aws

Benutzerhandbuch

Amazon Verified Permissions

Amazon Verified Permissions: Benutzerhandbuch

Table of Contents

Was ist Amazon Verified Permissions?

Amazon Verified Permissions ist ein skalierbarer, detaillierter Berechtigungsverwaltungs- und Autorisierungsservice für benutzerdefinierte Anwendungen, die von Ihnen erstellt wurden. Verified Permissions ermöglicht es Ihren Entwicklern, sichere Anwendungen schneller zu erstellen, indem die Autorisierung externalisiert und die Richtlinienverwaltung und -verwaltung zentralisiert wird. Verified Permissions verwendet die Cedar-Richtliniensprache, um detaillierte Berechtigungen für Anwendungsbenutzer zu definieren.

Themen

Autorisierung im Bereich Verifizierte Berechtigungen

Verified Permissions ermöglicht die Autorisierung, indem überprüft wird, ob ein Principal in einer benutzerdefinierten Anwendung eine Aktion mit einer Ressource in einem bestimmten Kontext ausführen darf. Verified Permissions geht davon aus, dass der Principal zuvor auf andere Weise identifiziert und authentifiziert wurde, z. B. mithilfe von Protokollen wie OpenID Connect, einem gehosteten Anbieter wie Amazon Cognito oder einer anderen Authentifizierungslösung. Verified Permissions ist unabhängig davon, wo der Benutzer verwaltet wird und wie er authentifiziert wurde.

Verified Permissions ist ein Service, der es Kunden ermöglicht, Richtlinien in der zu erstellen, zu verwalten und zu testen. AWS Management Console Berechtigungen werden in der Richtliniensprache von Cedar ausgedrückt. Die Client-Anwendung ruft Autorisierungs-APIs auf, um die im Service gespeicherten Cedar-Richtlinien auszuwerten und eine Zugriffsentscheidung darüber zu treffen, ob eine Aktion zulässig ist.

Sprache der Cedar-Richtlinie

Die Autorisierungsrichtlinien in Verified Permissions werden in der Cedar-Richtliniensprache geschrieben. Cedar ist eine Open-Source-Sprache zum Schreiben von Autorisierungsrichtlinien und zum Treffen von Autorisierungsentscheidungen auf der Grundlage dieser Richtlinien. Wenn Sie eine Anwendung erstellen, müssen Sie sicherstellen, dass nur autorisierte Benutzer auf die Anwendung zugreifen können und nur das tun können, wozu jeder Benutzer berechtigt ist. Mit Cedar können Sie Ihre Geschäftslogik von der Autorisierungslogik entkoppeln. Im Code Ihrer Anwendung stellen Sie Anfragen, die an Ihre Operationen gestellt werden, einen Aufruf an die Cedar-Autorisierungs-Engine mit der Frage „Ist diese Anfrage autorisiert?“ voran. Anschließend kann die Anwendung entweder den angeforderten Vorgang ausführen, wenn die Entscheidung „Zulassen“ lautet, oder eine Fehlermeldung zurückgeben, wenn die Entscheidung „Verweigern“ lautet.

Verified Permissions verwendet derzeit Cedar Version 2.4.

Weitere Informationen zu Cedar finden Sie im Folgenden:

- Referenzhandbuch zur Sprachenpolitik von Cedar
- Cedar GitHub Repository

Vorteile verifizierter Berechtigungen

Beschleunigen Sie die Anwendungsentwicklung

Beschleunigen Sie die Anwendungsentwicklung, indem Sie die Autorisierung von der Geschäftslogik entkoppeln.

Sicherere Anwendungen

Verified Permissions ermöglicht es Entwicklern, sicherere Anwendungen zu erstellen.

Funktionen für Endbenutzer

Mit Verified Permissions können Sie umfassendere Funktionen für die Verwaltung von Berechtigungen für Endbenutzer bereitstellen.

Zugehörige Services

- Amazon Cognito — Amazon Cognito ist eine Identitätsplattform für Web- und mobile Apps. Es handelt sich um ein Benutzerverzeichnis, einen Authentifizierungsserver und einen Autorisierungsservice für OAuth-2.0-Zugriffs-Token und AWS -Anmeldeinformationen. Wenn Sie einen Richtlinienspeicher erstellen, haben Sie die Möglichkeit, Ihre Prinzipale und Gruppen aus einem Amazon Cognito Cognito-Benutzerpool zu erstellen. Weitere Informationen finden Sie im Amazon Cognito Entwicklerhandbuch.
- Amazon API Gateway — Amazon API Gateway ist ein AWS Service für die Erstellung, Veröffentlichung, Wartung, Überwachung und Sicherung von REST, HTTP und WebSocket APIs in jeder Größenordnung. Wenn Sie einen Richtlinienspeicher erstellen, haben Sie die Möglichkeit, Ihre Aktionen und Ressourcen aus einer API in API Gateway zu erstellen. Weitere Informationen zu API Gateway finden Sie im API Gateway Developer Guide.
- AWS IAM Identity Center— Mit IAM Identity Center können Sie die Anmeldesicherheit für Ihre Mitarbeiteridentitäten, auch Workforce-Benutzer genannt, verwalten. IAM Identity Center bietet einen zentralen Ort, an dem Sie Workforce-Benutzer erstellen oder verbinden und ihren Zugriff auf all ihre Anwendungen zentral verwalten können. AWS-Konten Weitere Informationen finden Sie im AWS IAM Identity Center -Benutzerhandbuch.

Zugriff auf verifizierte Berechtigungen

Sie können mit Amazon Verified Permissions auf eine der folgenden Arten arbeiten.

AWS Management Console

Die Konsole ist eine browserbasierte Oberfläche zur Verwaltung verifizierter Berechtigungen und AWS Ressourcen. Weitere Informationen zum Zugriff auf verifizierte Berechtigungen über die Konsole finden Sie im AWS-Anmeldung Benutzerhandbuch unter So melden Sie sich an AWS.

- Konsole „Amazon Verified Permissions“

AWS Tools für die Befehlszeile

Sie können die AWS Befehlszeilentools verwenden, um Befehle an der Befehlszeile Ihres Systems auszugeben, um verifizierte Berechtigungen und AWS Aufgaben auszuführen. Die Verwendung der Kommandozeile kann schneller und bequemer sein als die Konsole. Die Befehlszeilen-Tools können auch beim Erstellen von Skripts für AWS -Aufgaben hilfreich sein.

AWS stellt zwei Gruppen von Befehlszeilentools bereit: das AWS Command Line Interface(AWS CLI) und das AWS Tools for Windows PowerShell. Informationen zur Installation und Verwendung von finden Sie im AWS Command Line Interface Benutzerhandbuch. AWS CLI Informationen zur Installation und Verwendung der Tools für Windows PowerShell finden Sie im AWS Tools for Windows PowerShell Benutzerhandbuch.

- verifiedpermissions in der Befehlsreferenz AWS CLI
- Von Amazon verifizierte Berechtigungen in AWS Tools for Windows PowerShell

AWS SDKs

AWS bietet SDKs (Software Development Kits), die aus Bibliotheken und Beispielcode für verschiedene Programmiersprachen und Plattformen (Java, Python, Ruby, .NET, iOS, Android usw.) bestehen. Die SDKs bieten eine bequeme Möglichkeit, programmatischen Zugriff auf verifizierte Berechtigungen und zu erstellen. AWS Mithilfe der SDKs lassen sich unter anderem Anforderungen kryptografisch signieren, Fehler verwalten und Anforderungen automatisch wiederholen.

Weitere Informationen und das Herunterladen von AWS SDKs finden Sie unter Tools für. Amazon Web Services

Im Folgenden finden Sie Links zur Dokumentation für Ressourcen mit verifizierten Berechtigungen in verschiedenen 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 CDK-Konstrukte

Das AWS Cloud Development Kit (AWS CDK) ist ein Open-Source-Framework für die Softwareentwicklung, mit dem Cloud-Infrastruktur im Code definiert und bereitgestellt werden kann. AWS CloudFormation Konstrukte oder wiederverwendbare Cloud-Komponenten können zum Erstellen von Vorlagen verwendet werden. AWS CloudFormation Diese Vorlagen können dann zur Bereitstellung Ihrer Cloud-Infrastruktur verwendet werden.

Weitere Informationen und das Herunterladen von AWS CDKs finden Sie unter AWS Cloud Development Kit.

Im Folgenden finden Sie Links zur Dokumentation für AWS CDK Ressourcen mit verifizierten Berechtigungen, z. B. Konstrukte.

- Von Amazon verifizierte Berechtigungen L2 CDK Construct

API für verifizierte Berechtigungen

Sie können auf verifizierte Berechtigungen und AWS programmgesteuert zugreifen, indem Sie die Verified Permissions API verwenden, mit der Sie HTTPS-Anfragen direkt an den Dienst senden können. Wenn Sie die -API nutzen, müssen Sie Code zur digitalen Signierung von Anfragen mittels Ihrer Anmeldeinformationen einschließen.

- Referenzhandbuch zur Amazon Verified Permissions API

Preise für verifizierte Berechtigungen

Verified Permissions bietet gestaffelte Preise, die auf der Anzahl der Autorisierungsanfragen pro Monat basieren, die Ihre Anträge bei Verified Permissions stellen. Es gibt auch Preise für Richtlinienverwaltungsaktionen, die auf der Anzahl der cURL-Policy-API-Anfragen (Client-URL) pro Monat basieren, die von Ihren Anwendungen an Verified Permissions gestellt werden.

Eine vollständige Liste der Gebühren und Preise für verifizierte Berechtigungen finden Sie unter Amazon Verified Permissions — Preise.

Um Ihre Rechnung anzuzeigen, navigieren Sie zu Fakturierungs- und Kostenverwaltungs-Dashboard in der AWS Billing and Cost Management -Konsole. Ihre Abrechnung enthält Links zu Nutzungsberichten mit Details zu Ihrer Abrechnung. Weitere Informationen zur AWS-Konto Abrechnung finden Sie im AWS Billing Benutzerhandbuch.

Wenn Sie Fragen zu AWS Abrechnung, Konten und Veranstaltungen haben, wenden Sie sich an AWS Support.

Begriffe und Konzepte von Amazon Verified Permissions

Sie sollten die folgenden Konzepte zur Verwendung von Amazon Verified Permissions verstehen.

