

암호화된 리소스에 필요한 IAM 정책

계정 내 IAM 역할을 통해 Verified Permissions를 호출하는 고객은 해당 IAM 정책에 리소스의 암호화 및 복호화에 고객 관리형 키 를 활용할 수 있는 적절한 권한이 있는지 확인해야 합니다.

로 암호화된 정책 스토어를 생성하는 경우 고객 관리형 키다음 IAM 정책은 필요한 최소 베어 AWS KMS 및 Verified Permissions 작업을 보여줍니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": "verifiedpermissions:CreatePolicyStore",
            "Resource": "*",
            "Effect": "Allow"
        },
        {
            "Action": [
                "kms:Decrypt",
                "kms:Encrypt",
                "kms:ReEncryptTo",
                "kms:ReEncryptFrom",
                "kms:DescribeKey",
                "kms:GenerateDataKeyWithoutPlaintext"
            ],
            "Resource": "*",
            "Effect": "Allow"
        }
    ]
}
```

> **Note**
>
> 로 암호화된 정책 스토어를 검색(Get* 및 List* 작업)하고 삭제하기 위해 추가 권한이 필요하지 고객 관리형 키않습니다.

로 암호화된 정책 스토어 업데이트 고객 관리형 키, (Get* 및 List* 작업) 검색, 로 암호화된 정책 스토어의 하위 리소스 업데이트 및 삭제 고객 관리형 키를 위해 다음 IAM 정책은 필요한 최소한의 베어 AWS KMS 와 Verified Permissions 작업을 보여줍니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": "verifiedpermissions:*",
            "Resource": "*",
            "Effect": "Allow"
        },
        {
            "Action": [
                "kms:Decrypt"
            ],
            "Resource": "*",
            "Effect": "Allow"
        }
    ]
}
```

단일 IAM 정책으로 고객은 IAM 역할 정책에 다음을 간단히 추가할 수 있습니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": "verifiedpermissions:*",
            "Resource": "*",
            "Effect": "Allow"
        },
        {
            "Action": [
                "kms:Decrypt",
                "kms:Encrypt",
                "kms:ReEncryptTo",
                "kms:ReEncryptFrom",
                "kms:DescribeKey",
                "kms:GenerateDataKeyWithoutPlaintext"
            ],
            "Resource": "*",
            "Effect": "Allow"
        }
```

```
    ]
}
```

암호화된 정책 스토어 관리

정책 스토어는 모든 관련 정책 리소스를 포함하는 엔트리 레벨 컨테이너입니다. 정책 저장소 및 하위 리소스의 계층 구조에 대한 자세한 내용은 Amazon Verified Permissions 사용 설명서의 Amazon Verified Permissions 정책 저장소를 참조하세요.

Verified Permissions에서 정책 스토어를 생성할 때 AWS KMS 키를 사용하여 저장 시 암호화를 활성화할 수 있습니다. 이렇게 하면 다음이 보장됩니다.

- 정책 스토어 및 하위 리소스에 고객 관리형 키 대한 모든 읽기, 업데이트 및 삭제 작업은 복호화 프로세스에 제공된를 사용합니다.
- 모든 권한 부여 결정 호출(예: IsAuthorized, BatchIsAuthorized, IsAuthorizedWithToken 등)은 고객 관리형 키 복호화 프로세스에 제공된를 사용합니다.

암호화된 정책 저장소 생성

암호화된 정책 스토어를 생성하기 전에 사용 고객 관리형 키 중인에 Amazon Verified Permissions가 암호화/복호화에 키를 활용하기 위한 적절한 키 정책 문이 설정되어 있는지 확인합니다. 필요한 권한은 the section called “Amazon Verified Permissions에 대한 AWS KMS 키 사용 권한 부여” 섹션을 참조하세요.

사용 AWS CLI:

```
aws verifiedpermissions create-policy-store --region us-east-1 --encryption-settings
 file://encrypted.json --validation-settings "{\"mode\": \"OFF\"}"
```

encrypted.json는 다음과 같습니다.

```
{
    "kmsEncryptionSettings": {
        "key": "arn:aws:kms:us-east-1:111122223333:key/12345678-90ab-cdef-ghij-
klmnopqrstuv",
        "encryptionContext": {
            "<ENCRYPTION_CONTEXT_KEY_1>": "<ENCRYPTION_CONTEXT_VALUE_1>",
            "<ENCRYPTION_CONTEXT_KEY_2>": "<ENCRYPTION_CONTEXT_VALUE_2>",
```

```
            ...
        }
    }
}
```

를 고객 관리형 키 ARNkey으로 바꾸고 및 <ENCRYPTION_CONTEXT_KEY> <ENCRYPTION_CONTEXT_VALUE> 페어를 원하는 encryptionContext 키-값 페어로 바꿉니다. 키-값 페어를 추가하지 않으려면를 완전히 생략할 encryptionContext 수 있습니다.

> **Important**
>
> 사용자 지정 암호화 컨텍스트에 키-값 페어aws:verifiedpermissions:policy-store-arn를 포함하지 마십시오. 이는 자동으로 추가되며 전달된 사용자 지정 암호화 컨텍스트 키-값 페어의 일부인 경우 검증 오류가 발생합니다.

정책 스토어의 하위 리소스에서 사용 가능한 APIs에 대한 자세한 내용은 Amazon Verified Permissions API 참조 안내서의 작업을 참조하세요.

> **Note**
>
> 잘못된 AWS KMS 키 정책으로 인해 Amazon Verified Permissions 리소스에서 사용 AWS KMS 고객 관리형 키 중인가 삭제, 비활성화 또는 액세스할 수 없는 경우 리소스의 복호화가 실패하여 오래된 권한 부여 결정이 발생합니다. 액세스 손실은 상황에 따라 일시적(키 정책을 수정할 수 있음)이거나 영구적(삭제된 키를 복원할 수 없음)일 수 있습니다. AWS KMS 키 삭제 또는 비활성화와 같은 중요한 작업에 대한 액세스를 제한하는 것이 좋습니다. 또한 조직에서 권한 있는 사용자가 Amazon Verified Permissions에 액세스할 수 없는 경우 AWS 에 액세스할 수 있도록 AWS 브레이크 글래스 액세스 절차를 설정하는 것이 좋습니다.

와의 Amazon Verified Permissions 상호 작용 모니터링 AWS KMS

를 고객 관리형 키 통해 Amazon Verified Permissions의 사용을 모니터링할 수 있습니다 AWS CloudTrail. Verified Permissions를 AWS KMS 통해에 대한 각 요청에는 암호화 컨텍스트와 사용 중인 키 ARN(본인 고객 관리형 키)이 요청 파라미터에 포함됩니다.

에 대한 AWS CloudTrail 로그 항목의 예GenerateDataKeyWithoutPlaintext:

```
{
```

```
    "eventVersion": "1.11",
    "userIdentity": {
        "type": "AWSService",
        "invokedBy": "verifiedpermissions.amazonaws.com"
    },
    "eventTime": "2025-09-28T16:51:04Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "GenerateDataKeyWithoutPlaintext",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "verifiedpermissions.amazonaws.com",
    "userAgent": "verifiedpermissions.amazonaws.com",
    "requestParameters": {
        "keyId": "arn:aws:kms:us-east-1:111122223333:key/abcdefgh-0123-ijkl-4567-
mnopqrstuvwx",
        "encryptionContext": {
            "aws:verifiedpermissions:policy-store-arn":
 "arn:aws:verifiedpermissions::111122223333:policy-store/PSt123456789012",
            "policy_store_editor": "Janus"
        },
        ...
    },
    ...
}
```

에 대한 AWS CloudTrail 로그 항목의 예`Decrypt`:

```
{
    "eventVersion": "1.11",
    "userIdentity": {
        "type": "AWSService",
        "invokedBy": "verifiedpermissions.amazonaws.com"
    },
    "eventTime": "2025-09-28T16:53:21Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "Decrypt",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "verifiedpermissions.amazonaws.com",
    "userAgent": "verifiedpermissions.amazonaws.com",
    "requestParameters": {
        "encryptionAlgorithm": "SYMMETRIC_DEFAULT",
        "keyId": "arn:aws:kms:us-east-1:111122223333:key/abcdefgh-0123-ijkl-4567-
mnopqrstuvwx",
        "encryptionContext": {
```

```
            "aws:verifiedpermissions:policy-store-arn":
 "arn:aws:verifiedpermissions::111122223333:policy-store/PSt123456789012",
            "policy_store_owner": "Elias"
        }
    },
    ...
}
```

에 대한 AWS CloudTrail 로그 항목의 예ReEncrypt:

```
{
    "eventVersion": "1.11",
    "userIdentity": {
        "type": "AWSService",
        "invokedBy": "verifiedpermissions.amazonaws.com"
    },
    "eventTime": "2025-09-28T16:51:04Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "ReEncrypt",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "verifiedpermissions.amazonaws.com",
    "userAgent": "verifiedpermissions.amazonaws.com",
    "requestParameters": {
        "sourceKeyId": "arn:aws:kms:us-east-1:111122223333:key/abcdefgh-0123-ijkl-4567-
mnopqrstuvwx",
        "destinationEncryptionContext": {
            "aws:verifiedpermissions:policy-store-arn":
 "arn:aws:verifiedpermissions::111122223333:policy-store/PSt123456789012"
        },
        "sourceEncryptionAlgorithm": "SYMMETRIC_DEFAULT",
        "destinationKeyId": "arn:aws:kms:us-east-1:111122223333:key/abcdefgh-0123-
ijkl-4567-mnopqrstuvwx",
        "sourceEncryptionContext": {
            "aws:verifiedpermissions:policy_store_arn":
 "arn:aws:verifiedpermissions::111122223333:policy-store/PSt123456789012"
        },
        "destinationEncryptionAlgorithm": "SYMMETRIC_DEFAULT",
        ...
    },
    ...
}
```

로그 항목에는 Amazon Verified Permissions의 보안 주체 `invokedBy` 참조 및 `requestParameters` 맵에 포함`encryptionContext/sourceEncryptionContext/destinationEncryptionContext`이 포함됩니다.

에 대한 AWS CloudTrail 로그 항목의 예`DescribeKey`:

```
{
    "eventVersion": "1.11",
    "userIdentity": {
        "type": "AWSService",
        "invokedBy": "verifiedpermissions.amazonaws.com"
    },
    "eventTime": "2025-09-28T16:51:02Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "DescribeKey",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "verifiedpermissions.amazonaws.com",
    "userAgent": "verifiedpermissions.amazonaws.com",
    "requestParameters": {
        "keyId": "arn:aws:kms:us-east-1:111122223333:key/abcdefgh-0123-ijkl-4567-
mnopqrstuvwx"
    },
    ...
}
```

로그 항목에는 Amazon Verified Permissions의 보안 주체 `invokedBy` 참조가 포함됩니다.

AWS CloudTrail 로그 항목에 대한 자세한 내용은 AWS CloudTrail 사용 설명서의 AWS CloudTrail 이벤트 이해를 참조하세요.

제한 사항

이 주제에서는 Verified Permissions 및 리소스 암호화를 고객 관리형 키위한 활용의 현재 제한 사항에 대해 설명합니다.

- 일단 활성화되면 정책 스토어에 대한 암호화를 비활성화할 수 없습니다.
- 암호화 없이 정책 스토어를 생성한 후에는에서 암호화하도록 정책 스토어를 업데이트할 수 없습니다. 고객 관리형 키
- 기존 암호화된 정책 스토어의에 고객 관리형 키 대한 Verified Permissions 액세스를 취소하면 기한 경과 권한 결정이 발생할 수 있습니다.

- 를 사용하여 정책 스토어를 생성한 후에는 사용자 지정 암호화 컨텍스트 값을 수정할 고객 관리형 키 수 없습니다. 이는 암호화된 정책 스토어 생성 중에 설정된 정적 값입니다.

Amazon Verified Permissions의 고객 관리형 키 문제 해결

이 주제에서는 Amazon Verified Permissions를 사용할 때 발생할 수 있는 일반적인 고객 관리형 키 관련 오류를 설명하고 이를 해결하기 위한 문제 해결 단계를 제공합니다.

액세스 거부: AWS KMS 권한 문제

오류: " 리소스가이 리전에 존재하지 않거나, 액세스를 허용하는 리소스 기반 정책이 없거나, 리소스 기반 정책이 액세스를 명시적으로 거부하기 때문에 서비스 또는 호출자가 제공된 AWS KMS 키를 사용할 권한이 없습니다."

이는 서비스 또는 호출자가 IAM 정책/AWS KMS 키 정책에 필요한 `kms:*` 작업(들) 권한이 없거나 참조되는 키가 존재하지 않거나 더 이상 존재하지 않음을 의미할 수 있습니다.

다음을 사용한 문제 해결 AWS CloudTrail:

- 에서 `kms.amazonaws.com` 이벤트 찾기 AWS CloudTrail
- 허용되지 않는 것으로 식별된 AWS KMS 작업의 이벤트 이름(예: `Decrypt`, `ReEncrypt`, `GenerateDataKeyWithoutPlaintext`, `DescribeKey`등)을 검색합니다.
- `errorCode` 및 `errorMessage` 필드를 검토합니다.
- `userIdentity`에서 작업을 시도한 위탁자를 확인합니다.

이 문제를 해결하려면 사용자 또는 IAM 보안 주체에게 IAM 정책 및 AWS KMS 키 정책에서 적절한 AWS KMS 작업 액세스 권한을 부여합니다. 자세한 내용은 the section called “전체 AWS KMS 키 정책” 섹션을 참조하세요.

검증 예외: AWS KMS 키 구성

오류: "구성된 AWS KMS 키에 유효한 구성이 없습니다"

즉, 현재 구성으로 인해 서비스에서 고객 관리형 키 암호화에 참조되는 키를 사용할 수 없습니다. 이유에는 비활성화된 키, 지원되지 않는 EncryptionAlgorithm 또는 지원되지 않는 KeyUsage 유형이 포함될 수 있습니다.

제한 예외: AWS KMS 속도 제한

오류: "전화할 수 있는 속도를 초과했습니다 AWS KMS"

이 오류는 키에 대한 암호화 작업 AWS KMS 제한을 초과했음을 의미합니다. https://docs.aws.amazon.com/kms/latest/developerguide/requests-per-second.html.

관련 정보

- Verified Permissions 정책 스토어 관리
- AWS KMS 모범 사례
- AWS KMS 암호화 컨텍스트
- AWS CloudTrail 통합
- AWS CloudTrail 로그 항목 예제

Amazon Verified Permissions의 Identity and Access Management

AWS Identity and Access Management (IAM)는 관리자가 AWS 리소스에 대한 액세스를 안전하게 제어할 수 AWS 서비스 있도록 도와주는 입니다. IAM 관리자는 Verified Permissions 리소스를 사용할 수 있는 인증(로그인) 및 권한(권한 있음)을 받을 수 AWS 서비스 있는 사용자를 제어합니다. IAM 는 추가 비용 없이 사용할 수 있는 입니다.

주제

- 대상
- ID를 통한 인증
- 정책을 사용하여 액세스 관리
- Amazon Verified Permissions의 작동 방식 IAM
- IAM Verified Permissions에 대한 정책
- Amazon Verified Permissions 자격 증명 기반 정책 예제
- AWS Amazon Verified Permissions에 대한 관리형 정책
- Amazon Verified Permissions 자격 증명 및 액세스 문제 해결

대상

AWS Identity and Access Management (IAM)를 사용하는 방법은 역할에 따라 다릅니다.

- 서비스 사용자 - 기능에 액세스할 수 없는 경우 관리자에게 권한 요청(참조Amazon Verified Permissions 자격 증명 및 액세스 문제 해결)

- 서비스 관리자 - 사용자 액세스 결정 및 권한 요청 제출(Amazon Verified Permissions의 작동 방식 IAM 참조)
- IAM 관리자 - 액세스를 관리하기 위한 정책 작성(참조Amazon Verified Permissions 자격 증명 기반 정책 예제)

ID를 통한 인증

인증은 자격 증명 자격 증명을 AWS 사용하여에 로그인하는 방법입니다. AWS 계정 루트 사용자, IAM 사용자 또는 IAM 역할을 수임하여 인증을 받아야 합니다.

AWS IAM Identity Center (IAM Identity Center), Single Sign-On 인증 또는 Google/Facebook 자격 증명과 같은 자격 증명 소스의 자격 증명을 사용하여 페더레이션 자격 증명으로 로그인할 수 있습니다. 로그인하는 방법에 대한 자세한 내용은 AWS 로그인 사용 설명서의 AWS 계정에 로그인하는 방법 섹션을 참조하세요.

프로그래밍 방식 액세스를 위해는 요청에 암호화 방식으로 서명할 수 있는 SDK 및 CLI를 AWS 제공합니다. 자세한 내용은 IAM 사용 설명서의 AWS API 요청에 대한 서명 버전 4를 참조하세요.

AWS 계정 루트 사용자

를 생성할 때 모든 AWS 서비스 및 리소스에 대한 완전한 액세스 권한이 있는 AWS 계정 theroot 사용자라는 하나의 로그인 자격 증명으로 AWS 계정시작합니다. 일상적인 태스크에 루트 사용자를 사용하지 않을 것을 강력히 권장합니다. 루트 사용자 자격 증명이 필요한 태스크는 IAM 사용 설명서의 루트 사용자 자격 증명이 필요한 태스크를 참조하세요.

페더레이션 ID

가장 좋은 방법은 인간 사용자에게 자격 증명 공급자와의 페더레이션을 사용하여 임시 자격 증명을 AWS 서비스 사용하여에 액세스하도록 요구하는 것입니다.

페더레이션 자격 증명은 엔터프라이즈 디렉터리, 웹 자격 증명 공급자 또는 자격 증명 소스의 자격 증명을 AWS 서비스 사용하여 Directory Service 에 액세스하는 사용자입니다. 페더레이션 ID는 임시 자격 증명을 제공하는 역할을 수임합니다.

중앙 집중식 액세스 관리를 위해 AWS IAM Identity Center를 추천합니다. 자세한 정보는 AWS IAM Identity Center 사용 설명서의 What is IAM Identity Center?를 참조하세요.

IAM 사용자 및 그룹

IAM 사용자는 단일 개인 또는 애플리케이션에 대한 특정 권한을 가진 ID입니다. 장기 자격 증명이 있는 IAM 사용자 대신 임시 자격 증명을 사용하는 것이 좋습니다. 자세한 내용은 IAM 사용 설명서의 자격 증명 공급자와의 페더레이션을 사용하여 임시 자격 증명을 AWS 사용하여에 액세스하도록 인간 사용자에게 요구하기를 참조하세요.

IAM 그룹은 IAM 사용자 모음을 지정하고 대규모 사용자 집합에 대한 관리 권한을 더 쉽게 만듭니다. 자세한 내용은 사용 IAM 설명서의 IAM 사용자 사용 사례를 참조하세요.

IAM 역할

IAM 역할은 특정 권한이 AWS 계정 있는 내의 자격 증명입니다. IAM 사용자와 유사하지만, 특정 개인과 연결되지 않습니다. IAM 역할을 전환 AWS Management Console 하여에서 역할을 일시적으로 수임할 수 있습니다. AWS CLI 또는 AWS API 작업을 호출하거나 사용자 지정 URL을 사용하여 역할을 수임할 수 있습니다. 역할 사용 방법에 대한 자세한 내용은 IAM 사용 설명서의 IAM 역할 사용을 참조하세요.

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 IAM 사용 설명서의 서드 파티 자격 증명 공급자(페더레이션)를 위한 역할 생성을 참조하세요. IAM Identity Center를 사용하는 경우, 권한 집합을 구성합니다. 인증 후 ID가 액세스할 수 있는 항목을 관리하기 위해 IAM Identity Center는 권한 세트를 IAM의 역할과 연관 짓습니다. 권한 세트에 대한 자세한 내용은 AWS IAM Identity Center 사용 설명서의 권한 세트를 참조하세요.
- 임시 IAM 사용자 권한 - IAM 사용자 또는 역할은 IAM 역할을 수임하여 특정 작업에 대해 다른 권한을 일시적으로 수임할 수 있습니다.
- 크로스 계정 액세스 – IAM 역할을 사용하여 다른 계정의 사용자(신뢰할 수 있는 보안 주체)가 내 계정의 리소스에 액세스하도록 할 수 있습니다. 역할은 교차 계정 액세스를 부여하는 기본적인 방법입니다. 그러나 일부 에서는 정책을 리소스에 직접 연결할 AWS 서비스수 있습니다(역할을 프록시로 사용하는 대신). 교차 계정 액세스를 위한 역할과 리소스 기반 정책의 차이점을 알아보려면 IAM 사용 설명서의 IAM 역할이 리소스 기반 정책과 어떻게 다른지 참조하세요.
- 에서 실행되는 애플리케이션 Amazon EC2 - IAM 역할을 사용하여 EC2 인스턴스에서 실행되고 AWS CLI 또는 AWS API 요청을 수행하는 애플리케이션의 임시 자격 증명을 관리할 수 있습니다. 이것은 EC2 인스턴스 내에 액세스 키를 저장하는 경우에 바람직한 방법입니다. EC2 인스턴스에 AWS 역할을 할당하고 모든 애플리케이션에서 사용할 수 있도록 하려면 인스턴스에 연결된 인스턴스 프

로파일을 생성합니다. 인스턴스 프로필에는 역할이 포함되어 있으며 EC2 인스턴스에서 실행되는 프로그램이 임시 보안 인증을 얻을 수 있습니다. 자세한 내용은 IAM 사용 설명서의 IAM 역할을 사용하여 Amazon EC2 인스턴스에서 실행되는 애플리케이션에 권한 부여를 참조하세요.

IAM 역할 또는 IAM 사용자를 사용할지 여부를 알아보려면 IAM 사용 설명서의 IAM 역할 생성 시기(사용자 대신)를 참조하세요.

정책을 사용하여 액세스 관리

정책을 AWS 생성하고 자격 증명 또는 리소스에 연결하여 AWS 에서 액세스를 제어합니다. 정책은 자격 증명 또는 리소스와 연결될 때 권한을 정의합니다.는 보안 주체가 요청할 때 이러한 정책을 AWS 평가합니다. 대부분의 정책은에 JSON 문서 AWS 로 저장됩니다. JSON 정책 문서에 대한 자세한 내용은 IAM 사용 설명서의 JSON 정책 개요를 참조하세요.

정책을 사용하여 관리자는 어떤 보안 주체가 어떤 리소스에 대해 어떤 조건에서 작업을 수행할 수 있는지 정의하여 누가 무엇을 액세스할 수 있는지 지정합니다.

기본적으로 사용자 및 역할에는 어떠한 권한도 없습니다. IAM 관리자는 IAM 정책을 생성하여 역할에 추가한 다음 사용자가 작업을 수행하는 데 사용되는 방법에 관계없이 assume. IAM policies에서 권한을 정의할 수 있습니다.

ID 기반 정책

ID 기반 정책은 ID(사용자, 사용자 그룹 또는 역할)에 연결하는 JSON 권한 정책 문서입니다. 이러한 정책은 자격 증명이 수행할 수 있는 작업, 대상 리소스 및 이에 관한 조건을 제어합니다. 자격 증명 기반 정책을 생성하는 방법을 알아보려면 IAM 사용 설명서의 고객 관리형 정책을 사용하여 사용자 지정 IAM 권한 정의를 참조하세요.

ID 기반 정책은 인라인 정책(단일 ID에 직접 포함) 또는 관리형 정책(여러 ID에 연결된 독립 실행형 정책)일 수 있습니다. 관리형 정책과 인라인 정책 중에서 선택하는 방법을 알아보려면 IAM 사용 설명서의 관리형 정책과 인라인 정책 중에서 선택을 참조하세요.

리소스 기반 정책

리소스 기반 정책은 리소스에 연결하는 JSON 정책 설명서입니다. 예를 들면 IAM 역할 신뢰 정책 및 Amazon S3 버킷 정책이 있습니다. 리소스 기반 정책을 지원하는 서비스에서 서비스 관리자는 이러한 정책을 사용하여 특정 리소스에 대한 액세스를 통제할 수 있습니다. 리소스 기반 정책에서 보안 주체를 지정해야 합니다.

리소스 기반 정책은 해당 서비스에 있는 인라인 정책입니다. 리소스 기반 정책의 IAM 에서는 AWS 관리형 정책을 사용할 수 없습니다.

액세스 제어 목록(ACL)

액세스 제어 목록(ACL)은 어떤 위탁자(계정 멤버, 사용자 또는 역할)가 리소스에 액세스할 수 있는 권한을 가지고 있는지를 제어합니다. ACL은 JSON 정책 문서 형식을 사용하지 않지만 리소스 기반 정책과 유사합니다.

Amazon S3 AWS WAF, 및 Amazon VPC는 ACLs. ACL에 관한 자세한 내용은 Amazon Simple Storage Service 개발자 가이드의 액세스 제어 목록(ACL) 개요를 참조하세요.

기타 정책 타입

AWS 는 보다 일반적인 정책 유형에서 부여한 최대 권한을 설정할 수 있는 추가 정책 유형을 지원합니다.

- 권한 경계 - 자격 증명 기반 정책이 IAM 엔터티에 부여할 수 있는 최대 권한을 설정합니다. 자세한 내용은 IAM 사용 설명서의 IAM 개체에 대한 권한 경계를 참조하세요.
- 서비스 제어 정책(SCP) - AWS Organizations내 조직 또는 조직 단위에 대한 최대 권한을 지정합니다. 자세한 내용은AWS Organizations 사용 설명서의 서비스 제어 정책을 참조하세요.
- 리소스 제어 정책(RCP) – 계정의 리소스에 사용할 수 있는 최대 권한을 설정합니다. 자세한 내용은 AWS Organizations 사용 설명서의 리소스 제어 정책(RCP)을 참조하세요.
- 세션 정책 – 역할 또는 페더레이션 사용자에 대해 임시 세션을 프로그래밍 방식으로 생성할 때 파라미터로 전달하는 고급 정책입니다. 자세한 정보는 IAM 사용 설명서의 세션 정책을 참조하세요.

여러 정책 유형

여러 정책 유형이 요청에 적용되는 경우, 결과 권한은 이해하기가 더 복잡합니다. 에서 여러 정책 유형이 관련될 때 요청을 허용할지 여부를 AWS 결정하는 방법을 알아보려면 IAM 사용 설명서의 정책 평가 로직을 참조하세요.

Amazon Verified Permissions의 작동 방식 IAM

IAM 를 사용하여 Verified Permissions에 대한 액세스를 관리하기 전에 Verified Permissions에서 사용할 수 있는 IAM 기능에 대해 알아봅니다.

IAM Amazon Verified Permissions에서 사용할 수 있는 기능

IAM 기능	Verified Permissions 지원
자격 증명 기반 정책	예
리소스 기반 정책	아니요
정책 작업	예
정책 리소스	예
정책 조건 키	아니요
ACL	아니요
ABAC(정책의 태그)	예
임시 보안 인증	예
엔터티 권한	예
서비스 역할	아니요
서비스 연결 역할	아니요

Verified Permissions 및 기타 AWS 서비스에서 대부분의 IAM 기능을 사용하는 방법을 전체적으로 알아보려면 IAM 사용 설명서의 AWS 로 작업하는 서비스를 IAM 참조하세요.

Verified Permissions 자격 증명 기반 정책

보안 인증 기반 정책 지원	예

자격 증명 기반 정책은 IAM 사용자, 사용자 그룹 또는 역할과 같은 자격 증명에 연결할 수 있는 JSON 권한 정책 문서입니다. 이러한 정책은 사용자 및 역할이 어떤 리소스와 어떤 조건에서 어떤 작업을 수행할 수 있는지를 제어합니다. 자격 증명 기반 정책을 생성하는 방법을 알아보려면 IAM 사용 설명서의 고객 관리형 정책을 사용하여 사용자 지정 IAM 권한 정의를 참조하세요.

IAM 자격 증명 기반 정책을 사용하면 허용되거나 거부된 작업 및 리소스와 작업이 허용되거나 거부되는 조건을 지정할 수 있습니다. JSON 정책에서 사용하는 모든 요소에 대해 알아보려면 IAM 사용 설명서의 IAM JSON 정책 요소 참조를 참조하세요.

Verified Permissions 자격 증명 기반 정책 예제

Verified Permissions 자격 증명 기반 정책의 예제를 보려면 Amazon Verified Permissions 자격 증명 기반 정책 예제을(를) 참조하세요.

Verified Permissions 내 리소스 기반 정책

리소스 기반 정책 지원	아니요

리소스 기반 정책은 리소스에 연결하는 JSON 정책 문서입니다. 리소스 기반 정책의 예로는 IAM 역할 신뢰 정책 및 Amazon S3 버킷 정책이 있습니다. 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, 페더레이션 사용자 또는가 포함될 수 있습니다 AWS 서비스.

교차 계정 액세스를 활성화하려면 전체 계정 또는 다른 계정의 IAM 엔터티를 리소스 기반 정책의 보안 주체로 지정할 수 있습니다. 자세한 내용은 IAM 사용 설명서의 에서 교차 계정 리소스 액세스를 IAM 참조하세요.

Verified Permissions 정책 작업

정책 작업 지원	예

관리자는 AWS JSON 정책을 사용하여 누가 무엇에 액세스할 수 있는지 지정할 수 있습니다. 즉, 어떤 보안 주체가 어떤 리소스와 어떤 조건에서 작업을 수행할 수 있는지를 지정할 수 있습니다.

JSON 정책의 `Action`요소는 정책에서 액세스를 허용하거나 거부하는 데 사용할 수 있는 작업을 설명합니다. 연결된 작업을 수행할 수 있는 권한을 부여하기 위한 정책에 작업을 포함하세요.

Verified Permissions 작업 목록을 보려면 서비스 권한 부여 참조의 Amazon Verified Permissions에서 정의한 작업을 참조하세요.

Verified Permissions의 정책 작업은 작업 앞에 다음 접두사를 사용합니다.