Konzepte für verifizierte Berechtigungen

- Autorisierungsmodell
- Autorisierungsanforderung
- Autorisierungsantwort
- Erwägte Richtlinien
- Kontextdaten
- Festlegen von Richtlinien
- Entitätsdaten
- Berechtigungen, Autorisierung und Prinzipale
- Durchsetzung von Richtlinien
- Richtlinienspeicher
- Zuverlässige Richtlinien
- Unterschiede zwischen verifizierten Berechtigungen und Cedar

Konzepte für Zedarrichtlinien

- Autorisierung
- Entität
- Gruppen und Hierarchien
- Namespaces
- Richtlinie
- Richtlinienvorlage
- Schema

Autorisierungsmodell

Das Autorisierungsmodell beschreibt den Umfang der von der Anwendung gestellten Autorisierungsanforderungen und die Grundlage für die Auswertung dieser Anforderungen. Es wird

in Bezug auf die verschiedenen Arten von Ressourcen, die Aktionen, die für diese Ressourcen ausgeführt werden, und die Typen, die Prinzipale ausführen, definiert. Es berücksichtigt auch den Kontext, in dem diese Aktionen durchgeführt werden.

Die rollenbasierte Zugriffskontrolle (RBAC) ist eine Bewertungsbasis, auf der Rollen definiert und einer Reihe von Berechtigungen zugeordnet werden. Diese Rollen können dann einer oder mehreren Identitäten zugewiesen werden. Die zugewiesene Identität erhält die Berechtigungen, die der Rolle zugeordnet sind. Wenn die mit der Rolle verknüpften Berechtigungen geändert werden, wirkt sich die Änderung automatisch auf jede Identität aus, der die Rolle zugewiesen wurde. Cedar kann RBAC-Entscheidungen durch die Verwendung von Prinzipalgruppen unterstützen.

Die attributbasierte Zugriffskontrolle (ABAC) ist eine Auswertungsbasis, bei der die einer Identität zugeordneten Berechtigungen durch Attribute dieser Identität bestimmt werden. Cedar kann ABAC-Entscheidungen durch die Verwendung von Richtlinienbedingungen unterstützen, die auf Attribute des Prinzipals verweisen.

Die Cedar-Richtliniensprache ermöglicht die Kombination von RBAC und ABAC in einer einzigen Richtlinie, indem Berechtigungen für eine Gruppe von Benutzern definiert werden können, die attributbasierte Bedingungen haben.

Autorisierungsanforderung

Eine Autorisierungsanforderung ist eine Anforderung aus Verified Permissions durch eine Anwendung zur Auswertung einer Reihe von Richtlinien, um festzustellen, ob ein Prinzipal eine Aktion für eine Ressource für einen bestimmten Kontext ausführen kann.

Autorisierungsantwort

Die Autorisierungsantwort ist die Antwort auf die Autorisierungsanforderung . Sie enthält eine Entscheidung zum Zulassen oder Verweigern sowie zusätzliche Informationen, wie z. B. die IDs der Bestimmungsrichtlinien.

Erwägte Richtlinien

Als Richtlinien gelten alle Richtlinien, die von Verified Permissions bei der Auswertung einer Autorisierungsanforderung ausgewählt werden.

Kontextdaten

Kontextdaten sind Attributwerte, die zusätzliche Informationen liefern, die ausgewertet werden können.

Festlegen von Richtlinien

Die Festlegung von Richtlinien ist die Richtlinie, die die Autorisierungsantwort bestimmt. Wenn es beispielsweise zwei erfüllte Richtlinien gibt, wobei eine Ablehnung und die andere eine Zugriffserlaubnis ist, dann ist die Zugriffsverweigerungsrichtlinie die Bestimmungsrichtlinie. Wenn es mehrere Richtlinien für eine erfüllte Genehmigung und keine erfüllten verbotenen Richtlinien gibt, gibt es mehrere Bestimmungsrichtlinien. Für den Fall, dass keine Richtlinien übereinstimmen und die Antwort verweigert wird, gibt es keine bestimmenden Richtlinien.

Entitätsdaten

Entitätsdaten sind Daten über den Prinzipal, die Aktion und die Ressource. Entitätsdaten, die für die Richtlinienauswertung relevant sind, sind Gruppenmitgliedschaften bis hinauf zur Entitätshierarchie und Attributwerten des Prinzipals und der Ressource.

Berechtigungen, Autorisierung und Prinzipale

Verified Permissions verwaltet detaillierte Berechtigungen und Autorisierung innerhalb von benutzerdefinierten Anwendungen, die Sie erstellen.

Ein Prinzipal ist der Benutzer einer Anwendung, entweder menschlicher oder maschineller Benutzer, die eine Identität hat, die an eine Kennung wie einen Benutzernamen oder eine Maschinen-ID gebunden ist. Der Authentifizierungsprozess bestimmt, ob der Prinzipal wirklich die Identität ist, für die er sich ausgibt.

Zu dieser Identität gehören eine Reihe von Anwendungsberechtigungen, die bestimmen, was dieser Prinzipal innerhalb dieser Anwendung tun darf. Autorisierung ist der Prozess der Bewertung dieser Berechtigungen, um festzustellen, ob ein Prinzipal eine bestimmte Aktion in der Anwendung ausführen darf. Diese Berechtigungen können als Richtlinien ausgedrückt werden.

Durchsetzung von Richtlinien

Bei der Durchsetzung von Richtlinien wird die Bewertungsentscheidung innerhalb der Anwendung außerhalb von Verified Permissions durchgesetzt. Wenn die Bewertung von Verified Permissions eine Verweigerung zurückgibt, würde die Durchsetzung sicherstellen, dass der Prinzipal am Zugriff auf die Ressource gehindert wurde.

Richtlinienspeicher

Ein Richtlinienspeicher ist ein Container für Richtlinien und Vorlagen. Jeder Speicher enthält ein Schema, mit dem Richtlinien validiert werden, die dem Speicher hinzugefügt wurden. Standardmäßig hat jede Anwendung ihren eigenen Richtlinienspeicher, aber mehrere Anwendungen können einen einzelnen Richtlinienspeicher gemeinsam nutzen. Wenn eine Anwendung eine Autorisierungsanforderung stellt, identifiziert sie den Richtlinienspeicher, der zur Bewertung dieser Anforderung verwendet wird. Richtlinienspeicher bieten eine Möglichkeit, eine Reihe von Richtlinien zu isolieren, und können daher in einer mandantenfähigen Anwendung verwendet werden, um die Schemata und Richtlinien für jeden Mandanten zu enthalten. Eine einzelne Anwendung kann separate Richtlinienspeicher für jeden Mandanten haben.

Bei der Auswertung einer Autorisierungsanforderung berücksichtigt Verified Permissions nur die Teilmenge der Richtlinien im Richtlinienspeicher, die für die Anforderung relevant sind. Die Relevanz wird basierend auf dem Geltungsbereich der Richtlinie bestimmt. Der Bereich identifiziert den spezifischen Prinzipal und die Ressource, für den/die die Richtlinie gilt, sowie die Aktionen, die der Prinzipal für die Ressource ausführen kann. Die Definition des Umfangs trägt zur Verbesserung der Leistung bei, indem die Menge der berücksichtigten Richtlinien eingeschränkt wird.

Zuverlässige Richtlinien

Zu zufriedene Richtlinien sind die Richtlinien, die den Parametern der Autorisierungsanforderung entsprechen.

Unterschiede zwischen verifizierten Berechtigungen und Cedar

Amazon Verified Permissions verwendet zur Ausführung seiner Autorisierungsaufgaben die Cedar Policy Language Engine. Es gibt jedoch einige Unterschiede zwischen der systemeigenen Cedar-Implementierung und der Implementierung von Cedar, die in Verified Permissions zu finden sind. In diesem Thema werden diese Unterschiede beschrieben.

Namespace-Definition

Die Implementierung von Verified Permissions von Cedar unterscheidet sich von der nativen Cedar-Implementierung wie folgt:

- Verified Permissions unterstützt nur einNamespace in einem Schemain einem Richtlinienspeicher definiert.
- Sie können eine nicht über die ändern.Namensraummit den folgenden Werten:`aws`,`amazon`, oder`cedar`.

Unterstützung für Richtlinienvorlagen

Sowohl Verified Permissions als auch Cedar erlauben Platzhalter im Gültigkeitsbereich nur für`principal`und`resource`. Verified Permissions erfordert jedoch auch, dass weder`principal`und`resource`sind uneingeschränkt.

Die folgende Richtlinie ist in Cedar gültig, wird jedoch von Verified Permissions abgelehnt, weil`principal`ist uneingeschränkt.

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

Die beiden folgenden Beispiele sind sowohl für Cedar als auch für Verified Permissions gültig, da beide`principal`und`resource`Einschränkungen haben.

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

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

Unterstützung für Schemas

Verified Permissions erfordert, dass alle JSON-Schlüsselnamen des Schemas keine leeren Zeichenfolgen sind. Cedar erlaubt in einigen Fällen leere Zeichenfolgen, z. B. für Eigenschaften.

Unterstützung für Erweiterungstypen

Verified Permissions unterstützt CedarErweiterungstypenin Richtlinien, unterstützt aber derzeit nicht, sie in die Definition eines Schemas oder als Teil der`entities`Parameter des`IsAuthorized`und`IsAuthorizedWithToken`Operationen.

Zu den Erweiterungstypen gehört der Festpunkt (`decimal`) und IP-Adresse (`ipaddr`) Datentypen.

Cedar JSON-Format für Entitäten

Derzeit müssen Sie bei Verified Permissions die Liste der Entitäten, die in einer Autorisierungsanfrage berücksichtigt werden sollen, anhand der für dieEntitiesDefinition, das ist ein Array vonEntityItemElemente. Verified Permissions unterstützt derzeit nicht die Weitergabe der Liste der Entitäten, die in einer Autorisierungsanfrage berücksichtigt werden sollenJSON-Format. Spezifische Anforderungen für die Formatierung Ihrer Entitäten für die Verwendung in Verified Permissions finden Sie unterFormatierung von Entitäten in Amazon Verified Permissions.

Definition von Aktionsgruppen

Die Autorisierungsmethoden von Cedar erfordern eine Liste der Entitäten, die bei der Bewertung einer Autorisierungsanfrage anhand der Richtlinien berücksichtigt werden müssen.