```
verifiedpermissions
```

단일 문에서 여러 작업을 지정하려면 쉼표로 구분합니다.

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

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

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

Verified Permissions 자격 증명 기반 정책의 예제를 보려면 Amazon Verified Permissions 자격 증명 기반 정책 예제을(를) 참조하세요.

Verified Permissions 정책 리소스

정책 리소스 지원	예

관리자는 AWS JSON 정책을 사용하여 누가 무엇에 액세스할 수 있는지 지정할 수 있습니다. 즉, 어떤 보안 주체가 어떤 리소스와 어떤 조건에서 작업을 수행할 수 있는지를 지정할 수 있습니다.

Resource JSON 정책 요소는 작업이 적용되는 하나 이상의 객체를 지정합니다. 모범 사례에 따라 Amazon 리소스 이름(ARN)을 사용하여 리소스를 지정합니다. 리소스 수준 권한을 지원하지 않는 작업의 경우, 와일드카드(*)를 사용하여 해당 문이 모든 리소스에 적용됨을 나타냅니다.

```
"Resource": "*"
```

Verified Permissions 리소스 유형 및 해당 ARN 목록을 보려면 서비스 권한 부여 참조의 Amazon Verified Permissions에서 정의한 리소스 유형을 참조하세요. 각 리소스의 ARN을 지정할 수 있는 작업을 알아보려면 Amazon Verified Permissions에서 정의한 작업을 참조하세요.

Verified Permissions 정책 조건 키

서비스별 정책 조건 키 지원	아니요

관리자는 AWS JSON 정책을 사용하여 누가 무엇에 액세스할 수 있는지 지정할 수 있습니다. 즉, 어떤 보안 주체가 어떤 리소스와 어떤 조건에서 작업을 수행할 수 있는지를 지정할 수 있습니다.

`Condition` 요소는 정의된 기준에 따라 문이 실행되는 시기를 지정합니다. 같음(equals) 또는 미만(less than)과 같은 조건 연산자를 사용하여 정책의 조건을 요청의 값과 일치시키는 조건식을 생성할 수 있습니다. 모든 AWS 전역 조건 키를 보려면 IAM 사용 설명서의 AWS 전역 조건 컨텍스트 키를 참조하세요.

Verified Permissions ACL

ACL 지원	아니요

ACL(액세스 통제 목록)은 어떤 보안 주체(계정 멤버, 사용자 또는 역할)가 리소스에 액세스할 수 있는 권한을 가지고 있는지를 제어합니다. ACL은 JSON 정책 문서 형식을 사용하지 않지만 리소스 기반 정책과 유사합니다.

Verified Permissions ABAC

ABAC 지원(정책의 태그)	예

속성 기반 액세스 제어(ABAC)는 태그라고 불리는 속성을 기반으로 권한을 정의하는 권한 부여 전략입니다. IAM 엔터티 및 AWS 리소스에 태그를 연결한 다음 보안 주체의 태그가 리소스의 태그와 일치할 때 작업을 허용하는 ABAC 정책을 설계할 수 있습니다.

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.

ABAC에 대한 자세한 내용은 IAM 사용 설명서의 ABAC 권한 부여로 권한 정의를 참조하세요. ABAC 설정 단계가 포함된 튜토리얼을 보려면 IAM 사용 설명서의 ABAC(속성 기반 액세스 제어) 사용을 참조하세요.

Verified Permissions에서 임시 자격 증명 사용

임시 보안 인증 지원	예

임시 자격 증명은 AWS 리소스에 대한 단기 액세스를 제공하며 페더레이션 또는 전환 역할을 사용할 때 자동으로 생성됩니다. 장기 액세스 키를 사용하는 대신 임시 자격 증명을 동적으로 생성하는 것이 AWS 좋습니다. 자세한 내용은 IAM 사용 설명서의 IAM 및에서 작동하는의 임시 보안 자격 증명을 참조하세요. AWS 서비스IAM

Verified Permissions의 서비스 간 보안 주체 권한

보안 주체 권한 지원	예

전달 액세스 세션(FAS)은를 호출하는 보안 주체의 권한을 다운스트림 서비스에 AWS 서비스 대한 요청과 AWS 서비스함께 사용합니다. FAS 요청 시 정책 세부 정보는 전달 액세스 세션을 참조하세요.

Verified Permissions 서비스 역할

서비스 역할 지원	아니요

서비스 역할은 서비스가 사용자를 대신하여 작업을 수행하기 위해 수임하는 IAM 역할입니다. IAM 관리자는 내에서 서비스 역할을 생성, 수정 및 삭제할 수 있습니다 IAM. 자세한 내용은 IAM 사용 설명서의 에 권한을 위임할 역할 생성을 AWS 서비스 참조하세요.

Verified Permissions 서비스 연결 역할

서비스 연결 역할 지원	아니요

서비스 연결 역할은에 연결된 서비스 역할의 한 유형입니다 AWS 서비스. 서비스는 사용자를 대신하여 작업을 수행하기 위해 역할을 수임할 수 있습니다. 서비스 연결 역할은에 표시 AWS 계정 되며 서비스가 소유합니다. IAM 관리자는 서비스 연결 역할에 대한 권한을 볼 수 있지만 편집할 수는 없습니다.

서비스 연결 역할 생성 또는 관리에 대한 자세한 내용은 AWS 에서 작업하는 서비스를 IAM 참조하세요. 서비스 연결 역할 열에서 Yes가 포함된 서비스를 테이블에서 찾습니다. 해당 서비스에 대한 서비스 연결 역할 설명서를 보려면 예(Yes) 링크를 선택합니다.

IAM Verified Permissions에 대한 정책

Verified Permissions는 애플리케이션 내 사용자 권한을 관리합니다. 애플리케이션이 Verified Permissions APIs를 호출하거나 AWS Management Console 사용자가 Verified Permissions 정책 스토어에서 Cedar 정책을 관리할 수 있도록 하려면 필요한 IAM 권한을 추가해야 합니다.

자격 증명 기반 정책은 IAM 사용자, 사용자 그룹 또는 역할과 같은 자격 증명에 연결할 수 있는 JSON 권한 정책 문서입니다. 이러한 정책은 사용자 및 역할이 어떤 리소스와 어떤 조건에서 어떤 작업을 수행할 수 있는지를 제어합니다. 자격 증명 기반 정책을 생성하는 방법을 알아보려면 IAM 사용 설명서의 IAM 정책 생성을 참조하세요.

IAM 자격 증명 기반 정책을 사용하면 허용되거나 거부된 작업 및 리소스와 작업이 허용되거나 거부되는 조건(아래 나열됨)을 지정할 수 있습니다. ID 기반 정책에서는 위탁자가 연결된 사용자 또는 역할에 적용되므로 위탁자를 지정할 수 없습니다. JSON 정책에서 사용할 수 있는 모든 요소에 대해 알아보려면 IAM 사용 설명서의 IAM JSON 정책 요소 참조를 참조하세요.

작업	설명
CreateIdentitySource	새 자격 증명 소스를 생성하는 작업입니다.
CreatePolicy	정책 스토어에 Cedar 정책을 생성하는 작업입니다. 정적 정책을 생성하거나 정책 템플릿에 연결된 정책을 생성할 수 있습니다.
CreatePolicyStore	새 정책 스토어를 생성하는 작업입니다.
CreatePolicyTemplate	새 정책 템플릿을 생성하는 작업입니다.
DeleteIdentitySource	자격 증명 소스를 삭제하는 작업입니다.
DeletePolicy	정책 스토어에서 정책을 삭제하는 작업입니다.
DeletePolicyStore	정책 스토어를 삭제하는 작업입니다.
DeletePolicyTemplate	정책 템플릿을 삭제하는 작업입니다.
GetIdentitySource	자격 증명 소스를 가져오는 작업입니다.
GetPolicy	지정된 정책에 대한 정보를 검색하는 작업입니다.

작업	설명
GetPolicyStore	지정된 정책 스토어에 대한 정보를 검색하는 작업입니다.
GetPolicyTemplate	정책 템플릿을 가져오는 작업입니다.
GetSchema	스키마를 가져오는 작업입니다.
IsAuthorized	권한 부여 요청에 설명된 매개 변수를 기반으로 권한 부여 응답을 받기 위한 작업입니다.
IsAuthorizedWithToken	보안 주체가 자격 증명 토큰에서 오는 권한 부여 요청에 설명된 파라미터를 기반으로 권한 부여 응답을 가져오는 작업입니다. ???
ListIdentitySources	의 모든 자격 증명 소스를 나열하는 작업입니다 AWS 계정.
ListPolicies	정책 스토어의 모든 정책을 나열하는 작업입니다.
ListPolicyStores	의 모든 정책 스토어를 나열하는 작업입니다 AWS 계정.
ListPolicyTemplates	의 모든 정책 템플릿을 나열하는 작업입니다 AWS 계정.
ListTagsForResource	리소스의 모든 태그를 나열하는 작업입니다.
PutSchema	정책 저장소에 스키마를 추가하는 작업입니다.
TagResource	리소스에 태그를 추가하는 작업입니다.
UpdateIdentitySource	자격 증명 소스를 업데이트하는 작업입니다.
UpdatePolicy	정책 스토어에서 정책을 업데이트하는 작업입니다.
UpdatePolicyStore	정책 스토어를 업데이트하는 작업입니다.

작업	설명
UpdatePolicyTemplate	정책 템플릿을 업데이트하는 작업입니다.
UntagResource	리소스에서 태그를 제거하는 작업입니다.

CreatePolicy 작업에 IAM 대한 권한 정책 예제:

JSON

```
{
"Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "verifiedpermissions:CreatePolicy"
            ],
            "Resource": "*"
        }
    ]
 }
```

Amazon Verified Permissions 자격 증명 기반 정책 예제

기본적으로 사용자 및 역할에는 Verified Permissions 리소스를 생성하거나 수정할 수 있는 권한이 없습니다. 또한 AWS Management Console, AWS Command Line Interface (AWS CLI) 또는 AWS API를 사용하여 작업을 수행할 수 없습니다. IAM 관리자는 필요한 리소스에 대한 작업을 수행할 수 있는 권한을 사용자 및 역할에 부여하는 IAM 정책을 생성해야 합니다. 그런 다음 관리자는 해당 권한이 필요한 사용자에 이러한 정책을 연결해야 합니다.

이러한 예제 JSON 정책 문서를 사용하여 자격 IAM 증명 기반 정책을 생성하는 방법을 알아보려면 IAM 사용 설명서의 정책 생성을 IAM 참조하세요.

각 리소스 유형에 대한 ARN 형식을 포함하여 Verified Permissions에서 정의한 작업 및 리소스 유형에 대한 자세한 내용은 서비스 권한 부여 참조에서 Amazon Verified Permissions에서 사용되는 작업, 리소스 및 조건 키를 참조하세요.

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. 이 작업으로 인해 AWS 계정에 비용이 발생할 수 있습니다. ID 기반 정책을 생성하거나 편집할 때는 다음 지침과 권장 사항을 따르세요.

- AWS 관리형 정책을 시작하고 최소 권한으로 전환 - 사용자 및 워크로드에 권한 부여를 시작하려면 많은 일반적인 사용 사례에 대한 권한을 부여하는 AWS 관리형 정책을 사용합니다. 에서 사용할 수 있습니다 AWS 계정. 사용 사례에 맞는 AWS 고객 관리형 정책을 정의하여 권한을 추가로 줄이는 것이 좋습니다. 자세한 정보는 IAM 사용 설명서의 AWS 관리형 정책 또는 직무에 관한AWS 관리형 정책을 참조하세요.
- 최소 권한 적용 - IAM 정책으로 권한을 설정할 때 작업을 수행하는 데 필요한 권한만 부여합니다. 이렇게 하려면 최소 권한으로 알려진 특정 조건에서 특정 리소스에 대해 수행할 수 있는 작업을 정의합니다. 를 사용하여 권한을 적용하는 IAM 방법에 대한 자세한 내용은 IAM 사용 설명서의 의 정책 및 권한을 IAM 참조하세요.
- IAM 정책의 조건을 사용하여 액세스를 추가로 제한 - 정책에 조건을 추가하여 작업 및 리소스에 대한 액세스를 제한할 수 있습니다. 예를 들어, SSL을 사용하여 모든 요청을 전송해야 한다고 지정하는 정책 조건을 작성할 수 있습니다. AWS 서비스와 같은 특정를 통해 사용되는 경우 조건을 사용하여 서비스 작업에 대한 액세스 권한을 부여할 수도 있습니다 CloudFormation. 자세한 내용은 IAM 사용 설명서의 IAM JSON 정책 요소: 조건을 참조하세요.
- IAM Access Analyzer를 사용하여 IAM 정책을 검증하여 안전하고 기능적인 권한을 보장합니다. IAM Access Analyzer는 정책이 정책 IAM 언어(JSON) 및 IAM 모범 사례를 준수하도록 신규 및 기존 정책을 검증합니다. IAM Access Analyzer는 100개 이상의 정책 확인 항목과 실행 가능한 추천을 제공하여 안전하고 기능적인 정책을 작성하도록 돕습니다. 자세한 내용은 IAM 사용 설명서의 IAM Access Analyzer로 정책 검증을 참조하세요.
- 다중 인증(MFA) 필요 -에서 IAM 사용자 또는 루트 사용자가 필요한 시나리오가 있는 경우 추가 보안을 위해 MFA를 AWS 계정켭니다. API 작업을 직접적으로 호출할 때 MFA가 필요하면 정책에 MFA 조건을 추가합니다. 자세한 내용은 IAM 사용 설명서의 MFA를 사용한 보안 API 액세스를 참조하세요.

의 모범 사례에 대한 자세한 내용은 IAM 사용 설명서의 의 보안 모범 사례를 IAM IAM참조하세요.

Verified Permissions 콘솔 사용

Amazon Verified Permissions 콘솔에 액세스하려면 최소한의 권한 집합이 있어야 합니다. 이러한 권한은에서 Verified Permissions 리소스에 대한 세부 정보를 나열하고 볼 수 있도록 허용해야 합니다 AWS 계정. 최소 필수 권한보다 더 제한적인 ID 기반 정책을 생성하는 경우, 콘솔이 해당 정책에 연결된 엔티티(사용자 또는 역할)에 대해 의도대로 작동하지 않습니다.

AWS CLI 또는 AWS API만 호출하는 사용자에게 최소 콘솔 권한을 허용할 필요는 없습니다. 대신, 수행하려는 API 작업과 일치하는 작업에만 액세스할 수 있도록 합니다.

사용자와 역할이 Verified Permissions 콘솔을 계속 사용할 수 있도록 하려면 Verified Permissions *ConsoleAccess* 또는 *ReadOnly* AWS 관리형 정책도 엔터티에 연결합니다. 자세한 내용은 IAM 사용 설명서의 사용자에게 권한 추가를 참조하세요.

사용자가 자신의 고유한 권한을 볼 수 있도록 허용

이 예제는 IAM 사용자가 자신의 사용자 ID에 연결된 인라인 및 관리형 정책을 볼 수 있도록 허용하는 정책을 생성하는 방법을 보여줍니다. 이 정책에는 콘솔에서 또는 AWS CLI 또는 AWS API를 사용하여 프로그래밍 방식으로이 작업을 완료할 수 있는 권한이 포함됩니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ViewOwnUserInfo",
            "Effect": "Allow",
            "Action": [
                "iam:GetUserPolicy",
                "iam:ListGroupsForUser",
                "iam:ListAttachedUserPolicies",
                "iam:ListUserPolicies",
                "iam:GetUser"
            ],
            "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": "*"
        }
    ]
}
```

AWS Amazon Verified Permissions에 대한 관리형 정책

사용자, 그룹 및 역할에 권한을 추가하려면 직접 정책을 작성하는 것보다 AWS 관리형 정책을 사용하는 것이 더 쉽습니다. 팀에 필요한 권한만 제공하는 IAM 고객 관리형 정책을 생성하는 데는 시간과 전문 지식이 필요합니다. 빠르게 시작하려면 AWS 관리형 정책을 사용할 수 있습니다. 이 정책은 일반적인 사용 사례를 다루며 사용자의 AWS 계정에서 사용할 수 있습니다. AWS 관리형 정책에 대한 자세한 내용은 IAM 사용 설명서의 AWS 관리형 정책을 참조하세요.

AWS 서비스는 AWS 관리형 정책을 유지 관리하고 업데이트합니다. AWS 관리형 정책에서는 권한을 변경할 수 없습니다. 서비스에서 때때로 추가 권한을 AWS 관리형 정책에 추가하여 새로운 기능을 지원합니다. 이 유형의 업데이트는 정책이 연결된 모든 ID(사용자, 그룹 및 역할)에 적용됩니다. 서비스는 새로운 기능이 시작되거나 새 작업을 사용할 수 있을 때 AWS 관리형 정책에 업데이트됩니다. 서비스는 AWS 관리형 정책에서 권한을 제거하지 않으므로 정책 업데이트로 인해 기존 권한이 손상되지 않습니다.

또한 여러 서비스에 걸쳐 있는 직무에 대한 관리형 정책을 AWS 지원합니다. 예를 들어 ReadOnlyAccess AWS 관리형 정책은 모든 AWS 서비스 및 리소스에 대한 읽기 전용 액세스를 제공합니다. 서비스가 새 기능을 시작하면는 새 작업 및 리소스에 대한 읽기 전용 권한을 AWS 추가합니다. 직무 정책의 목록과 설명은 IAM 사용 설명서의 AWS 직무에 대한 관리형 정책을 참조하세요.

AWS 관리형 정책: AmazonVerifiedPermissionsFullAccess

`AmazonVerifiedPermissionsFullAccess` 관리형 정책은 Verified Permissions에 대한 전체 액세스 권한을 부여합니다. Amazon Cognito기반 자격 증명 소스로 작업하려면 AmazonCognitoReadOnly 정책과 같은 별도의 정책을 연결해야 합니다.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "AccountLevelPermissions",
      "Effect": "Allow",
      "Action": [
        "verifiedpermissions:CreatePolicyStore",
        "verifiedpermissions:ListPolicyStores"
      ],
      "Resource": "*"
    },
    {
      "Sid": "PolicyStoreLevelPermissions",
      "Effect": "Allow",
      "Action": [
        "verifiedpermissions:*"
      ],
      "Resource": [
        "arn:aws:verifiedpermissions::*:policy-store/*"
      ]
    }
  ]
}
```

AWS 관리형 정책: AmazonVerifiedPermissionsReadOnlyAccess

`AmazonVerifiedPermissionsReadOnlyAccess` 관리형 정책은 Verified Permissions에 대한 읽기 전용 액세스 권한을 부여합니다.

이 정책은 권한 부여 쿼리 APIs `IsAuthorized` 및를 포함하여 Amazon Verified Permissions의 모든 읽기 작업에 대한 액세스 권한을 부여합니다`IsAuthorizedWithToken`.

> **Note**
>
> BatchIsAuthorized 및에 대한 액세스 BatchIsAuthorizedWithToken 권한은 IsAuthorizedWithToken각각 IsAuthorized 및에 대한 액세스 권한이 부여될 때 자동으로 부여됩니다.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "AccountLevelPermissions",
      "Effect": "Allow",
      "Action": [
        "verifiedpermissions:ListPolicyStores"
      ],
      "Resource": "*"
    },
    {
      "Sid": "PolicyStoreLevelPermissions",
      "Effect": "Allow",
      "Action": [
        "verifiedpermissions:GetIdentitySource",
        "verifiedpermissions:GetPolicy",
        "verifiedpermissions:GetPolicyStore",
        "verifiedpermissions:GetPolicyTemplate",
        "verifiedpermissions:GetSchema",
        "verifiedpermissions:IsAuthorized",
        "verifiedpermissions:IsAuthorizedWithToken",
        "verifiedpermissions:ListIdentitySources",
        "verifiedpermissions:ListPolicies",
        "verifiedpermissions:ListPolicyTemplates"
      ],
      "Resource": [
        "arn:aws:verifiedpermissions::*:policy-store/*"
      ]
    }
  ]
}
```

AWS 관리형 정책에 대한 Verified Permissions 업데이트

이 서비스가 이러한 변경 사항을 추적하기 시작한 이후 Verified Permissions의 AWS 관리형 정책 업데이트에 대한 세부 정보를 봅니다. 이 페이지의 변경 사항에 대한 자동 알림을 받으려면 Verified Permissions 문서 기록 페이지에서 RSS 피드를 구독하세요.

변경	설명	Date
AmazonVerifiedPermissionsFullAccess – 새 정책	Verified Permissions는 Verified Permissions에 대한 전체 액세스를 허용하는 새 정책을 추가했습니다.	2024년 10월 11일
AmazonVerifiedPermissionsReadOnlyAccess – 새 정책	Verified Permissions는 권한 부여 쿼리 APIs `IsAuthorized` 및를 포함하여 Amazon Verified Permissions의 모든 읽기 작업에 대한 액세스를 허용하는 새 정책을 추가했습니다`IsAuthorizedWithToken` .	2024년 10월 11일
Verified Permissions에서 변경 사항 추적 시작	Verified Permissions가 AWS 관리형 정책에 대한 변경 사항 추적을 시작했습니다.	2024년 10월 11일

Amazon Verified Permissions 자격 증명 및 액세스 문제 해결

다음 정보를 사용하여 Verified Permissions 및 IAM에서 발생할 수 있는 공통적인 문제를 진단하고 수정할 수 있습니다.

주제

- Verified Permissions에서 작업을 수행할 권한이 없음
- iam:PassRole을 수행하도록 인증되지 않음
- 내 외부의 사람이 내 Verified Permissions 리소스에 액세스 AWS 계정 하도록 허용하고 싶습니다.

Verified Permissions에서 작업을 수행할 권한이 없음

작업을 수행할 권한이 없다는 오류가 표시되면 작업을 수행할 수 있도록 정책을 업데이트해야 합니다.