Sie können die Aktionen und Aktionsgruppen, die von Ihrer Anwendung verwendet werden, im Schema definieren. Cedar nimmt das Schema jedoch nicht als Teil einer Evaluierungsanfrage auf. Stattdessen verwendet Cedar das Schema nur, um die von Ihnen eingereichten Richtlinien und Richtlinienvorlagen zu validieren. Da Cedar bei Bewertungsanfragen nicht auf das Schema verweist, selbst wenn Sie Aktionsgruppen im Schema definiert haben, müssen Sie auch die Liste aller Aktionsgruppen als Teil der Entitätenliste angeben, die Sie an die Autorisierungs-API-Operationen übergeben müssen.

Verified Permissions erledigt das für Sie. Alle Aktionsgruppen, die Sie in Ihrem Schema definieren, werden automatisch an die Entitätsliste angehängt, an die Sie als Parameter übergeben`IsAuthorized`oder`IsAuthorizedWithToken`Operationen.

Längen- und Größenbeschränkungen

Verified Permissions unterstützt die Speicherung in Form von Policy-Stores für Ihr Schema, Ihre Richtlinien und Richtlinienvorlagen. Aufgrund dieser Speicherung legt Verified Permissions einige Längen- und Größenbeschränkungen fest, die für Cedar nicht relevant sind.

Objekt	Verifiziertes Berechtigungslimit (in Byte)	Zederngrenze
Umfang der Police[1]	10.000	Keine

Objekt	Verifiziertes Berechtigungslimit (in Byte)	Zederngrenze
Beschreibung der Inline-Richtlinie	150	Gilt nicht für Zeder
Größe der Richtlinienvorlage	10.000	Keine
Größe des Schemas	10.000	Keine
Typ der Entität	200	Keine
Policy ID (Richtlinien-ID)	64	Keine
ID der Richtlinienvorlage	64	Keine
Entity-ID	200	Keine
ID des Richtlinienspeichers	64	Gilt nicht für Zeder

[1] Unter Verifizierte Berechtigungen gibt es eine Obergrenze für Richtlinien pro Richtlinienspeicher, die auf der Gesamtgröße der im Richtlinienspeicher erstellten Richtlinien, Aktionen und Ressourcen basiert. Die Gesamtgröße aller Richtlinien, die sich auf eine einzelne Ressource beziehen, darf 200.000 Byte nicht überschreiten. Bei Richtlinien, die mit Vorlagen verknüpft sind, wird die Größe der Richtlinienvorlage nur einmal gezählt, zuzüglich der Größe jedes Parametersatzes, der zur Instanziierung jeder mit der Vorlage verknüpften Richtlinie verwendet wird.

Erste Schritte mit verifizierten Berechtigungen

Verwenden Sie dieses Tutorial, um mit Amazon Verified Permissions zu beginnen.

Themen

Melden Sie sich für eine an AWS-Konto

Wenn Sie noch keine haben AWS-Konto, führen Sie die folgenden Schritte aus, um eine zu erstellen.

Um sich für eine anzumelden AWS-Konto

1. Öffnen Sie https://portal.aws.amazon.com/billing/signup.
2. Folgen Sie den Online-Anweisungen.

   Bei der Anmeldung müssen Sie auch einen Telefonanruf entgegennehmen und einen Verifizierungscode über die Telefontasten eingeben.

   Wenn Sie sich für eine anmelden AWS-Konto, Root-Benutzer des AWS-Kontoswird eine erstellt. Der Root-Benutzer hat Zugriff auf alle AWS-Services und Ressourcen des Kontos. Aus Sicherheitsgründen sollten Sie einem Benutzer Administratorzugriff zuweisen und nur den Root-Benutzer verwenden, um Aufgaben auszuführen, für die Root-Benutzerzugriff erforderlich ist.

AWS sendet Ihnen nach Abschluss des Anmeldevorgangs eine Bestätigungs-E-Mail. Sie können jederzeit Ihre aktuelle Kontoaktivität anzeigen und Ihr Konto verwalten. Rufen Sie dazu https://aws.amazon.com/ auf und klicken Sie auf Mein Konto.

Erstellen Sie einen Benutzer mit Administratorzugriff

Nachdem Sie sich für einen angemeldet haben AWS-Konto, sichern Sie Ihren Root-Benutzer des AWS-Kontos AWS IAM Identity Center, aktivieren und erstellen Sie einen Administratorbenutzer, sodass Sie den Root-Benutzer nicht für alltägliche Aufgaben verwenden.

Sichern Sie Ihre Root-Benutzer des AWS-Kontos

1. Melden Sie sich AWS Management Consoleals Kontoinhaber an, indem Sie Root-Benutzer auswählen und Ihre AWS-Konto E-Mail-Adresse eingeben. Geben Sie auf der nächsten Seite Ihr Passwort ein.

   Hilfe bei der Anmeldung mit dem Root-Benutzer finden Sie unter Anmelden als Root-Benutzer im AWS-Anmeldung Benutzerhandbuch zu.
2. Aktivieren Sie die Multi-Faktor-Authentifizierung (MFA) für den Root-Benutzer.

   Anweisungen finden Sie im Benutzerhandbuch unter Aktivieren eines virtuellen MFA-Geräts für Ihren AWS-Konto IAM Root-Benutzer (Konsole).

Erstellen Sie einen Benutzer mit Administratorzugriff

1. Aktivieren Sie das IAM Identity Center.

   Anweisungen finden Sie unter Aktivieren AWS IAM Identity Center im AWS IAM Identity Center Benutzerhandbuch.
2. Gewähren Sie einem Benutzer in IAM Identity Center Administratorzugriff.

   Ein Tutorial zur Verwendung von IAM-Identity-Center-Verzeichnis als Identitätsquelle finden Sie unter Benutzerzugriff mit der Standardeinstellung konfigurieren IAM-Identity-Center-Verzeichnis im AWS IAM Identity Center Benutzerhandbuch.

Melden Sie sich als Benutzer mit Administratorzugriff an

- Um sich mit Ihrem IAM-Identity-Center-Benutzer anzumelden, verwenden Sie die Anmelde-URL, die an Ihre E-Mail-Adresse gesendet wurde, als Sie den IAM-Identity-Center-Benutzer erstellt haben.

Hilfe bei der Anmeldung mit einem IAM Identity Center-Benutzer finden Sie im AWS-Anmeldung Benutzerhandbuch unter Anmeldung beim AWS Zugriffsportal.

Weisen Sie weiteren Benutzern Zugriff zu

1. Erstellen Sie in IAM Identity Center einen Berechtigungssatz, der der bewährten Methode zur Anwendung von Berechtigungen mit den geringsten Rechten folgt.

   Anweisungen finden Sie im Benutzerhandbuch unter Einen Berechtigungssatz erstellen.AWS IAM Identity Center
2. Weisen Sie Benutzer einer Gruppe zu und weisen Sie der Gruppe dann Single Sign-On-Zugriff zu.

   Anweisungen finden Sie im AWS IAM Identity Center Benutzerhandbuch unter Gruppen hinzufügen.

IAM Richtlinien für verifizierte Berechtigungen

Verified Permissions verwaltet die Berechtigungen von Benutzern innerhalb Ihrer Anwendung. Damit Ihre Anwendung die Verified Permissions APIs aufrufen kann oder damit AWS Management Console Benutzer Cedar-Richtlinien in einem Richtlinienspeicher für verifizierte Berechtigungen verwalten können, müssen Sie die erforderlichen IAM Berechtigungen hinzufügen.

Identitätsbasierte Richtlinien sind Richtliniendokumente für JSON-Berechtigungen, die Sie an eine Identität anhängen können, z. B. an einen IAM Benutzer, eine Benutzergruppe oder eine Rolle. Diese Richtlinien steuern, welche Aktionen die Benutzer und Rollen für welche Ressourcen und unter welchen Bedingungen ausführen können. Informationen zum Erstellen einer identitätsbasierten Richtlinie finden Sie unter IAM Richtlinien erstellen im Benutzerhandbuch. IAM

Mit IAM identitätsbasierten Richtlinien können Sie zulässige oder verweigerte Aktionen und Ressourcen sowie die Bedingungen angeben, unter denen Aktionen zugelassen oder verweigert werden (siehe unten). Sie können den Prinzipal nicht in einer identitätsbasierten Richtlinie angeben, da er für den Benutzer oder die Rolle gilt, dem er zugeordnet ist. Weitere Informationen zu allen Elementen, die Sie in einer JSON-Richtlinie verwenden können, finden Sie im IAM Benutzerhandbuch unter Referenz zu IAM JSON-Richtlinienelementen.

Action (Aktion)	Beschreibung
CreatePolicyStore	Aktion zum Erstellen eines neuen Richtlini enspeichers.
DeletePolicyStore	Aktion zum Löschen eines Richtlinienspeichers.
ListPolicyStores	Aktion zum Auflisten aller Richtlinienspeicher in der AWS-Konto.
CreatePolicy	Aktion zum Erstellen einer Cedar-Richtlinie in einem Richtlinienspeicher. Sie können entweder eine statische Richtlinie oder eine mit einer Richtlinienvorlage verknüpfte Richtlinie erstellen.
DeletePolicy	Aktion zum Löschen einer Richtlinie aus einem Richtlinienspeicher.
GetPolicy	Aktion zum Abrufen von Informationen zu einer bestimmten Richtlinie.
ListPolicies	Aktion zum Auflisten aller Richtlinien in einem Richtlinienspeicher.
IsAuthorized	Aktion, um eine Autorisierungsantwort auf der Grundlage der in der Autorisierungsanfrage beschriebenen Parameter zu erhalten.

IAM Beispielrichtlinie für die Genehmigung der CreatePolicy Aktion:

```
{
"Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "verifiedpermissions:CreatePolicy"
            ],
```

```
            "Resource": "*"
        }
    ]
}
```

Erstellen Sie Ihren ersten Richtlinienspeicher für verifizierte Berechtigungen

Wenn Sie sich zum ersten Mal bei der Verified Permissions-Konsole anmelden, können Sie wählen, wie Sie Ihren ersten Richtlinienspeicher und Ihre erste Cedar-Richtlinie erstellen möchten. Folgen Sie dem Anmeldeverfahren, das Ihrem Benutzertyp entspricht, wie im Abschnitt So melden Sie sich bei AWS an im AWS -Anmelde-Benutzerhandbuch beschrieben. Wählen Sie auf der Startseite der Konsole den Service Amazon Verified Permissions aus. Wählen Sie Erste Schritte.

Einen Beispiel-Policy-Store erstellen

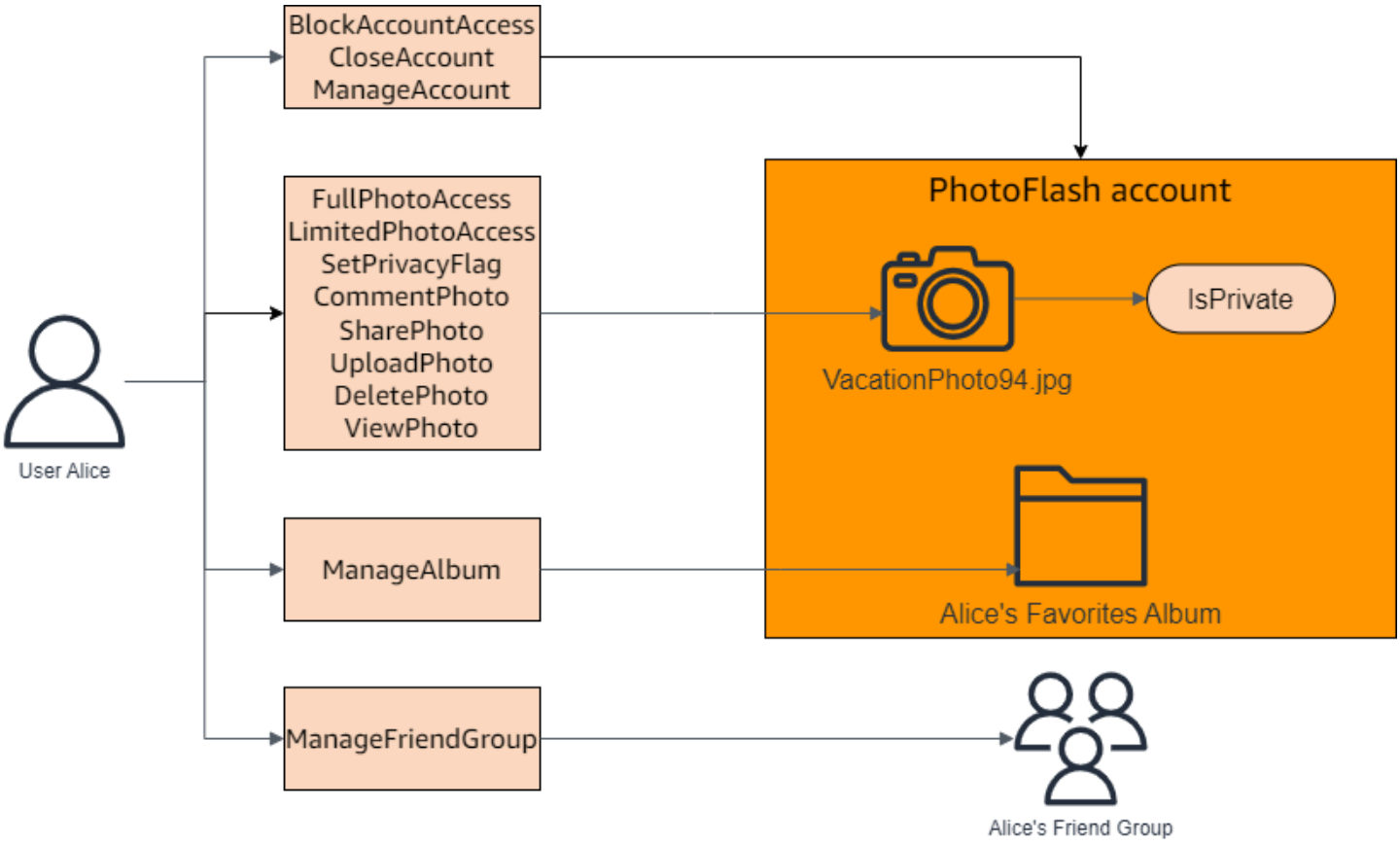
Wenn Sie Verifizierte Berechtigungen zum ersten Mal verwenden, empfehlen wir, einen der Beispiel-Richtlinienspeicher zu verwenden, um sich mit der Funktionsweise von Verified Permissions vertraut zu machen. Die Beispiel-Richtlinienspeicher bieten vordefinierte Richtlinien und ein Schema.

So erstellen Sie einen Richtlinienspeicher mit der Konfigurationsmethode „Beispiel für den Richtlinienspeicher“

1. Wählen Sie in der Konsole „Verifizierte Berechtigungen“ die Option Neuen Richtlinienspeicher erstellen aus.
2. Wählen Sie im Abschnitt Startoptionen die Option Beispielrichtlinienspeicher aus.
3. Wählen Sie im Abschnitt Beispielprojekt den Typ der Beispielanwendung Verified Permissions aus, die Sie verwenden möchten. Wählen Sie für dieses Tutorial den PhotoFlashRichtlinienspeicher aus.
4. Basierend auf dem ausgewählten Beispielprojekt wird automatisch ein Namespace für das Schema Ihres Beispielrichtlinienspeichers generiert.
5. Wählen Sie Richtlinienspeicher erstellen aus.

   Ihr Richtlinienspeicher wird mit Richtlinien, Richtlinienvorlagen und einem Schema für den Beispiel-Richtlinienspeicher erstellt.

Das folgende Diagramm veranschaulicht die Beziehungen zwischen den PhotoFlash Beispielaktionen für den Richtlinienspeicher und den Ressourcentypen, für die sie gelten.



Erstellen von mit Vorlagen verknüpften Richtlinien für einen Beispielrichtlinienspeicher

Der PhotoFlash Beispielrichtlinienspeicher umfasst Richtlinien, Richtlinienvorlagen und ein Schema. Sie können mit Vorlagen verknüpfte Richtlinien auf der Grundlage der Richtlinienvorlagen erstellen, die im Beispielrichtlinienspeicher enthalten sind.

So erstellen Sie mit Vorlagen verknüpfte Richtlinien für den Beispielrichtlinienspeicher

1. Öffnen Sie die Konsole Verified Permissions unter https://console.aws.amazon.com/verifiedpermissions/. Wählen Sie Ihren Richtlinienspeicher aus.
2. Wählen Sie im Navigationsbereich auf der linken Seite Policies (Richtlinien).
3. Wählen Sie Richtlinie erstellen und dann Richtlinie mit Vorlage erstellen aus.
4. Wählen Sie das Optionsfeld neben der Richtlinienvorlage mit der Beschreibung Vollzugriff auf nicht private geteilte Fotos gewähren aus und klicken Sie dann auf Weiter.

5. Geben `PhotoFlash::User::"Alice"` Sie für Principal ein. Geben Sie für Ressource den Wert ein`PhotoFlash::Album::"Bob-Vacation-Album"`.
6. Wählen Sie „Mit Vorlage verknüpfte Richtlinie erstellen“.

   Die neue, mit der Vorlage verknüpfte Richtlinie wird unter Richtlinien angezeigt.
7. Erstellen Sie eine weitere mit einer Vorlage verknüpfte Richtlinie für den PhotoFlash Beispielrichtlinienspeicher. Wählen Sie Richtlinie erstellen und anschließend mit Vorlage verknüpfte Richtlinie erstellen aus.
8. Wählen Sie das Optionsfeld neben der Richtlinienvorlage mit der Beschreibung Beschränkten Zugriff auf geteilte Fotos gewähren aus und wählen Sie dann Weiter aus.
9. Geben `PhotoFlash::FriendGroup::"MySchoolFriends"` Sie für Principal ein. Geben Sie für Ressource den Wert ein`PhotoFlash::Album::"Alice’s favorite album"`.
10. Wählen Sie „Mit Vorlage verknüpfte Richtlinie erstellen“.

    Die neue, mit der Vorlage verknüpfte Richtlinie wird unter Richtlinien angezeigt.

Wir werden die neuen, mit Vorlagen verknüpften Richtlinien im nächsten Abschnitt des Tutorials testen. Weitere Beispiele für Werte, für die Sie eine mit einer Vorlage verknüpfte Richtlinie erstellen können, finden Sie unter. PhotoFlash PhotoFlash Beispiele für vorlagenverknüpfte Richtlinien

Testen eines Beispiel-Richtlinienspeichers

Nachdem Sie Ihren Beispielrichtlinienspeicher und die mit Vorlagen verknüpften Richtlinien erstellt haben, können Sie die statischen Beispielrichtlinien für verifizierte Berechtigungen und Ihre neuen vorlagenverknüpften Richtlinien testen, indem Sie eine simulierte Autorisierungsanfrage mit dem Prüfstand für verifizierte Berechtigungen ausführen.

Je nachdem, wann Sie Ihren Beispielrichtlinienspeicher erstellt haben, können sich Ihre Richtlinienvorlagen von den Referenzen in diesem Verfahren unterscheiden. Bevor Sie mit diesem Teil des Tutorials beginnen, überprüfen Sie, ob Sie alle nachfolgenden Richtlinienvorlagen in Ihrem PhotoFlash Beispiel-Richtlinienspeicher haben. Wenn Ihre Richtlinie nicht mit diesen Richtlinien übereinstimmt, bearbeiten Sie die vorhandenen Richtlinien oder erstellen Sie über die Option Beispielprojekt einen neuen Richtlinienspeicher PhotoFlash.

Gewähren Sie vollen Zugriff auf geteilte Fotos, die nicht privat sind

```
permit (
```

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

Gewähren Sie eingeschränkten Zugriff auf geteilte Fotos, die nicht privat sind

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

Um Beispiele für Policy-Store-Richtlinien zu testen

1. Öffnen Sie die Konsole „Verified Permissions“ unter https://console.aws.amazon.com/verifiedpermissions/. Wählen Sie Ihren Richtlinienspeicher aus.
2. Wählen Sie im Navigationsbereich auf der linken Seite die Option Testbench aus.
3. Wählen Sie den Visuellen Modus.
4. Wählen Sie im Abschnitt Principal PhotoFlash: :User aus den Prinzipaltypen in Ihrem Schema aus. Geben Sie einen Bezeichner für den Benutzer in das Textfeld ein. z. B. `Alice`.
5. Wählen Sie für den Prinzipal nicht die Option Übergeordnetes Element hinzufügen aus.
6. Stellen Sie für das Attribut Account: Entity sicher, dass die Entität PhotoFlash: :Account ausgewählt ist. Geben Sie eine Kennung für das Konto ein. z. B. `Alice-account`.
7. Wählen Sie im Bereich Ressource den Ressourcentyp PhotoFlash: :Photo aus. Geben Sie eine Kennung für das Foto in das Textfeld ein. z. B. `photo.jpeg`.
8. Wählen Sie Add a parent und wählen Sie PhotoFlash: :Account als Entitätstyp aus. Geben Sie dieselbe Kennung für das Elternkonto für das Foto ein, die Sie im Feld Konto: Entität für den Benutzer angegeben haben. z. B. `Alice-account`.
9. Wählen Sie im Abschnitt Aktion PhotoFlash: :Action::“ ViewPhoto "aus der Liste der gültigen Aktionen aus.
10. Wählen Sie im Abschnitt Zusätzliche Entitäten die Option Diese Entität hinzufügen aus, um die vorgeschlagene Kontoentität hinzuzufügen.