다음의 예제 오류는 mateojackson IAM 사용자가 콘솔을 사용하여 가상 *my-example-widget* 리소스에 대한 세부 정보를 보려고 하지만 가상 verifiedpermissions:*GetWidget* 권한이 없을 때 발생합니다.

```
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 액세스할 수 있도록 mateojackson 사용자 정책을 업데이트해야 합니다.

도움이 필요한 경우 AWS 관리자에게 문의하세요. 관리자는 로그인 자격 증명을 제공한 사람입니다.

iam:PassRole을 수행하도록 인증되지 않음

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

일부 AWS 서비스 에서는 새 서비스 역할 또는 서비스 연결 역할을 생성하는 대신 기존 역할을 해당 서비스에 전달할 수 있습니다. 이렇게 하려면 역할을 서비스에 전달할 권한이 있어야 합니다.

다음 오류 예제는 marymajor(이)라는 IAM 사용자가 콘솔을 사용하여 Verified Permissions에서 작업을 수행하려고 하는 경우에 발생합니다. 하지만 작업을 수행하려면 서비스 역할이 부여한 권한이 서비스에 있어야 합니다. Mary는 서비스에 역할을 전달할 권한이 없습니다.

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

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

도움이 필요한 경우 AWS 관리자에게 문의하세요. 관리자는 로그인 자격 증명을 제공한 사람입니다.

내 외부의 사람이 내 Verified Permissions 리소스에 액세스 AWS 계정 하도록 허용하고 싶습니다.

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 제

어 목록(ACL)을 지원하는 서비스의 경우, 이러한 정책을 사용하여 다른 사람에게 리소스에 대한 액세스 권한을 부여할 수 있습니다.

자세한 내용은 다음을 참조하세요.

- Verified Permissions에서 이러한 기능을 지원하는지 여부를 알아보려면 Amazon Verified Permissions의 작동 방식 IAM을(를) 참조하세요.
- 소유 AWS 계정 한의 리소스에 대한 액세스 권한을 제공하는 방법을 알아보려면 IAM 사용 설명서의 소유한 다른의 IAM 사용자에게 액세스 권한 제공을 참조 AWS 계정 하세요.
- 리소스에 대한 액세스 권한을 타사에 제공하는 방법을 알아보려면 IAM 사용 설명서의 타사 AWS 계정 소유에 대한 액세스 권한 제공을 AWS 계정참조하세요.
- ID 페더레이션을 통해 액세스 권한을 제공하는 방법을 알아보려면 IAM 사용 설명서의 외부에서 인증된 사용자에게 액세스 권한 제공(ID 페더레이션)을 참조하세요.
- 교차 계정 액세스를 위한 역할과 리소스 기반 정책 사용의 차이점을 알아보려면 IAM 사용 설명서의 의 교차 계정 리소스 액세스를 IAM 참조하세요.

Amazon Verified Permissions에 대한 규정 준수 확인

새로운 AWS 가입 환경은 규제 워크로드용으로 설계되지 않았습니다. 새로운 AWS 가입 환경을 사용하고 있지만 규제 대상 워크로드 AWS 에를 사용하려는 경우 AWS 환경에 가입 AWS (고급)하거나 고급 기능을 활성화할 수 있습니다.

AWS 서비스 가 특정 규정 준수 프로그램의 범위 내에 있는지 알아보려면 AWS 서비스 규정 준수 프로그램 제공 범위 내를 참조하고 관심 있는 규정 준수 프로그램을 선택합니다. 일반 정보는 AWS 규정 준수 프로그램.

를 사용하여 타사 감사 보고서를 다운로드할 수 있습니다 AWS Artifact. 자세한 내용은 Downloading Reports inDownloading AWS Artifact을 참조하세요.

사용 시 규정 준수 책임은 데이터의 민감도, 회사의 규정 준수 목표 및 관련 법률과 규정에 따라 AWS 서비스 결정됩니다. 사용 시 규정 준수 책임에 대한 자세한 내용은 AWS 보안 설명서를 AWS 서비스참조하세요.

Amazon Verified Permissions의 복원력

AWS 글로벌 인프라는 AWS 리전 및 가용 영역을 중심으로 구축됩니다.는 지연 시간이 짧고 처리량이 높으며 중복성이 높은 네트워킹과 연결된 물리적으로 분리되고 격리된 여러 가용 영역을 AWS 리전 제

공합니다. 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 스토어를 생성하면 개인 내에 생성 AWS 리전되고 해당 리전의 가용 영역을 구성하는 데이터 센터에 자동으로 복제됩니다. 현재 Verified Permissions는 리전 간 복제는 지원하지 않습니다.

AWS 리전 및 가용 영역에 대한 자세한 내용은 AWS 글로벌 인프라를 참조하세요.

Amazon Verified Permissions API 호출 모니터링

모니터링은 Amazon Verified Permissions 및 기타 AWS 솔루션의 신뢰성, 가용성 및 성능을 유지하는 데 중요한 부분입니다.는 Verified Permissions를 모니터링하고, 이상이 있을 때 이를 보고하고, 필요한 경우 자동 조치를 취할 수 있는 다음 도구를 AWS 제공합니다.

- AWS CloudTrail는 AWS 계정에서 또는 계정을 대신하여 수행한 API 호출 및 관련 이벤트를 캡처하고 지정한 Amazon S3 버킷에 로그 파일을 전송합니다. 호출한 사용자 및 계정 AWS, 호출이 수행된 소스 IP 주소, 호출이 발생한 시기를 식별할 수 있습니다. 자세한 내용은 AWS CloudTrail 사용 설명서를 참조하십시오.

CloudTrail을 사용한 Verified Permissions 모니터링에 대한 자세한 내용은 섹션을 참조하세요를 사용하여 Amazon Verified Permissions API 호출 로깅 AWS CloudTrail.

를 사용하여 Amazon Verified Permissions API 호출 로깅 AWS CloudTrail

Amazon Verified Permissions는 Verified Permissions에서 사용자 AWS CloudTrail, 역할 또는 서비스가 수행한 작업에 대한 레코드를 제공하는 AWS 서비스와 통합됩니다. CloudTrail은 Verified Permissions에 대한 모든 API 호출을 이벤트로 캡처합니다. 캡처되는 호출에는 Verified Permissions 콘솔로부터의 호출과 Verified Permissions API 작업에 대한 코드 호출이 포함됩니다. 추적을 생성하면 Verified Permissions에 대한 이벤트를 포함하여 CloudTrail 이벤트를 Amazon S3 버킷에 지속적으로 배포할 수 있습니다. 추적을 구성하지 않은 경우에도 CloudTrail 콘솔의 이벤트 기록에서 최신 관리 작업 이벤트를 볼 수 있지만와 같은 API 호출에 대한 이벤트는 볼 수 없습니다`isAuthorized`. CloudTrail에서 수집한 정보를 사용하여 Verified Permissions에 수행된 요청, 요청이 수행된 IP 주소, 요청을 수행한 사람, 요청이 수행된 시간 및 추가 세부 정보를 확인할 수 있습니다.

CloudTrail에 대한 자세한 내용은 AWS CloudTrail 사용 설명서를 참조하세요.

CloudTrail의 Verified Permissions 정보

CloudTrail은 계정을 생성할 AWS 계정 때에서 활성화됩니다. Verified Permissions에서 활동이 발생하면 해당 활동이 이벤트 기록의 다른 AWS 서비스 이벤트와 함께 CloudTrail 이벤트에 기록됩니다. AWS 계정에서 최신 이벤트를 확인, 검색 및 다운로드할 수 있습니다. 자세한 내용은 CloudTrail 이벤트 기록을 사용하여 이벤트 보기를 참조하세요.

Verified Permissions에 대한 이벤트를 AWS 계정포함하여에 이벤트를 지속적으로 기록하려면 추적을 생성합니다. 추적을 통해 CloudTrail은 로그 파일을 Amazon S3 버킷에 전달할 수 있습니다. 콘솔에서 트레일을 생성하면 기본적으로 모든 AWS 리전에 트레일이 적용됩니다. 추적은 AWS 파티션의 모든 리전에서 이벤트를 로깅하고 지정한 Amazon S3 버킷으로 로그 파일을 전송합니다. 또한 CloudTrail 로그에서 수집된 이벤트 데이터를 추가로 분석하고 조치를 취하도록 다른 AWS 서비스를 구성할 수 있습니다. 자세한 내용은 다음 자료를 참조하세요.

- 추적 생성 개요
- CloudTrail 지원 서비스 및 통합
- CloudTrail에 대한 Amazon SNS 알림 구성
- 여러 리전에서 CloudTrail 로그 파일 받기 및 여러 계정에서 CloudTrail 로그 파일 받기

모든 Verified Permissions 작업은 CloudTrail에서 로깅되며 Amazon Verified Permissions API 참조 가이드에서 문서화됩니다. 예를 들어 `CreateIdentitySource`, `DeletePolicy` 및 `ListPolicyStores` 작업을 직접적으로 호출하면 CloudTrail 로그 파일에 항목이 생성됩니다.

모든 이벤트 또는 로그 항목에는 요청을 생성했던 사용자에 관한 정보가 포함됩니다. ID 정보를 이용하면 다음을 쉽게 판단할 수 있습니다.

- 요청이 루트 또는 AWS Identity and Access Management (IAM) 사용자 자격 증명으로 이루어졌는지 여부입니다.
- 역할 또는 페더레이션 사용자의 임시 자격 증명을 사용하여 요청이 생성되었는지 여부.
- 요청이 다른 AWS 서비스에서 이루어졌는지 여부입니다.

자세한 내용은 CloudTrail userIdentity 요소를 참조하세요.

추적 또는 이벤트 데이터 스토어를 생성하는 경우 IsAuthorized와 IsAuthorizedWithToken과 같은 데이터 이벤트는 기본적으로 로깅되지 않습니다. CloudTrail 데이터 이벤트를 로깅하려면 활동을 수집할 지원되는 리소스 또는 리소스 유형을 추적에 명시적으로 추가해야 합니다. 자세한 내용을 알아보려면 AWS CloudTrail 사용 설명서의 데이터 이벤트를 참조하세요.

Verified Permissions 로그 파일 항목 이해

추적은 지정한 Amazon S3 버킷에 이벤트를 로그 파일로 전송할 수 있는 구성입니다. CloudTrail 로그 파일에는 하나 이상의 로그 항목이 포함될 수 있습니다. 이벤트는 모든 소스로부터의 단일 요청을 나타내며 요청 작업, 작업 날짜와 시간, 요청 파라미터 등에 대한 정보가 들어 있습니다. CloudTrail 로그 파일은 퍼블릭 API 직접 호출의 주문 스택 트레이스가 아니므로 특정 순서로 표시되지 않습니다.

권한 부여 API 호출의 경우 결정과 같은 응답 요소는 additionalEventData 대신에 포함됩니다responseElements.

주제

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

> **Note**
> 데이터 프라이버시의 예에서 일부 필드가 수정되었습니다.

IsAuthorized

다음 예제에서는 IsAuthorized API 호출에 대한 CloudTrail 로그 항목을 보여줍니다.

```
{
    "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"
}
```

IsAuthorizedWithToken

다음 예제에서는 IsAuthorizedWithToken API 호출에 대한 CloudTrail 로그 항목을 보여줍니다.

```
{
    "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": "IsAuthorizedWithToken",
    "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-with-token",
    "requestParameters": {
        "policyStoreId": "PSEXAMPLEabcdefg111111",
        "resource": {
            "entityType": "PhotoFlash::Photo",
            "entityId": "VacationPhoto94.jpg"
        },
        "action": {
            "actionType": "PhotoFlash::Action",
            "actionId": "ViewPhoto"
        }
    },
    "responseElements": null,
    "additionalEventData": {
        "decision": "ALLOW",
        "principal": {
            "entityType": "PhotoFlash::User",
            "entityId": "us-east-1_EXAMPLE|EXAMPLE_SUBJECT_ID"
        }
```

```
    },
    "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

다음 예제에서는 BatchIsAuthorized API 호출에 대한 CloudTrail 로그 항목을 보여줍니다.

```
{
    "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,
    "resources": [
        {
    "accountId": "123456789012",
    "type": "AWS::VerifiedPermissions::PolicyStore",
    "ARN": "arn:aws:verifiedpermissions::123456789012:policy-store/
PSEXAMPLEabcdefg111111"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": false,
    "recipientAccountId": "123456789012",
    "eventCategory": "Data"
}
```

BatchIsAuthorizedWithToken

다음 예제에서는 BatchIsAuthorizedWithToken API 호출에 대한 CloudTrail 로그 항목을 보여줍니다.


```
{
    "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": "BatchIsAuthorizedWithToken",
    "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.batch-is-authorized-with-
token",
    "requestParameters": {
        "policyStoreId": "PSEXAMPLEabcdefg111111",
        "requests": [
            {
                "action": {
                    "actionType": "PhotoFlash::Action",
                    "actionId": "ViewPhoto"
                },
                "resource": {
                    "entityType": "PhotoFlash::Photo",
                    "entityId": "VacationPhoto94.jpg"
                }
            },
            {
                "action": {
                    "actionType": "PhotoFlash::Action",
                    "actionId": "DeletePhoto"
                },
                "resource": {
                    "entityType": "PhotoFlash::Photo",
                    "entityId": "VacationPhoto94.jpg"
```

```
                }
            }
        ]
    },
    "responseElements": null,
    "additionalEventData": {
        "principal": {
            "entityType": "PhotoFlash::User",
            "entityId": "us-east-1_EXAMPLE|EXAMPLE_SUBJECT_ID"
        },
        "results": [
            {
                "request": {
                    "action": {
                        "actionType": "PhotoFlash::Action",
                        "actionId": "ViewPhoto"
                    },
                    "resource": {
                        "entityType": "PhotoFlash::Photo",
                        "entityId": "VacationPhoto94.jpg"
                    }
                },
                "decision": "ALLOW"
            },
            {
                "request": {
                    "action": {
                        "actionType": "PhotoFlash::Action",
                        "actionId": "DeletePhoto"
                    },
                    "resource": {
                        "entityType": "PhotoFlash::Photo",
                        "entityId": "VacationPhoto94.jpg"
                    }
                },
                "decision": "DENY"
            }
        ]
    },
    "requestID": "c67aa1d9-9fd7-45ad-9a23-e8f9f6dd66c8",
    "eventID": "5ed90624-a4f7-4ef6-a07a-0e6486d30fbe",
    "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"
}
```

CreatePolicyStore

다음 예제에서는 `CreatePolicyStore` API 호출에 대한 CloudTrail 로그 항목을 보여줍니다.

```
{
  "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

다음 예제에서는 `ListPolicyStores` API 호출에 대한 CloudTrail 로그 항목을 보여줍니다.

```
{
  "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

다음 예제에서는 DeletePolicyStore API 호출에 대한 CloudTrail 로그 항목을 보여줍니다.


```
{
  "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

다음 예제에서는 PutSchema API 호출에 대한 CloudTrail 로그 항목을 보여줍니다.

```
{
  "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

다음 예제에서는 GetSchema API 호출에 대한 CloudTrail 로그 항목을 보여줍니다.


```
{
  "eventVersion": "1.08",
  "userIdentity": {
    "type": "AssumedRole",
    "principalId": "EXAMPLE_PRINCIPAL_ID",
    "arn": "arn:aws:iam::222222222222:role/ExampleRole",
    "accountId": "222222222222",
    "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": "222222222222",
      "type": "AWS::VerifiedPermissions::PolicyStore",
      "ARN": "arn:aws:verifiedpermissions::222222222222:policy-store/
PSEXAMPLEabcdefg111111"
    }
  ],
  "eventType": "AwsApiCall",
  "managementEvent": true,
  "recipientAccountId": "222222222222",
  "eventCategory": "Management"
}
```


CreatePolicyTemplate

다음 예제에서는 CreatePolicyTemplate API 호출에 대한 CloudTrail 로그 항목을 보여줍니다.

```
{
  "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

다음 예제에서는 DeletePolicyTemplate API 호출에 대한 CloudTrail 로그 항목을 보여줍니다.


```
{
  "eventVersion": "1.08",
  "userIdentity": {
    "type": "AssumedRole",
    "principalId": "EXAMPLE_PRINCIPAL_ID",
    "arn": "arn:aws:iam::222222222222:role/ExampleRole",
    "accountId": "222222222222",
    "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": "222222222222",
      "type": "AWS::VerifiedPermissions::PolicyStore",
      "ARN": "arn:aws:verifiedpermissions::222222222222:policy-store/
PSEXAMPLEabcdefg111111"
    }
  ],
  "eventType": "AwsApiCall",
  "managementEvent": true,
  "recipientAccountId": "222222222222",
  "eventCategory": "Management"
}
```

CreatePolicy

다음 예제에서는 CreatePolicy API 호출에 대한 CloudTrail 로그 항목을 보여줍니다.


```
{
  "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

다음 예제에서는 GetPolicy API 호출에 대한 CloudTrail 로그 항목을 보여줍니다.

```
{
  "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

다음 예제에서는 CreateIdentitySource API 호출에 대한 CloudTrail 로그 항목을 보여줍니다.

```
{
  "eventVersion": "1.08",
  "userIdentity": {
    "type": "AssumedRole",
    "principalId": "EXAMPLE_PRINCIPAL_ID",
    "arn": "arn:aws:iam::333333333333:role/ExampleRole",
    "accountId": "333333333333",
    "accessKeyId": "AKIAIOSFODNN7EXAMPLE"
  },
  "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

다음 예제에서는 GetIdentitySource API 호출에 대한 CloudTrail 로그 항목을 보여줍니다.

```
{
  "eventVersion": "1.08",
  "userIdentity": {
    "type": "AssumedRole",
    "principalId": "EXAMPLE_PRINCIPAL_ID",
    "arn": "arn:aws:iam::333333333333: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

다음 예제에서는 ListIdentitySources API 호출에 대한 CloudTrail 로그 항목을 보여줍니다.