11. Wählen Sie oben auf der Seite Autorisierungsanfrage ausführen aus, um die Autorisierungsanfrage für die Cedar-Richtlinien im Beispielrichtlinienspeicher zu simulieren. Auf dem Prüfstand sollte die Entscheidung, die Anfrage zuzulassen, angezeigt werden.

Die folgende Tabelle enthält zusätzliche Werte für den Prinzipal, die Ressource und die Aktion, die Sie mit dem Prüfstand für verifizierte Berechtigungen testen können. Die Tabelle enthält die Entscheidung über die Autorisierungsanfrage auf der Grundlage der statischen Richtlinien, die im PhotoFlash Beispielrichtlinienspeicher enthalten sind, und der mit Vorlagen verknüpften Richtlinien, die Sie im vorherigen Abschnitt erstellt haben.

Hauptwert	Hauptkonto: Unternehmenswert	Wert der Ressource	Wert der übergeordneten Ressource	Action (Aktion)	Entscheidung über die Autorisierung
PhotoFlash: :Benutzer | Alice	PhotoFlash: :Konto | Alice-Konto	PhotoFlash: :Foto | photo.jpeg	PhotoFlash: :Konto | BOB-Konto	PhotoFlash: :Aktion::“ "ViewPhoto	Deny
PhotoFlash: :Benutzer | Alice	PhotoFlash: :Konto | Alice-Konto	PhotoFlash: :Foto | photo.jpeg	PhotoFlash: :Konto | ALICE-Konto	PhotoFlash: :Aktion::“ "ViewPhoto	Sobald Sie die Details auf dieser Seite überprüft haben, klicken Sie auf
PhotoFlash: :Benutzer | Alice	PhotoFlash: :Konto | Alice-Konto	PhotoFlash: :Foto | Bob-photo.jpeg	PhotoFlash:: Album | Bob-Urlaubsalbum	PhotoFlash: :Aktion::“ "ViewPhoto	Sobald Sie die Details auf dieser Seite überprüft haben, klicken Sie auf

Hauptwert	Hauptkonto: Unternehmenswert	Wert der Ressource	Wert der übergeordneten Ressource	Action (Aktion)	Entscheidung über die Autorisierung
PhotoFlash: :Benutzer | Alice	PhotoFlash: :Konto | Alice-Konto	PhotoFlash: :Foto | Bob-photo.jpeg	PhotoFlash:: Album | Bob-Urlaubsalbum	PhotoFlash: :Aktion::“ "DeletePhoto	Deny
PhotoFlash: :Benutzer | Alice	PhotoFlash: :Konto | Alice-Konto	PhotoFlash: :Photo | Bob-photo.jpeg,IsPrivate: Boolean | wahr	PhotoFlash:: Album | Bob-Urlaubsalbum	PhotoFlash: :Aktion::“ "ViewPhoto	Deny
PhotoFlash: :Benutzer | Jane, PhotoFlash:: | FriendGroup MySchoolFriends	PhotoFlash: :Konto | Jane-Konto	PhotoFlash: :Foto | photo.jpeg	PhotoFlash: :Album | Alices Lieblingsalbum	PhotoFlash: :Aktion::“ "ViewPhoto	Sobald Sie die Details auf dieser Seite überprüft haben, klicken Sie auf
PhotoFlash: :Benutzer | Jane, PhotoFlash:: | FriendGroup MySchoolFriends	PhotoFlash: :Konto | Jane-Konto	PhotoFlash: :Foto | photo.jpeg	PhotoFlash: :Album | Alices Lieblingsalbum	PhotoFlash: :Aktion::“ "DeletePhoto	Deny

Erstellen Sie einen Richtlinienspeicher mit einer verbundenen API und einem Identitätsanbieter

Ein häufiger Anwendungsfall für Amazon Verified Permissions ist die Autorisierung von Anfragen von einem Anwendungsclient an eine Back-End-API. AWS hat einen Dienst zur Authentifizierung von Anwendungsbenutzern: Amazon Cognito. AWS hat auch einen Service für sichere gehostete APIs: Amazon API Gateway. Wenn Sie einen Richtlinienspeicher für verifizierte Berechtigungen mit diesen beiden kombinieren AWS-Services, können Sie die Benutzerpool-Authentifizierung und die API-Autorisierung in Ihrer Anwendung mit einem konsistenten, zentralisierten Satz von Richtlinien verbinden. Richtlinienspeicher für verifizierte Berechtigungen bieten integrierte Unterstützung für Amazon Cognito Cognito-Benutzerpool-Identitätsquellen und API Gateway Gateway-APIs.

Um einen Richtlinienspeicher zu erstellen, der mit einem vorhandenen Benutzerpool und einer API verknüpft ist, wählen Sie Beim Erstellen eines neuen Richtlinienspeichers die Option Mit Cognito und API Gateway einrichten.

Ein API-verknüpfter Richtlinienspeicher stellt automatisch Ihr Autorisierungsmodell und Ihre Ressourcen für Autorisierungsanfragen bereit. Der Erstellungsprozess „Setup with Cognito and API Gateway“ generiert einen Richtlinienspeicher mit einer Benutzerpool-Identitätsquelle und einem Lambda-Autorisierer, der API Gateway mit verifizierten Berechtigungen verbindet. Anfänglich können Sie API-Anfragen auf der Grundlage der Gruppenmitgliedschaften der Benutzer autorisieren. Verifizierte Berechtigungen können beispielsweise nur Benutzern Zugriff gewähren, die Mitglieder der `Directors` Gruppe sind.

Wenn Ihre Anwendung wächst, können Sie eine differenzierte Autorisierung mit Benutzerattributen und OAuth 2.0-Bereichen implementieren. Verifizierte Berechtigungen können beispielsweise nur Benutzern Zugriff gewähren, die über ein `email` Attribut in der Domäne verfügen. `mycompany.co.uk`

Nachdem Sie das Autorisierungsmodell für Ihre API automatisiert haben, sind Sie weiterhin dafür verantwortlich, Benutzer zu authentifizieren und API-Anfragen in Ihrer Anwendung zu generieren sowie Ihren Richtlinienspeicher zu verwalten.

Weitere Informationen hierzu finden Sie unter API-verknüpfte Richtlinienspeicher.

Richtlinien von Amazon Verified Permissions

Ein Richtlinienspeicher ist ein Container für Richtlinien und Richtlinienvorlagen. Jeder Richtlinienspeicher enthält ein Schema, das verwendet wird, um Richtlinien zu validieren, die dem Richtlinienspeicher hinzugefügt wurden. Wir empfehlen, einen Richtlinienspeicher pro Anwendung oder einen Richtlinienspeicher pro Mandant für Anwendungen mit mehreren Mandanten zu erstellen. Sie müssen einen Richtlinienspeicher angeben, wenn Sie eine Autorisierungsanfrage stellen.

Wir empfehlen, in Ihren Richtlinienspeichern Namespaces für Cedar-Entitäten zu verwenden, um Unklarheiten zu vermeiden. Ein Namespace ist ein Zeichenkettenpräfix für einen Typ, getrennt durch ein Paar Doppelpunkte () als Trennzeichen. `::` Verified Permissions unterstützt einen Namespace pro Richtlinienspeicher. Weitere Informationen finden Sie unter Namespaces im Cedar Policy Language Reference Guide.

Themen

Richtlinienspeicher für verifizierte Berechtigungen erstellen

Sie können einen Richtlinienspeicher mit den folgenden Methoden erstellen:

- Folgen Sie einer Anleitung zur Einrichtung — Sie definieren einen Ressourcentyp mit gültigen Aktionen und einen Prinzipaltyp, bevor Sie Ihre erste Richtlinie erstellen.
- Mit API Gateway und einer Identitätsquelle einrichten — Definieren Sie Ihre Hauptentitäten mit Benutzern, die sich mit einem Identitätsanbieter (IdP) anmelden, und Ihre Aktionen und Ressourcenentitäten über eine Amazon API Gateway Gateway-API. Wir empfehlen diese Option, wenn Sie möchten, dass Ihre Anwendung API-Anfragen mit der Gruppenmitgliedschaft von Benutzern autorisiert.
- Beginnen Sie mit einem Beispielrichtlinienspeicher — Wählen Sie einen vordefinierten Beispiel-Richtlinienspeicher für Projekte aus. Wir empfehlen diese Option, wenn Sie mehr über verifizierte Berechtigungen erfahren und Beispielrichtlinien ansehen und testen möchten.

- Erstellen Sie einen leeren Richtlinienspeicher — Sie definieren das Schema und alle Zugriffsrichtlinien selbst. Wir empfehlen diese Option, wenn Sie bereits mit der Konfiguration eines Richtlinienspeichers vertraut sind.

Guided setup

So erstellen Sie einen Richtlinienspeicher mit der Konfigurationsmethode „Geführtes Setup“

Der Assistent für die Einrichtung mit Anleitung führt Sie durch den Prozess der Erstellung der ersten Iteration Ihres Richtlinienspeichers. Sie erstellen ein Schema für Ihren ersten Ressourcentyp, beschreiben die Aktionen, die für diesen Ressourcentyp gelten, und beschreiben den Prinzipaltyp, für den Sie Berechtigungen erteilen. Anschließend erstellen Sie Ihre erste Richtlinie. Sobald Sie diesen Assistenten abgeschlossen haben, können Sie Ihrem Richtlinienspeicher etwas hinzufügen, das Schema erweitern, um andere Ressourcen- und Prinzipaltypen zu beschreiben, und zusätzliche Richtlinien und Vorlagen erstellen.

1. Wählen Sie in der Konsole „Verifizierte Berechtigungen“ die Option Neuen Richtlinienspeicher erstellen aus.
2. Wählen Sie im Abschnitt Startoptionen die Option Geführte Installation aus.
3. Geben Sie eine Beschreibung des Policy-Stores ein. Bei diesem Text kann es sich um einen beliebigen Text handeln, der zu Ihrer Organisation passt, und zwar als freundlicher Hinweis auf die Funktion des aktuellen Richtlinienspeichers, z. B. Wetteraktualisierungen.
4. Geben Sie im Abschnitt Details einen Namespace für Ihr Schema ein.
5. Wählen Sie Weiter aus.
6. Geben Sie im Fenster Ressourcentyp einen Namen für Ihren Ressourcentyp ein.
7. (Optional) Wählen Sie Attribut hinzufügen, um Ressourcenattribute hinzuzufügen. Geben Sie den Attributnamen ein und wählen Sie einen Attributtyp für jedes Attribut der Ressource aus. Wählen Sie aus, ob jedes Attribut erforderlich ist. Verified Permissions verwendet die angegebenen Attributwerte, wenn Richtlinien anhand des Schemas überprüft werden. Um ein Attribut zu entfernen, das für den Ressourcentyp hinzugefügt wurde, wählen Sie neben dem Attribut die Option Entfernen aus.
8. Geben Sie im Feld Aktionen die Aktionen ein, die für den angegebenen Ressourcentyp autorisiert werden sollen. Um weitere Aktionen für den Ressourcentyp hinzuzufügen, wählen Sie Aktion hinzufügen aus. Um eine Aktion zu entfernen, die für den Ressourcentyp hinzugefügt wurde, wählen Sie neben der Aktion die Option Entfernen aus.

9. Geben Sie im Feld Name des Prinzipaltyps den Namen für einen Prinzipaltyp ein, der die angegebenen Aktionen für Ihren Ressourcentyp verwenden wird.
10. Wählen Sie Weiter aus.
11. Wählen Sie im Fenster Prinzipaltyp die Identitätsquelle für Ihren Prinzipaltyp aus.

    - Wählen Sie Benutzerdefiniert, wenn die ID und die Attribute des Prinzipals direkt von Ihrer Verified Permissions-Anwendung bereitgestellt werden sollen. Wählen Sie Attribut hinzufügen, um Hauptattribute hinzuzufügen. Geben Sie den Attributnamen ein und wählen Sie einen Attributtyp für jedes Attribut des Prinzipals aus. Verified Permissions verwendet die angegebenen Attributwerte, wenn Richtlinien anhand des Schemas überprüft werden. Um ein Attribut zu entfernen, das für den Prinzipaltyp hinzugefügt wurde, wählen Sie neben dem Attribut die Option Entfernen aus.
    - Wählen Sie Cognito User Pool, wenn die ID und die Attribute des Prinzipals aus einer von Amazon Cognito generierten ID oder einem Zugriffstoken bereitgestellt werden. Wählen Sie Connect user pool. Wählen Sie die AWS-RegionBenutzerpool-ID des Amazon Cognito Cognito-Benutzerpools aus, zu dem Sie eine Verbindung herstellen möchten, und geben Sie sie ein. Wählen Sie Connect aus. Weitere Informationen finden Sie unter Autorisierung mit von Amazon verifizierten Berechtigungen im Amazon Cognito Developer Guide.
12. Wählen Sie Weiter aus.
13. Geben Sie im Abschnitt Richtliniendetails eine optionale Richtlinienbeschreibung für Ihre erste Cedar-Police ein.
14. Wählen Sie im Feld Principals Scope die Principals aus, denen im Rahmen der Richtlinie Berechtigungen erteilt werden sollen.

    - Wählen Sie Spezifischer Hauptbenutzer aus, um die Richtlinie auf einen bestimmten Prinzipal anzuwenden. Wählen Sie den Principal im Feld Principal, der Aktionen ausführen darf, und geben Sie eine Entitäts-ID für den Principal ein.
    - Wählen Sie Alle Prinzipale aus, um die Richtlinie auf alle Prinzipale in Ihrem Richtlinienspeicher anzuwenden.
15. Wählen Sie im Feld Umfang der Ressourcen aus, auf welche Ressourcen die angegebenen Prinzipale reagieren dürfen.

    - Wählen Sie Bestimmte Ressource aus, um die Richtlinie auf eine bestimmte Ressource anzuwenden. Wählen Sie die Ressource im Feld Ressource, für die diese Richtlinie gelten soll, und geben Sie eine Entitäts-ID für die Ressource ein.

- Wählen Sie Alle Ressourcen aus, um die Richtlinie auf alle Ressourcen in Ihrem Richtlinienspeicher anzuwenden.

16. Wählen Sie im Feld Aktionsbereich aus, für welche Aktionen die angegebenen Prinzipale autorisiert werden sollen.

    - Wählen Sie Bestimmte Gruppe von Aktionen aus, um die Richtlinie auf bestimmte Aktionen anzuwenden. Aktivieren Sie die Kontrollkästchen neben den Aktionen im Feld Aktion (en), für die diese Richtlinie gelten soll.
    - Wählen Sie Alle Aktionen aus, um die Richtlinie auf alle Aktionen in Ihrem Richtlinienspeicher anzuwenden.

17. Sehen Sie sich die Richtlinie im Abschnitt Richtlinienvorschau an. Wählen Sie Richtlinienspeicher erstellen aus.

Set up with API Gateway and an identity source

So erstellen Sie einen Richtlinienspeicher mithilfe der Konfigurationsmethode „Mit API Gateway einrichten“ und einer Identitätsquelle

Die API-Gateway-Option schützt APIs mit Richtlinien für verifizierte Berechtigungen, die darauf ausgelegt sind, Autorisierungsentscheidungen von Benutzergruppen oder Rollen zu treffen. Diese Option erstellt einen Richtlinienspeicher zum Testen der Autorisierung mit Identitätsquellengruppen und eine API mit einem Lambda-Autorisierer.

Die Benutzer und ihre Gruppen in einem IdP werden entweder zu Ihren Prinzipalen (ID-Token) oder zu Ihrem Kontext (Zugriffstoken). Die Methoden und Pfade in einer API-Gateway-API werden zu den Aktionen, die Ihre Richtlinien autorisieren. Ihre Anwendung wird zur Ressource. Als Ergebnis dieses Workflows erstellt Verified Permissions einen Richtlinienspeicher, eine Lambda-Funktion und einen API-Lambda-Authorizer. Sie müssen den Lambda-Autorisierer Ihrer API zuweisen, nachdem Sie diesen Workflow abgeschlossen haben.

1. Wählen Sie in der Konsole „Verifizierte Berechtigungen“ die Option Neuen Richtlinienspeicher erstellen aus.
2. Wählen Sie im Abschnitt Startoptionen die Option Mit API Gateway und einer Identitätsquelle einrichten und dann Weiter aus.
3. Wählen Sie im Schritt Ressourcen und Aktionen importieren unter API eine API aus, die als Modell für die Ressourcen und Aktionen Ihres Richtlinienspeichers dienen soll.

a. Wählen Sie aus den in Ihrer API konfigurierten Phasen eine Bereitstellungsphase aus und wählen Sie API importieren aus. Weitere Informationen zu API-Phasen finden Sie im Amazon API Gateway Developer Guide unter Setting up a Stage for a REST API API API API API API.

b. Zeigen Sie eine Vorschau Ihrer Map mit importierten Ressourcen und Aktionen an.

c. Um Ressourcen oder Aktionen zu aktualisieren, ändern Sie Ihre API-Pfade oder Methoden und wählen Sie API importieren aus.

d. Wenn Sie mit Ihrer Auswahl zufrieden sind, wählen Sie Weiter.

4. Wählen Sie unter Identitätsquelle einen Identitätsanbietertyp aus. Sie können einen Amazon Cognito Cognito-Benutzerpool oder einen OpenID Connect (OIDC) IdP-Typ wählen.

5. Wenn Sie sich für Amazon Cognito entschieden haben:

a. Wählen Sie einen Benutzerpool aus, der sich in demselben AWS-Region und AWS-Konto wie in Ihrem Richtlinienspeicher befindet.

b. Wählen Sie den Tokentyp aus, der an die API übergeben werden soll und den Sie zur Autorisierung einreichen möchten. Beide Tokentypen enthalten Benutzergruppen, die Grundlage dieses API-verknüpften Autorisierungsmodells.

c. Unter App-Client-Validierung können Sie den Umfang eines Policy Stores auf eine Teilmenge der Amazon Cognito-App-Clients in einem Benutzerpool mit mehreren Mandanten beschränken. Um zu verlangen, dass sich dieser Benutzer bei einem oder mehreren angegebenen App-Clients in Ihrem Benutzerpool authentifiziert, wählen Sie Nur Token mit erwarteten App-Client-IDs akzeptieren aus. Um jeden Benutzer zu akzeptieren, der sich beim Benutzerpool authentifiziert, wählen Sie App-Client-IDs nicht validieren aus.

d. Wählen Sie Weiter aus.

6. Wenn Sie sich für einen OIDC-Anbieter entschieden haben:

a. Geben Sie im Feld Aussteller-URL die URL Ihres OIDC-Ausstellers ein. Dies ist der Dienstendpunkt, der beispielsweise den Autorisierungsserver, Signaturschlüssel und andere Informationen über Ihren Anbieter bereitstellt. `https://auth.example.com` Ihre Aussteller-URL muss ein OIDC-Discovery-Dokument unter hosten. `/.well-known/openid-configuration`

b. Wählen Sie unter Tokentyp den Typ des OIDC JWT aus, den Ihre Anwendung zur Autorisierung einreichen soll. Weitere Informationen finden Sie unter Arbeiten mit Identitätsquellen in Schemas und Richtlinien.

c. Wählen Sie unter Tokenansprüche aus, wie Sie Benutzerattribute in Ihrem Richtlinienspeicher einrichten möchten. Diese Attribute definieren die Ansprüche, auf die Ihre Richtlinien verweisen können.

   i. Wählen Sie eine Quelle für den Antrag aus.

      A. Um ein Mustertoken bereitzustellen, wählen Sie Aus JWT-Payload extrahieren und fügen Sie die Payload eines JWT Ihres ausgewählten Tokentyps ein. JWTs enthalten einen Header, eine Nutzlast und eine Signatur. Ihr Beispiel-JWT muss dekodiert und nur für Nutzdaten verwendet werden. Um die Payload zu analysieren, wählen Sie Extrahieren aus.