```
{
  "eventVersion": "1.08",
  "userIdentity": {
    "type": "AssumedRole",
    "principalId": "EXAMPLE_PRINCIPAL_ID",
    "arn": "arn:aws:iam::333333333333: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

다음 예제에서는 DeleteIdentitySource API 호출에 대한 CloudTrail 로그 항목을 보여줍니다.

```
{
  "eventVersion": "1.08",
  "userIdentity": {
    "type": "AssumedRole",
    "principalId": "EXAMPLE_PRINCIPAL_ID",
    "arn": "arn:aws:iam::333333333333:role/ExampleRole",
    "accountId": "333333333333",
    "accessKeyId": "AKIAIOSFODNN7EXAMPLE"
  },
  "eventTime": "2023-05-24T19:55:32Z",
  "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"
}
```

를 사용하여 Amazon Verified Permissions 리소스 생성 AWS CloudFormation

Amazon Verified Permissions는 AWS 리소스 및 인프라를 생성하고 관리하는 데 소요되는 시간을 줄일 수 있도록 리소스를 모델링하고 설정하는 데 도움이 되는 AWS CloudFormation서비스인와 통합됩니다. 원하는 모든 AWS 리소스(예: 정책 스토어)를 설명하는 템플릿을 생성하고 해당 리소스를 CloudFormation 프로비저닝하고 구성합니다.

를 사용하면 템플릿을 재사용하여 Verified Permissions 리소스를 일관되고 반복적으로 설정할 CloudFormation수 있습니다. 리소스를 한 번 설명한 다음 여러 AWS 계정 및 리전에서 동일한 리소스를 반복적으로 프로비저닝합니다.

> **Important**
>
> Amazon Cognito 자격 증명은 Amazon Verified Permissions AWS 리전 와 동일한 모든에서 사용할 수 없습니다. 와 같은 Amazon Cognito 자격 증명과 CloudFormation 관련하여에서 오류가 발생하는 경우 Amazon Cognito 자격 증명을 사용할 수 AWS 리전 있는 지리적으로 가장 가까운에 Amazon Cognito 사용자 풀 및 클라이언트를 생성하는 `Unrecognized resource types: AWS::Cognito::UserPool, AWS::Cognito::UserPoolClient`것이 좋습니다. Verified Permissions 자격 증명 소스를 생성할 때는 새로 생성된 이 사용자 풀을 사용하십시오.

확인된 권한 및 CloudFormation 템플릿

Verified Permissions 및 관련 서비스에 대한 리소스를 프로비저닝하고 구성하려면 CloudFormation 템플릿을 이해해야 합니다. 템플릿은 JSON 또는 YAML로 서식 지정된 텍스트 파일입니다. 이러한 템플릿은 CloudFormation 스택에서 프로비저닝하려는 리소스를 설명합니다. JSON 또는 YAML에 익숙하지 않은 경우 CloudFormation Designer를 사용하여 CloudFormation 템플릿을 시작할 수 있습니다. 자세한 내용은 AWS CloudFormation 사용 설명서의 CloudFormation Designer란 무엇입니까?를 참조하세요.

Verified Permissions는에서 자격 증명 소스, 정책, 정책 스토어, 정책 템플릿 및 정책 스토어 별칭 생성을 지원합니다 CloudFormation. Verified Permissions 리소스에 대한 JSON 및 YAML 템플릿 예제를 비롯한 자세한 내용은 AWS CloudFormation 사용 설명서의 Amazon Verified Permissions 리소스 유형 참조를 참조하세요.

AWS CDK 구문

AWS Cloud Development Kit (AWS CDK) 는 코드에서 클라우드 인프라를 정의하고 이를 프로비저닝하기 위한 오픈 소스 소프트웨어 개발 프레임워크입니다 CloudFormation. 구성 요소 또는 재사용 가능한 클라우드 구성 요소를 사용하여 CloudFormation 템플릿을 생성할 수 있습니다. 그런 다음 이러한 템플릿을 사용하여 클라우드 인프라를 배포할 수 있습니다.

자세한 내용을 알아보고 AWS CDK를 다운로드하려면 AWS 클라우드 개발 키트를 참조하세요.

다음은 구문과 같은 Verified Permissions AWS CDK 리소스에 대한 설명서 링크입니다.

- Amazon Verified Permissions L2 CDK 구성

에 대해 자세히 알아보기 CloudFormation

에 대해 자세히 알아보려면 다음 리소스를 CloudFormation참조하세요.

- AWS CloudFormation
- AWS CloudFormation 사용 설명서
- CloudFormation API Reference
- AWS CloudFormation 명령줄 인터페이스 사용 설명서

를 사용하여 Amazon Verified Permissions에 액세스 AWS PrivateLink

AWS PrivateLink 를 사용하여 VPC와 Amazon Verified Permissions 간에 프라이빗 연결을 생성할 수 있습니다. 인터넷 게이트웨이, NAT 디바이스, VPN 연결 또는 Direct Connect 연결을 사용하지 않고 VPC에 있는 것처럼 Verified Permissions에 액세스할 수 있습니다. VPC의 인스턴스에서 Verified Permissions에 액세스하는 데 퍼블릭 IP 주소가 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 안내서의 AWS PrivateLink를 통해 AWS 서비스 에 액세스를 참조하세요.

Verified Permissions에 대한 고려 사항

Verified Permissions에 대한 인터페이스 엔드포인트를 설정하려면 먼저 AWS PrivateLink 가이드의 고려 사항을 검토합니다.

Verified Permissions에서는 인터페이스 엔드포인트를 통해 모든 API 작업에 대한 호출 수행을 지원합니다.

VPC 엔드포인트 정책은 Verified Permissions에 대해 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 Verified Permissions에 대한 인터페이스 엔드포인트를 생성할 수 있습니다. 자세한 내용은 AWS PrivateLink 안내서의 인터페이스 엔드포인트 생성을 참조하세요.

다음 서비스 이름을 사용하여 Verified Permissions에 대한 인터페이스 엔드포인트를 생성합니다.

```
com.amazonaws.region.verifiedpermissions
```

인터페이스 엔드포인트에 프라이빗 DNS를 사용하도록 설정하는 경우, 리전에 대한 기본 DNS 이름을 사용하여 Verified Permissions에 API 요청을 할 수 있습니다. 예를 들어 `verifiedpermissions.us-east-1.amazonaws.com`입니다.

엔드포인트의 엔드포인트 정책 생성

엔드포인트 정책은 인터페이스 엔드포인트에 연결할 수 있는 IAM 리소스입니다. 기본 엔드포인트 정책은 인터페이스 엔드포인트를 통해 Verified Permissions에 대한 전체 액세스를 허용합니다. VPC에서 Verified Permissions에 허용되는 액세스를 제어하려면 인터페이스 엔드포인트에 사용자 지정 엔드포인트 정책을 연결합니다.

엔드포인트 정책은 다음 정보를 지정합니다.

- 작업을 수행할 수 있는 위탁자(AWS 계정, IAM 사용자, IAM 역할)
- 수행할 수 있는 작업
- 작업을 수행할 수 있는 리소스.

자세한 내용은 AWS PrivateLink 안내서의 엔드포인트 정책을 사용하여 서비스에 대한 액세스 제어를 참조하세요.

예: Verified Permissions 작업에 대한 VPC 엔드포인트 정책

다음은 사용자 지정 엔드포인트 정책의 예입니다. 이 정책을 인터페이스 엔드포인트에 연결하면 모든 리소스의 모든 보안 주체에 대해 나열된 Verified Permissions 작업에 대한 액세스 권한을 부여합니다.

```
{
   "Statement": [
      {
         "Principal": "*",
         "Effect": "Allow",
         "Action": [
            "verifiedpermissions:IsAuthorized",
            "verifiedpermissions:IsAuthorizedWithToken",
            "verifiedpermissions:GetPolicy"
         ],
         "Resource":"*"
      }
```

```
    ]
}
```

Amazon Verified Permissions 할당량

AWS 계정 에는 각 AWS 서비스에 대해 이전에 제한이라고 하는 기본 할당량이 있습니다. 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 Service Quotas 콘솔을 엽니다. 탐색 창에서 AWS 서비스를 선택하고 Verified Permissions를 선택합니다.

할당량 증가를 요청하려면 Service Quotas 사용 설명서의 할당량 증가 요청을 참조하세요. Service Quotas에서 아직 할당량을 사용할 수 없는 경우 한도 증가 양식을 사용합니다.

AWS 계정 에는 Verified Permissions와 관련된 다음과 같은 할당량이 있습니다.

주제

- 리소스 할당량
- 계층 구조 할당량
- 초당 작업 할당량

리소스 할당량

이름	기본값	조정 가능	설명
계정별 리전별 정책 스토어	지원되는 각 리전: 30,000	예	정책 스토어의 최대 수입니다.
정책 스토어당 정책 템플릿	각 지원되는 리전: 40	예	정책 스토어의 최대 정책 템플릿 수입니다.
정책 스토어당 자격 증명 소스	1	아니요	정책 스토어에 대해 정의할 수 있는 최대 자격 증명 소스 수입니다.

이름	기본값	조정 가능	설명
정책 스토어당 정책 스토어 별칭	10	예	단일 정책 스토어와 연결할 수 있는 최대 정책 스토어 별칭 수입니다.
권한 부여 요청 크기[1]	1MB	아니요	권한 부여 요청의 최대 크기입니다.
정책 크기	10,000바이트	예	개별 정책의 최대 크기입니다.
스키마 크기	100,000바이트	예	정책 스토어 스키마의 최대 크기입니다.
정책 스토어 스키마당 네임스페이스	100	예	정책 스토어 스키마에서 정의할 수 있는 최대 네임스페이스 수입니다.
리소스당 정책 크기	1MB2	예	특정 리소스를 참조하는 모든 정책의 최대 크기입니다.
리소스 보안 주체 조합당 정책 크기	200,000바이트2	아니요	보안 주체와 리소스의 특정 조합을 참조하는 모든 정책의 최대 크기입니다.

[1] 권한 부여 요청의 할당량은 IsAuthorized와 IsAuthorizedWithToken에 대해 동일합니다.

2 보안 주체와 리소스의 고유한 조합에 대한 모든 정책의 총 크기는 200,000바이트를 초과할 수 없습니다. 지정되지 않은 보안 주체 또는 리소스는 고유한 값으로 계산됩니다. 예를 들어 제한은 리소스가 지정되지 않은 특정 보안 주체 또는 보안 주체가 지정되지 않은 특정 리소스로 범위가 지정된 모든 정책의 총 크기에 적용됩니다. 이 제한은 조정할 수 없습니다. 또한 고유한 리소스(또는 지정되지 않은 리소스)에 대한 모든 정책의 총 크기는 1MB를 초과해서는 안 됩니다.이 제한은 계층 할당량에 대한 추가

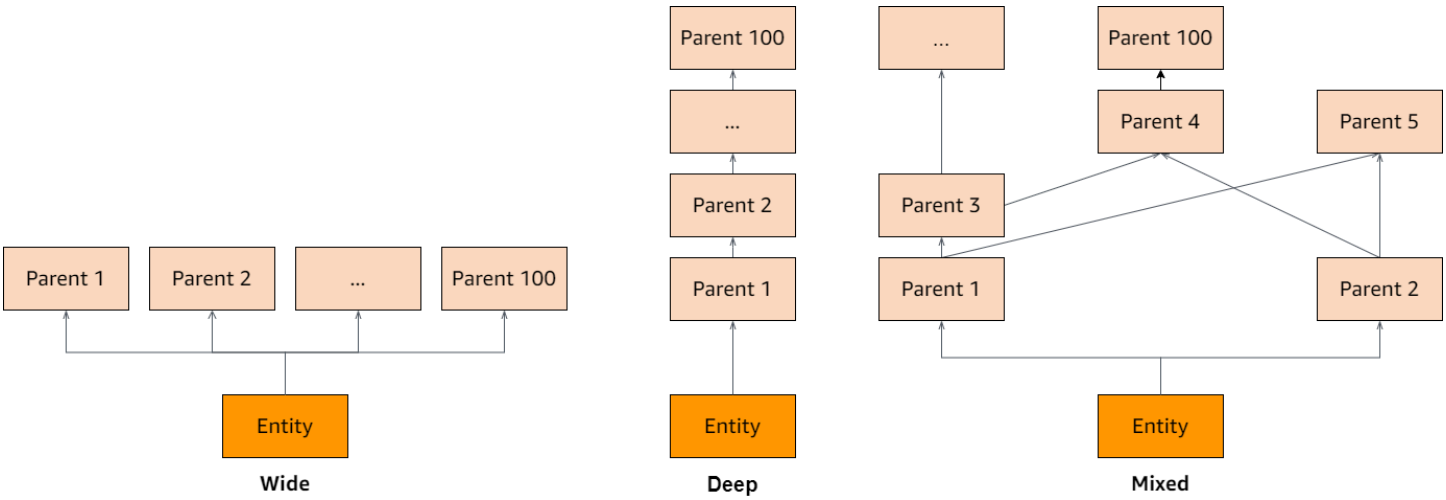
제약 조건으로 조정할 수 있습니다. Verified Permissions가이 할당량을 계산하는 방법, 예제 및 이를 유지하기 위한 전략에 대한 자세한 내용은 섹션을 참조하세요리소스당 정책 크기.

계층 구조 할당량

> **Note**
>
> 각 전이적 상위 할당량은 각 개체에 개별적으로 적용되며 직접 및 간접(전이적) 상위 할당량을 모두 합계에 가산합니다. 예를 들어 보안 주체당 전이적 상위 항목의 한도가 100인 경우 보안 주체는 중첩 그룹이 없는 100개의 직접 상위 그룹을 가질 수 있습니다. 또는 보안 주체는 각각 9개의 추가 상위 그룹에 속하는 10개의 직접 상위 그룹을 가질 수 있습니다. 해당 보안 주체에 대해 총 100개의 상위 항목으로 구성된 모든 조합이 유효합니다.

이름	기본값	조정 가능	설명
보안 주체당 전이적 상위 엔터티	100	아니요	각 보안 주체에 대한 최대 전이적 상위 엔터티 수입니다.
작업당 전이적 상위 엔터티	100	아니요	각 작업에 대한 최대 전이적 상위 엔터티 수입니다.
리소스당 전이적 상위 엔터티	100	아니요	각 리소스에 대한 최대 전이적 상위 엔터티 수입니다.

아래 다이어그램은 엔티티(보안 주체, 작업 또는 리소스)에 대해 전이적 상위 엔터티를 정의하는 방법을 보여줍니다.