      B. Um Ihren eigenen Satz von Attributen einzugeben, wählen Sie „Ansprüche manuell eingeben“.

   ii. Geben Sie jeden Token-Anspruchsnamen und jeden Anspruchswerttyp ein, den Sie den Attributen des Benutzerprinzipals- oder Aktionskontextes in Ihrem Schema hinzufügen möchten, oder bestätigen Sie diese.

d. Wählen Sie unter Benutzer- und Gruppenansprüche einen Benutzeranspruch für die Identitätsquelle aus. Dabei handelt es sich in der Regel `sub` um einen Anspruch anhand Ihrer ID oder Ihres Zugriffstoken, das die eindeutige Kennung für die zu prüfende Entität enthält. Identitäten des verbundenen OIDC-IdP werden dem Benutzertyp in Ihrem Richtlinienspeicher zugeordnet.

e. Wählen Sie unter Benutzer- und Gruppenansprüche einen Gruppenanspruch für die Identitätsquelle aus. Dabei handelt es sich in der Regel `groups` um einen Anspruch aus Ihrer ID oder Ihrem Zugriffstoken, der eine Liste der Benutzergruppen enthält. Ihr Richtlinienspeicher autorisiert Anfragen auf der Grundlage der Gruppenmitgliedschaft.

f. Geben Sie unter Zielgruppenvalidierung oder Client-IDs die Client-IDs oder Zielgruppen-URLs ein, die Ihr Richtlinienspeicher gegebenenfalls in Autorisierungsanfragen akzeptieren soll. Geben Sie für Zugriffstoken einen Wert für den Zielgruppenanspruch wie ein`https://myapp.example.com`. Geben Sie für ID-Token eine Client-ID wie ein`1example23456789`.

g. Wählen Sie Weiter aus.

7. Wenn Sie sich für Amazon Cognito entschieden haben, fragt Verified Permissions Ihren Benutzerpool nach Gruppen ab. Geben Sie für OIDC-Anbieter Gruppennamen manuell ein. Mit dem Schritt Aktionen Gruppen zuweisen werden Richtlinien für Ihren Richtlinienspeicher erstellt, die es Gruppenmitgliedern ermöglichen, Aktionen auszuführen.

    a. Wählen Sie die Gruppen aus, die Sie in Ihre Richtlinien aufnehmen möchten, oder fügen Sie sie hinzu.

    b. Weisen Sie jeder der ausgewählten Gruppen Aktionen zu.

    c. Wählen Sie Weiter aus.

8. Überprüfen Sie unter App-Integration bereitstellen die Schritte, die Verified Permissions zur Erstellung Ihres Richtlinienspeichers und Lambda-Autorisierers unternimmt.

9. Wenn Sie bereit sind, die neuen Ressourcen zu erstellen, wählen Sie Create and Deploy aus.

10. Lassen Sie den Schritt Status des Richtlinienspeichers in Ihrem Browser geöffnet, um den Fortschritt der Ressourcenerstellung anhand von Verified Permissions zu überwachen.

11. Nach einiger Zeit, in der Regel etwa einer Stunde, oder wenn der Schritt „Lambda-Autorisierung bereitstellen“ Erfolg anzeigt, konfigurieren Sie Ihren Autorisierer.

    Verified Permissions hat eine Lambda-Funktion und einen Lambda-Authorizer in Ihrer API erstellt. Wählen Sie Open API, um zu Ihrer API zu navigieren.

    Informationen zum Zuweisen eines Lambda-Autorisierers finden Sie unter Verwenden von API Gateway Gateway-Lambda-Autorisierern im Amazon API Gateway Gateway-Entwicklerhandbuch.

    a. Navigieren Sie zu Autorisatoren für Ihre API und notieren Sie sich den Namen des Autorisierers, den Verified Permissions erstellt hat.

    b. Navigieren Sie zu Ressourcen und wählen Sie eine Methode der obersten Ebene in Ihrer API aus.

    c. Wählen Sie unter Einstellungen für Methodenanfragen die Option Bearbeiten aus.

    d. Geben Sie als Autorisierer den Namen des Autorisierers ein, den Sie sich zuvor notiert haben.

    e. Erweitern Sie HTTP-Anforderungsheader, geben Sie einen Namen oder ein und wählen Sie **`AUTHORIZATION`** Erforderlich aus.

    f. Stellen Sie die API-Phase bereit.

    g. Speichern Sie Ihre Änderungen.

12. Testen Sie Ihren Autorisierer mit einem Benutzerpool-Token des Tokentyps, den Sie im Schritt Identitätsquelle auswählen ausgewählt haben. Weitere Informationen zur Benutzerpool-Anmeldung und zum Abrufen von Token finden Sie unter Benutzerpool-Authentifizierungsablauf im Amazon Cognito Developer Guide.
13. Testen Sie die Authentifizierung erneut mit einem Benutzerpool-Token im `AUTHORIZATION` Header einer Anfrage an Ihre API.
14. Untersuchen Sie Ihren neuen Richtlinienspeicher. Fügen Sie Richtlinien hinzu und verfeinern Sie sie.

Sample policy store

So erstellen Sie einen Richtlinienspeicher mit der Beispiel-Konfigurationsmethode für den Richtlinienspeicher

1. Wählen Sie im Abschnitt Startoptionen die Option Beispiel für einen Richtlinienspeicher aus.
2. Wählen Sie im Abschnitt Beispielprojekt den Typ der Beispielanwendung Verified Permissions aus, die Sie verwenden möchten.

   - PhotoFlashist eine Beispiel-Webanwendung für Kunden, mit der Benutzer einzelne Fotos und Alben mit Freunden teilen können. Benutzer können detaillierte Berechtigungen dafür festlegen, wer ihre Fotos ansehen, kommentieren und erneut teilen darf. Kontoinhaber können auch Gruppen von Freunden erstellen und Fotos in Alben organisieren.
   - DigitalPetStore ist eine Beispielanwendung, mit der sich jeder registrieren und Kunde werden kann. Kunden können Haustiere zum Verkauf hinzufügen, nach Haustieren suchen und Bestellungen aufgeben. Kunden, die ein Haustier hinzugefügt haben, werden als Tierbesitzer registriert. Tierbesitzer können die Angaben zum Haustier aktualisieren, ein Tierbild hochladen oder den Tiereintrag löschen. Kunden, die eine Bestellung aufgegeben haben, werden als Inhaber der Bestellung registriert. Bestellungsinhaber können Einzelheiten zur Bestellung abrufen oder sie stornieren. Manager von Tierhandlungen haben Administratorzugriff.

     > **Note**
     >
     > Der DigitalPet Musterrichtlinienspeicher Store enthält keine Richtlinienvorlagen. Der Richtlinienspeicher PhotoFlashund der TinyTodoBeispielrichtlinienspeicher enthalten Richtlinienvorlagen.

- TinyTodoist eine Beispielanwendung, mit der Benutzer Aufgaben und Aufgabenlisten erstellen können. Listenbesitzer können ihre Listen verwalten und teilen und angeben, wer ihre Listen ansehen oder bearbeiten kann.

3. Basierend auf dem ausgewählten Beispielprojekt wird automatisch ein Namespace für das Schema Ihres Beispielrichtlinienspeichers generiert.
4. Wählen Sie Richtlinienspeicher erstellen aus.

   Ihr Richtlinienspeicher wird mit Richtlinien und einem Schema für den von Ihnen ausgewählten Beispiel-Richtlinienspeicher erstellt. Weitere Informationen zu vorlagenverknüpften Richtlinien, die Sie für die Beispielrichtlinienspeicher erstellen können, finden Sie unter. Beispiel für vorlagenverknüpfte Richtlinien für Beispielrichtlinienspeicher für Verified Permissions

Empty policy store

So erstellen Sie einen Richtlinienspeicher mit der Konfigurationsmethode „Richtlinienspeicher leeren“

1. Wählen Sie im Abschnitt Startoptionen die Option Richtlinienspeicher leeren aus.
2. Wählen Sie Richtlinienspeicher erstellen aus.

Ein leerer Richtlinienspeicher wird ohne Schema erstellt, was bedeutet, dass Richtlinien nicht validiert werden. Weitere Informationen zur Aktualisierung des Schemas für Ihren Richtlinienspeicher finden Sie unterSpeicherschema für Richtlinien von Amazon Verified Permissions.

Weitere Informationen zum Erstellen von Richtlinien für Ihren Richtlinienspeicher finden Sie unter Statische Richtlinien für Amazon Verified Permissions erstellen undErstellen von vorlagenverknüpften Richtlinien.

AWS CLI

Um einen leeren Richtlinienspeicher zu erstellen, verwenden Sie den AWS CLI.

Sie können einen Richtlinienspeicher erstellen, indem Sie den `create-policy-store` Vorgang verwenden.