초당 작업 할당량

Verified Permissions는 애플리케이션 요청이 API 작업에 대한 할당량을 초과할 AWS 리전 때의 서비스 엔드포인트에 대한 요청을 제한합니다. Verified Permissions는 초당 요청의 할당량을 초과하거나 동시 쓰기 작업을 시도할 때 예외를 반환할 수 있습니다. Service Quotas에서 현재 RPS 할당량을 볼 수 있습니다. 애플리케이션이 작업 할당량을 초과하지 않도록 하려면 재시도 및 지수 백오프에 맞게 최적화해야 합니다. 자세한 내용은 백오프 패턴으로 재시도 및 워크로드에서 API 제한 관리 및 모니터링을 참조하세요.

이름	기본값	조정 가능	설명
정책 스토어별 리전별 초당 BatchGetPolicy 요청	지원되는 각 리전: 10개	예	정책 스토어당 초당 최대 BatchGetPolicy 요청 수입니다.
BatchIsAuthorized 요청	지원되는 각 지역: 30개	예	정책 스토어당 초당 최대 BatchIsAuthorized 요청 수입니다.
정책 스토어별 리전별 초당 BatchIsAuthorizedWithToken 요청	지원되는 각 지역: 30개	예	정책 스토어당 초당 최대 BatchIsAuthorizedWithToken 요청 수입니다.

이름	기본값	조정 가능	설명
정책 스토어별 리전별 초당 CreateIdentitySource 요청	각 지원되는 지역: 1	예	정책 스토어당 초당 최대 CreateIdentitySource 요청 수입니다.
정책 스토어별 리전별 초당 CreatePolicy 요청	지원되는 각 리전: 10개	예	정책 스토어당 초당 최대 CreatePolicy 요청 수입니다.
계정별 리전별 초당 CreatePolicyStore 요청	지원되는 각 리전: 1	아니요	초당 최대 CreatePolicyStore 요청 수입니다.
정책 스토어별 리전별 초당 CreatePolicyTemplate 요청	지원되는 각 리전: 10개	예	정책 스토어당 초당 최대 CreatePolicyTemplate 요청 수입니다.
정책 스토어별 리전별 초당 DeleteIdentitySource 요청	각 지원되는 지역: 1	예	정책 스토어당 초당 최대 DeleteIdentitySource 요청 수입니다.
정책 스토어별 리전별 초당 DeletePolicy 요청	지원되는 각 리전: 10개	예	정책 스토어당 초당 최대 DeletePolicy 요청 수입니다.
계정별 리전별 초당 DeletePolicyStore 요청	지원되는 각 리전: 1	아니요	초당 최대 DeletePolicyStore 요청 수입니다.
정책 스토어별 리전별 초당 DeletePolicyTemplate 요청	지원되는 각 리전: 10개	예	정책 스토어당 초당 최대 DeletePolicyTemplate 요청 수입니다.

이름	기본값	조정 가능	설명
정책 스토어별 리전별 초당 GetIdentitySource 요청	지원되는 각 리전: 10개	예	정책 스토어당 초당 최대 GetIdentitySource 요청 수입니다.
정책 스토어별 리전별 초당 GetPolicy 요청	지원되는 각 리전: 10개	예	정책 스토어당 초당 최대 GetPolicy 요청 수입니다.
계정별 리전별 초당 GetPolicyStore 요청	지원되는 각 리전: 10개	예	초당 최대 GetPolicyStore 요청 수입니다.
정책 스토어별 리전별 초당 GetPolicyTemplate 요청	지원되는 각 리전: 10개	예	정책 스토어당 초당 최대 GetPolicyTemplate 요청 수입니다.
정책 스토어별 리전별 초당 GetSchema 요청	지원되는 각 리전: 10개	예	정책 스토어당 초당 최대 GetSchema 요청 수입니다.
정책 스토어별 리전별 초당 IsAuthorized 요청	지원되는 각 리전: 200	예	정책 스토어당 초당 최대 IsAuthorized 요청 수입니다.
정책 스토어별 리전별 초당 IsAuthorizedWithToken 요청	지원되는 각 리전: 200	예	정책 스토어당 초당 최대 IsAuthorizedWithToken 요청 수입니다.
ListIdentitySources 요청 수	지원되는 각 리전: 10개	예	정책 스토어당 초당 최대 ListIdentitySources 요청 수입니다.
ListPolicies 정책 스토어별 리전별 초당 요청 수	지원되는 각 리전: 10개	예	정책 스토어당 초당 최대 ListPolicies 요청 수입니다.

이름	기본값	조정 가능	설명
계정별 리전별 초당 ListPolicyStores 요청	지원되는 각 리전: 10개	예	초당 ListPolicyStores 요청의 최대 수
ListPolicyTemplates 정책 스토어별 리전별 초당 요청 수	지원되는 각 리전: 10개	예	정책 스토어당 초당 최대 ListPolicyTemplates 요청 수입니다.
정책 스토어별 리전별 초당 PutSchema 요청	지원되는 각 리전: 10개	예	정책 스토어당 초당 최대 PutSchema 요청 수입니다.
정책 스토어별 리전별 초당 UpdateIdentitySource 요청	각 지원되는 지역: 1	예	정책 스토어당 초당 최대 UpdateIdentitySource 요청 수입니다.
정책 스토어별 리전별 초당 UpdatePolicy 요청	지원되는 각 리전: 10개	예	정책 저장소당 초당 최대 UpdatePolicy 요청 수입니다.
계정별 리전별 초당 UpdatePolicyStore 요청	지원되는 각 리전: 10	아니요	초당 최대 UpdatePolicyStore 요청 수입니다.
정책 스토어별 리전별 초당 UpdatePolicyTemplate 요청	지원되는 각 리전: 10개	예	정책 스토어당 초당 최대 UpdatePolicyTemplate 요청 수입니다.

Amazon Verified Permissions 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
- 정책 스토어 별칭
- 정책 이름
- 정책 템플릿 이름
- 충족 대상 정책
- Amazon Verified Permissions와 Cedar 정책 언어의 차이점

Cedar 정책 언어 개념

- 권한 부여
- 엔터티
- 그룹 및 계층
- 네임스페이스
- 정책
- 정책 템플릿
- 스키마

권한 부여 모델

권한 부여 모델은 애플리케이션이 수행하는 권한 부여 요청의 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 보안 주체 그룹을 사용하여 RBAC 결정을 지원할 수 있습니다.

ABAC(속성 기반 액세스 제어)는 자격 증명과 연결되는 권한을 해당 자격 증명의 속성에 따라 결정하는 평가 기준입니다. Cedar는 보안 주체의 속성을 참조하는 정책 조건을 사용하여 ABAC 결정을 지원할 수 있습니다.

Cedar 정책 언어를 사용하면 속성 기반 조건을 가진 사용자 그룹에 대해 권한을 정의할 수 있으므로 RBAC와 ABAC를 단일 정책으로 결합할 수 있습니다.

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 권한 부여 요청에 대한 응답입니다. 여기에는 허용 또는 거부 결정과 함께 결정적 정책의 ID와 같은 추가 정보가 포함됩니다.

고려 대상 정책

고려 대상 정책은 인증 요청을 평가할 때 Verified Permissions이 포함하도록 선택한 전체 정책 집합입니다.

컨텍스트 데이터

컨텍스트 데이터는 평가할 추가 정보를 제공하는 속성 값입니다.

결정적 정책

결정적 정책은 권한 부여 응답을 결정하는 정책입니다. 예를 들어 충족된 정책이 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 여부를 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 정책으로 표현될 수 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 요청을 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.

정책 스토어 별칭

정책 스토어 별칭은 정책 스토어의 친숙한 이름입니다. 정책 스토어 별칭을 사용하여 `policyStoreId` 파라미터를 수락하는 Verified Permissions 작업에서 정책 스토어를 식별할 수 있습니다. 정책 스토어 별칭은 자체 ARN이 있는 독립적인 AWS 리소스입니다. ARNs 각 별칭은 한 번에 하나의 정책 저장소와 연결되며 여러 별칭을 동일한 정책 저장소와 연결할 수 있습니다. 자세한 내용은 Amazon Verified Permissions 정책 스토어 별칭 단원을 참조하십시오.

정책 이름

정책 이름은 정책의 선택적인 표시 이름입니다. 정책 이름은 정책 스토어 내의 모든 정책에 대해 고유해야 하며 접두사가 붙어야 합니다`name/`. `policyId` 파라미터를 수락하는 컨트롤 플레인 작업에서 정책 ID 대신 정책 이름을 사용할 수 있습니다. 정책을 생성하거나 업데이트할 때 이름을 설정할 수 있습니다. `GetPolicy` 및 만 출력에 이름을 `ListPolicies` 반환합니다.

정책 템플릿 이름

정책 템플릿 이름은 정책 템플릿의 선택적인 표시 이름입니다. 정책 템플릿 이름은 정책 스토어 내의 모든 정책 템플릿에 대해 고유해야 하며 접두사가 붙어야 합니다`name/`. `policyTemplateId` 파라미터를 수락하는 컨트롤 플레인 작업에서 정책 템플릿 ID 대신 정책 템플릿 이름을 사용할 수 있습니다. 정책 템플릿을 생성하거나 업데이트할 때 이름을 설정할 수 있습니다. `GetPolicyTemplate` 및 만 출력에 이름을 `ListPolicyTemplates` 반환합니다.

충족 대상 정책

충족 대상 정책은 권한 부여 요청의 매개변수와 일치하는 정책입니다.

Amazon Verified Permissions와 Cedar 정책 언어의 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.

> **Note**
>
> 2026년 5월부터 Verified Permissions는 Cedar와 연계되었으며 이제 여러 네임스페이스를 지원합니다.

정책 템플릿 지원

Verified Permissions과 Cedar는 모두 범위에서 `principal` 및 `resource`에 대한 자리 표시자만 허용합니다. 하지만 Verified Permissions를 사용하려면 `principal` 및 `resource`에 제약이 있어야 합니다.

다음 정책은 Cedar에서 유효하지만 `principal`에 제약이 없기 때문에 Verified Permissions에서는 거부됩니다.

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

다음 두 예제에서는 `principal` 및 `resource`에 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

Verified Permissions를 사용하려면 모든 스키마 JSON 키 이름이 비어 있지 않은 문자열이어야 합니다. Cedar는 속성 또는 네임스페이스와 같은 몇 가지 경우에 빈 문자열을 허용합니다.

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 평가 요청의 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 정의했더라도 권한 부여 API 작업에 전달해야 하는 엔터티 목록의 일부로 작업 그룹 목록도 포함해야 합니다.

Verified Permissions는 이 작업을 사용자 대신 수행합니다. 스키마에서 정의한 모든 작업 그룹은 `IsAuthorized` 또는 `IsAuthorizedWithToken` 작업에 매개 변수로 전달하는 엔터티 목록에 자동으로 추가됩니다.

엔터티 형식

`entityList` 파라미터를 사용하는 Verified Permissions 엔터티의 JSON 형식은 다음과 같은 점에서 Cedar와 다릅니다.

- Verified Permissions에서 JSON 객체는 모든 키-값 쌍을 이름이 `Record`인 JSON 객체로 래핑해야 합니다.
- Verified Permissions의 JSON 목록은 키 이름이 `Set`이고 값이 Cedar의 원래 JSON 목록인 JSON 키-값 쌍으로 래핑되어야 합니다.
- `String`, `Long`, 및 `Boolean` 유형 이름의 경우 Cedar의 각 키-값 쌍은 Verified Permissions의 JSON 객체로 대체됩니다. 객체 이름은 원래 키 이름입니다. JSON 객체 내에는 키-값 쌍이 하나 있는데, 여기서 키 이름은 스칼라 값(`String`, `Long`, 또는 `Boolean`)의 유형 이름이고 값은 Cedar 엔터티의 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 예제에서는 개체 목록이 각각 Cedar 및 Verified Permissions로 표현되는 방법을 보여줍니다.

Cedar

```
[
  {
    "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 예제에서는 Cedar 및 Verified Permissions의 권한 부여 요청에서 정책을 평가하기 위해 개체의 형식이 어떻게 지정되는지 보여줍니다.

Cedar

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

Verified Permissions

```
[
    {
        "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"
            }
        ]
    },
    {
        "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": []
    }
```

```
]
```

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(바이트 단위)	Cedar 제한
정책 크기[1]	10,000	없음
인라인 정책 설명	150	Cedar에는 해당 사항 없음
정책 템플릿 크기	10,000	없음
스키마 크기	100,000건	없음
엔터티 유형	200	없음
정책 ID	64	없음
정책 템플릿 ID	64	없음
엔터티 ID	200	없음
정책 스토어 ID	64	Cedar에는 해당 사항 없음

[1] 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 200,000바이트를 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.

Cedar 4로 Amazon Verified Permissions 업그레이드 FAQ

Amazon Verified Permissions는 사용하는 Cedar 버전을 버전 2에서 버전 4로 업그레이드하고 있습니다. Cedar는 정책 스토어에서 정책, 정책 템플릿 및 스키마를 작성하는 데 사용하는 오픈 소스 언어입니다. Verified Permissions의 Cedar 4 지원을 통해 `is` 연산자 및 엔터티 태그와 같은 새로운 기능을 사용하여 보다 표현적인 정책을 작성할 수 있습니다.

Amazon Verified Permissions는 정책 스토어를 Cedar 4로 자동 업그레이드합니다. 그러나 Cedar 2에 대해 작성된 일부 정책, 스키마 및 권한 부여 요청은 Cedar 4와 호환되지 않습니다. 정책 스토어의 경우 자동으로 업그레이드되지 않습니다. Cedar 4로 업그레이드하기 전에 정책, 정책 템플릿, 스키마 또는 애플리케이션 코드를 변경해야 할 수 있습니다.

주제

- Amazon Verified Permissions를 호출할 때 Cedar 2가 더 이상 지원되지 않는다는 오류가 표시되는 이유는 무엇입니까?
- 일부 정책, 정책 템플릿 및 스키마가 Cedar 4와 호환되지 않는 이유는 무엇입니까?
- 정책 스토어가 Cedar 2를 사용하는지 아니면 Cedar 4를 사용하는지 어떻게 알 수 있나요?
- Cedar 4로 업그레이드하려면 어떻게 해야 하나요?
- 정책 스토어를 Cedar 4에서 Cedar 2로 다운그레이드할 수 있나요?
- 정책 스토어가 Cedar 2에 대해 구성되었다는 오류 메시지가 표시되는 이유는 무엇입니까?
- 스키마를 Cedar 4와 호환되도록 하려면 어떻게 해야 하나요?
- 정책 및 템플릿을 Cedar 4와 호환되도록 하려면 어떻게 해야 하나요?

Amazon Verified Permissions를 호출할 때 Cedar 2가 더 이상 지원되지 않는다는 오류가 표시되는 이유는 무엇입니까?

2026년 4월부터 Cedar `IsAuthorized2`를 사용하는 정책 스토어에 대해 권한 부여 APIs(`BatchIsAuthorized`, `IsAuthorizedWithToken` 및 `BatchIsAuthorizedWithToken`)가 비활성화됩니다. 이 경우 다음 오류 메시지가 반환됩니다.

```
Your policy store uses Cedar 2.x, which is no longer supported. See https://
docs.aws.amazon.com/verifiedpermissions/latest/userguide/cedar4-faq.html or contact AWS
 Support for more information.
```

Verified Permissions를 호출할 때이 오류 메시지가 표시되면 정책 스토어가 Cedar 4와 호환되지 않아가 새 버전으로 마이그레이션할 수 없음을 의미합니다. Amazon Verified Permissions를 계속 사용하려면 Cedar 4를 사용할 새 정책 스토어를 생성하거나에 문의하세요 지원.

일부 정책, 정책 템플릿 및 스키마가 Cedar 4와 호환되지 않는 이유는 무엇입니까?

Cedar 팀은 Cedar 2 이후 몇 가지 이전 버전과 호환되지 않는 변경 사항을 적용하여 버그를 수정하고 언어를 간소화했습니다. 이러한 변경 사항은 다음과 같습니다.

- 정책, 정책 템플릿 및 스키마에 대한 구문 변경
- 더 많은 오류를 감지하는 더 정확한 정책 검사기
- 와 같은 내장 함수의 동작 변경 `isInRange`

이전 버전과 호환되지 않는 변경 사항의 전체 목록을 보려면 Cedar 변경 로그(*)에서 로 표시된 항목을 찾습니다.

정책 스토어가 Cedar 2를 사용하는지 아니면 Cedar 4를 사용하는지 어떻게 알 수 있나요?

Amazon Verified Permissions 콘솔을 사용하거나 GetPolicyStore 작업을 사용하여 정책 스토어에서 사용하는 Cedar 버전을 확인할 수 있습니다.

> **Note**
>
> 동일한 AWS 계정 및 리전의 모든 정책 스토어는 동일한 버전의 Cedar를 사용합니다.

Console

정책 스토어의 Cedar 버전을 확인하려면(콘솔)

1. 에 로그인 AWS Management Console 하고 https://console.aws.amazon.com/verifiedpermissions/ Amazon Verified Permissions 콘솔을 엽니다.
2. 탐색 창에서 정책 저장소를 선택한 다음 확인할 정책 저장소를 선택합니다.
3. 탐색 창에서 설정을 선택합니다.

4. 세부 정보 상자에서 Cedar 버전 필드를 찾습니다.

필드는 정책 스토어가 Cedar 2를 사용하는 CEDAR_2 경우를 읽고 Cedar 4를 사용하는 CEDAR_4 경우를 읽습니다.

CLI

정책 스토어의 Cedar 버전을 확인하려면(AWS CLI)

1. 아직 설치하지 않은 경우 AWS Command Line Interface (AWS CLI)를 설치하고 구성합니다. 자세한 내용은 최신 버전의 AWS CLI설치 또는 업데이트를 참조하세요.
2. get-policy-store 명령을 사용합니다. 다음 예에서 *policy-store-id*를 정책 스토어의 식별자로 바꿉니다.

```
aws verifiedpermissions get-policy-store \
  --policy-store-id policy-store-id
```

출력의 cedarVersion 필드에는 정책 스토어에서 사용 중인 Cedar 버전이 표시됩니다. 예제:

```
{
    "policyStoreId": "ABCDEFG12345678abcdefg",
    "arn": "arn:aws:verifiedpermissions::111122223333:policy-store/
ABCDEFG12345678abcdefg",
    "validationSettings": {
        "mode": "STRICT"
    },
    "createdDate": "2025-06-03T13:09:47.752255+00:00",
    "lastUpdatedDate": "2025-06-03T13:09:47.752255+00:00",
    "deletionProtection": "ENABLED",
    "cedarVersion": "CEDAR_2"
}
```

필드는 정책 스토어가 Cedar 2를 사용하는 CEDAR_2 경우를 읽고 Cedar 4를 사용하는 CEDAR_4 경우를 읽습니다.

Cedar 4로 업그레이드하려면 어떻게 해야 하나요?

Amazon Verified Permissions는 이미 대부분의 고객을 Cedar 4로 업그레이드했습니다. 정책 스토어를 생성한 적이 없는 경우 새로 생성한 모든 정책 스토어는 Cedar 4를 사용합니다. 기존 고객인 경우 이미

Cedar 4로 업그레이드했을 것입니다. 정책 스토어에서 사용하는 Cedar 버전을 정책 스토어가 Cedar 2를 사용하는지 아니면 Cedar 4를 사용하는지 어떻게 알 수 있나요? 확인하려면 섹션을 참조하세요.

업그레이드하지 않은 경우 Verified Permissions는 Cedar 4와 호환되지 않는 정책 스토어 중 하나에서 정책, 정책 템플릿, 스키마 또는 권한 부여 요청을 감지했습니다. 2025년 후반에 호환되지 않는 리소스가 무엇인지 설명하는 이메일 알림을 보내드리겠습니다. 더 빨리 업그레이드하려면를 사용하여 사례를 엽니다 지원.

> **Important**
>
> 동일한의 모든 정책 스토어는 동일한 버전의 Cedar를 AWS 계정 사용합니다. 계정의 정책 스토어 하나가 Cedar 4와 호환되지 않는 경우 해당 계정의 정책 스토어에서 Cedar 4를 사용할 수 없습니다.

정책 스토어를 Cedar 4에서 Cedar 2로 다운그레이드할 수 있나요?

아니요. 정책 스토어를 Cedar 4로 업그레이드한 후 문제가 발생하면에서 사례를 엽니다 지원.

정책 스토어가 Cedar 2에 대해 구성되었다는 오류 메시지가 표시되는 이유는 무엇입니까?

Amazon Verified Permissions의 일부 기능은 Cedar 4의 새로운 기능에 의존합니다. 정책 스토어에서 Cedar 4를 사용하지 않는 경우 다음 API 필드를 사용할 수 없습니다.

- IsAuthorized, BatchIsAuthorized IsAuthorizedWithToken 및 BatchIsAuthorizedWithToken 작업에서:
  - `datetimedecimal` 또는 `context` 필드의 `attributes` 또는 `duration` 값

정책 스토어가 업그레이드될 때까지 Cedar 2 이후에 도입된 정책, 정책 템플릿 또는 스키마에서 구문 또는 데이터 유형을 사용할 수 없습니다.

스키마를 Cedar 4와 호환되도록 하려면 어떻게 해야 하나요?

Verified Permissions 콘솔은 스키마의 일부 호환성 문제를 자동으로 해결할 수 있습니다. 스키마를 자동으로 수정할 수 없는 경우 콘솔에 수동으로 수정할 수 있는 오류 목록이 표시됩니다.

> **Important**
>
> Amazon Verified Permissions 콘솔의 코드 편집기는 정책 스토어가 Cedar 2를 사용하더라도 항상 Cedar 4의 오류 및 경고를 표시합니다. 변경 사항 저장 버튼 또는 Verified Permissions API를 사용하여 Cedar 4와 호환되지 않는 스키마를 계속 업데이트할 수 있습니다.

콘솔을 사용하여 스키마를 수정하려면

1. 에 로그인 AWS Management Console 하고 확인된 권한으로 Amazon Verified Permissions 콘솔을 엽니다.
2. 탐색 창에서 정책 저장소를 선택한 다음 확인할 정책 저장소를 선택합니다.
3. 탐색 창에서 스키마를 선택합니다.
4. 스키마를 자동으로 수정할 수 있는 경우 'Fix'를 클릭하여 호환되는 버전을 미리 봅니다'라는 배너가 표시됩니다. 수정을 선택합니다.
5. 스키마에 대한 변경 사항을 검토하고 업데이트된 스키마 미리 보기를 클릭합니다.
6. 업데이트된 스키마를 검토하고 변경 사항 저장을 클릭합니다.

스키마를 자동으로 수정할 수 없는 경우 콘솔에서 직접 수정할 수 있는 오류 목록을 볼 수 있습니다.

1. 위에서 설명한 대로 스키마 편집 페이지를 엽니다.
2. JSON 모드를 선택합니다.
3. 코드 편집기 왼쪽에 있는 거터의 빨간색 오류 아이콘 위에 마우스를 올려 놓습니다. 오류 메시지가 툴팁에 표시됩니다.

발생할 수 있는 몇 가지 일반적인 오류와 이를 해결하는 방법은 다음과 같습니다.

JSON에서 스키마 구문 분석 실패: `*field-name*`

Cedar 2를 사용하면 Cedar 스키마의 일부로 의미가 없더라도 유형 정의와 같은 스키마의 일부에 임의의 필드를 포함할 수 있습니다. Cedar 4에서는 더 이상 허용되지 않습니다. 이 오류를 해결하려면 JSON 스키마에서 *field-name*이라는 필드를 제거합니다. 유효한 스키마 필드 목록은 Cedar 설명서를 참조하세요.

알 수 없는 확장 유형 `*extension-name*`

Cedar 2에서 type가 인 속성을 선언Extension할 때 값이 유효한 확장 유형 이름인지 여부에 관계없이 name 필드에 값을 지정할 수 있습니다. 이제 Cedar 4에서 오류가 발생했습니다. 문제를 해결하려면 *extension-name*을 유효한 확장 유형 이름으로 바꿉니다. Cedar 설명서에서 유효한 확장 유형 이름 목록을 찾을 수 있습니다.

스키마의 오류를 해결하는 방법을 잘 모르는 경우에 문의하세요. 지원

정책 및 템플릿을 Cedar 4와 호환되도록 하려면 어떻게 해야 하나요?

Verified Permissions 콘솔에는 정책 또는 템플릿에서 Cedar 4와 호환되지 않는 오류가 표시됩니다.

콘솔에서 정책 또는 템플릿의 오류를 보려면

1. 에 로그인 AWS Management Console 하고 확인된 권한으로 Amazon Verified Permissions 콘솔을 엽니다.
2. 탐색 창에서 정책 저장소를 선택한 다음 확인할 정책 저장소를 선택합니다.
3. 필요에 따라 탐색 창에서 정책 또는 정책 템플릿을 선택합니다.
4. 호환되지 않는 정책 또는 템플릿을 선택합니다.
5. 편집을 선택합니다.
6. 코드 편집기 왼쪽에 있는 거터의 빨간색 오류 아이콘 위에 마우스를 올려 놓습니다. 오류 메시지가 툴팁에 표시됩니다.

발생할 수 있는 몇 가지 일반적인 오류와 이를 해결하는 방법은 다음과 같습니다.

정책에서 빈 세트 리터럴이 금지됨

Cedar 2에서는 구문을 사용하여 집합이 비어 있는지 `mySet == []` 확인할 수 있습니다. Cedar 4에서는이 구문을 사용하는 정책이 더 이상 스키마에 대해 검증되지 않습니다. 정책`mySet == []`에서를 로 바꿉니다`mySet.isEmpty()`.

Amazon Verified Permissions 사용 설명서에 대한 문서 기록

다음 표에서는 Verified Permissions에 대한 문서 릴리스를 소개합니다.

변경 사항	설명	날짜
정책 스토어 별칭에 대한 하드 삭제	이제 24시간 `PendingDeletion` 예약 기간을 우회하여 즉시 삭제하기 위해 정책 스토어 별칭을 하드 삭제할 수 있습니다.	2026년 4월 17일
정책 이름 및 정책 템플릿 이름	이제 정책 및 정책 템플릿에 이름을 할당하여 표시 이름으로 참조할 수 있습니다.	2025년 3월 4일
정책 스토어 별칭	이제 정책 스토어 별칭을 생성하여 표시 이름으로 정책 스토어를 참조할 수 있습니다.	2025년 2월 26일
새로운 AWS 관리형 정책	이제 Verified Permissions에서 `AmazonVerifiedPermissionsFullAccess` 및 `AmazonVerifiedPermissionsReadOnlyAccess` IAM 관리형 정책을 사용할 수 있습니다.	2024년 10월 11일
OIDC 자격 증명 소스	이제 OpenID Connect(OIDC) 자격 증명 공급자의 사용자에게 권한을 부여할 수 있습니다.	2024년 6월 8일
자격 증명 소스 토큰을 사용한 배치 권한 부여	이제 단일 `BatchIsAuthorizedWithToken` API 요청으로 Amazon Cognito 사용자 풀의 사용자에게 권한을 부여할 수 있습니다.	2024년 4월 5일

를 사용하여 정책 스토어 생성 API Gateway	이제 기존 API 및 Amazon Cognito 사용자 풀에서 정책 스토어를 생성할 수 있습니다.	2024년 4월 1일
컨텍스트 개념 및 예제	Verified Permissions를 사용한 권한 부여 요청의 컨텍스트에 대한 정보가 추가되었습니다.	2024년 2월 1일
권한 부여 개념 및 예제	Verified Permissions를 사용한 권한 부여 요청에 대한 정보가 추가되었습니다.	2024년 2월 1일
AWS CloudFormation 통합	Verified Permissions는에서 자격 증명 소스, 정책, 정책 스토어 및 정책 템플릿 생성을 지원합니다 CloudFormation.	2023년 6월 30일
최초 릴리스	Amazon Verified Permissions 사용 설명서의 최초 릴리스	2023년 6월 13일

기계 번역으로 제공되는 번역입니다. 제공된 번역과 원본 영어의 내용이 상충하는 경우에는 영어 버전이 우선합니다.