> **Note**
>
> Ein Richtlinienspeicher, den Sie mithilfe von erstellen, AWS CLI ist leer.
>
> - Informationen zum Hinzufügen eines Schemas finden Sie unterSpeicherschema für Richtlinien von Amazon Verified Permissions.
> - Informationen zum Hinzufügen von Richtlinien finden Sie unterStatische Richtlinien für Amazon Verified Permissions erstellen.
> - Informationen zum Hinzufügen von Richtlinienvorlagen finden Sie unterErstellen von Richtlinienvorlagen.

```
$ 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

Sie können mithilfe der `CreatePolicyStore` API einen Richtlinienspeicher erstellen. Weitere Informationen finden Sie unter CreatePolicyShop im Referenzhandbuch zur Amazon Verified Permissions API.

API-verknüpfte Richtlinienspeicher

Wenn Sie in der Amazon Verified Permissions-Konsole einen neuen Richtlinienspeicher erstellen, können Sie die Option Mit API Gateway einrichten und eine Identitätsquelle auswählen. Mit dieser Option erstellen Sie einen API-verknüpften Richtlinienspeicher, ein Autorisierungsmodell für Anwendungen, die sich bei Amazon Cognito Cognito-Benutzerpools oder einem OIDC-Identitätsanbieter (IdP) authentifizieren und Daten von Amazon API Gateway Gateway-APIs abrufen. Um zu beginnen, sehen Sie sich Erstellen Sie einen Richtlinienspeicher mit einer verbundenen API und einem Identitätsanbieter an.

Themen

- Wie verifizierte Berechtigungen API-Anfragen autorisieren
- Hinzufügen einer attributebasierten Zugriffskontrolle (ABAC)
- Überlegungen zu API-verknüpften Policy-Stores
- Fehlerbehebung bei API-verknüpften Policyspeichern

> **Important**
>
> Richtlinienspeicher, die Sie mit der Option „Mit API Gateway einrichten“ und einer Identitätsquelle in der Konsole „Verifizierte Berechtigungen“ erstellen, sind nicht für die sofortige Bereitstellung in der Produktion vorgesehen. Stellen Sie mit Ihrem ersten Richtlinienspeicher Ihr Autorisierungsmodell fertig und exportieren Sie die Ressourcen des Richtlinienspeichers in CloudFormation. Stellen Sie verifizierte Berechtigungen programmgesteuert mit dem AWS Cloud Development Kit (CDK) für die Produktion bereit. Weitere Informationen finden Sie unter Übergang zur Produktion mit AWS CloudFormation.

In einem Richtlinienspeicher, der mit einer API und einer Identitätsquelle verknüpft ist, präsentiert Ihre Anwendung ein Benutzerpool-Token in einem Autorisierungsheader, wenn sie eine Anfrage an die API stellt. Die Identitätsquelle Ihres Richtlinienspeichers ermöglicht die Tokenvalidierung für verifizierte Berechtigungen. Das Token bildet die `principal` Eingangs-Autorisierungsanfragen mit der IsAuthorizedWithTokenAPI. Verified Permissions erstellt Richtlinien rund um die Gruppenzugehörigkeit Ihrer Benutzer, wie sie in einem Gruppenanspruch in Form von Identitäts- (ID) und Zugriffstoken dargestellt werden, beispielsweise `cognito:groups` für Benutzerpools. Ihre API verarbeitet das Token aus Ihrer Anwendung in einem Lambda-Autorisierer und leitet es zur Autorisierungsentscheidung an Verified Permissions weiter. Wenn Ihre API die Autorisierungsentscheidung vom Lambda-Autorisierer erhält, leitet sie die Anfrage an Ihre Datenquelle weiter oder lehnt die Anfrage ab.

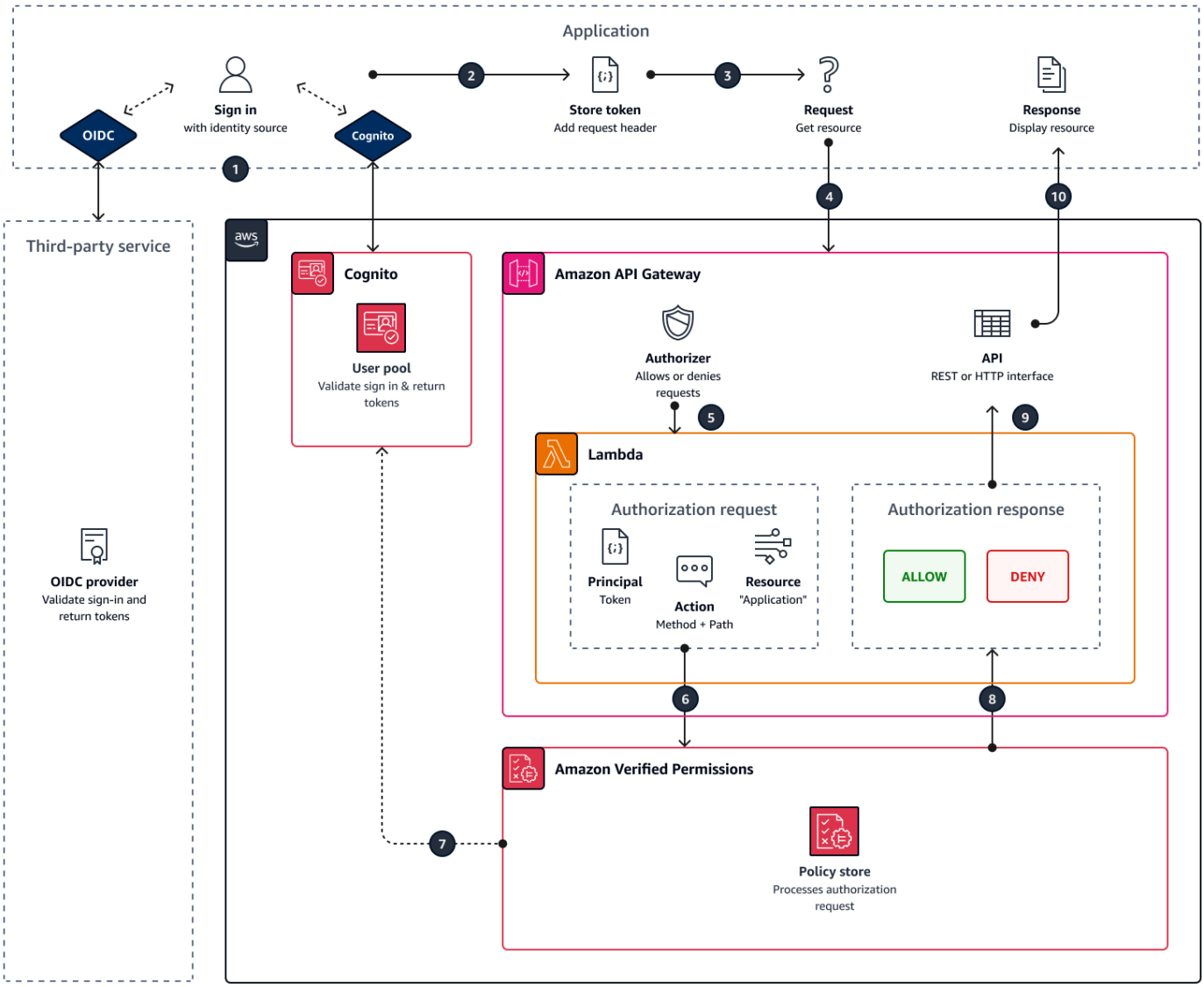
Komponenten der Identitätsquelle und der API-Gateway-Autorisierung mit verifizierten Berechtigungen

- Ein Amazon Cognito Cognito-Benutzerpool oder OIDC-IdP, der Benutzer authentifiziert und gruppiert. Benutzertoken füllen die Gruppenmitgliedschaft und den Prinzipal oder Kontext, den Verified Permissions in Ihrem Richtlinienspeicher auswertet.

- Eine API-Gateway-REST-API. Verified Permissions definiert beispielsweise Aktionen anhand von API-Pfaden und API-Methoden`MyAPI::Action::get /photo`.
- Eine Lambda-Funktion und ein Lambda-Authorizer für Ihre API. Die Lambda-Funktion nimmt Bearer-Token aus Ihrem Benutzerpool entgegen, fordert die Autorisierung von Verified Permissions an und gibt eine Entscheidung an API Gateway zurück. Der Workflow „Mit Cognito und API Gateway einrichten“ erstellt diesen Lambda-Authorizer automatisch für Sie.
- Ein Richtlinienspeicher für verifizierte Berechtigungen. Die Identitätsquelle des Richtlinienspeichers ist Ihr Benutzerpool. Das Policy Store-Schema spiegelt die Konfiguration Ihrer API wider, und die Richtlinien verknüpfen Benutzergruppen mit zulässigen API-Aktionen.
- Eine Anwendung, die Benutzer bei Ihrem IdP authentifiziert und Tokens an API-Anfragen anhängt.

Wie verifizierte Berechtigungen API-Anfragen autorisieren

Wenn Sie einen neuen Richtlinienspeicher erstellen und die Option Mit Cognito und API Gateway einrichten auswählen, erstellt Verified Permissions ein Richtlinienspeicherschema und Richtlinien. Das Schema und die Richtlinien spiegeln die API-Aktionen und die Benutzerpoolgruppen wider, die Sie zur Durchführung der Aktionen autorisieren möchten. Verified Permissions erstellt auch die Lambda-Funktion und den Autorisierer. Sie müssen den neuen Autorisierer für eine Methode in Ihrer API konfigurieren.



1. Ihr Benutzer meldet sich mit Ihrer Anwendung über Amazon Cognito oder einen anderen OIDC-IdP an. Der IdP gibt ID- und Zugriffstoken mit den Benutzerinformationen aus.
2. Ihre Anwendung speichert die JWTs. Weitere Informationen finden Sie unter Verwenden von Token mit Benutzerpools im Amazon Cognito Developer Guide.
3. Ihr Benutzer fordert Daten an, die Ihre Anwendung von einer externen API abrufen muss.
4. Ihre Anwendung fordert Daten von einer REST-API in API Gateway an. Sie hängt eine ID oder ein Zugriffstoken als Anforderungsheader an.
5. Wenn Ihre API über einen Cache für die Autorisierungsentscheidung verfügt, gibt sie die vorherige Antwort zurück. Wenn das Caching deaktiviert ist oder die API keinen aktuellen Cache hat, übergibt API Gateway die Anforderungsparameter an einen tokenbasierten Lambda-Authorizer.

6. Die Lambda-Funktion sendet mit der IsAuthorizedWithTokenAPI eine Autorisierungsanfrage an einen Richtlinienspeicher für verifizierte Berechtigungen. Die Lambda-Funktion übergibt die Elemente einer Autorisierungsentscheidung:
   a. Das Token des Benutzers als Principal.
   b. Die API-Methode kombiniert mit dem API-Pfad`GetPhoto`, z. B. als Aktion.
   c. Der Begriff `Application` als Ressource.
7. Verified Permissions validiert das Token. Weitere Informationen zur Validierung von Amazon Cognito-Token finden Sie unter Authorization with Amazon Verified Permissions im Amazon Cognito Developer Guide.
8. Verified Permissions bewertet die Autorisierungsanfrage anhand der Richtlinien in Ihrem Richtlinienspeicher und gibt eine Autorisierungsentscheidung zurück.
9. Der Lambda-Autorisierer gibt eine `Allow` `Deny` Oder-Antwort an API Gateway zurück.
10. Die API gibt Daten oder eine `ACCESS_DENIED` Antwort an Ihre Anwendung zurück. Ihre Anwendung verarbeitet die Ergebnisse der API-Anfrage und zeigt sie an.

Hinzufügen einer attributebasierten Zugriffskontrolle (ABAC)

Eine typische Authentifizierungssitzung mit einem IdP gibt ID- und Zugriffstoken zurück. Sie können jeden dieser Tokentypen als Bearer-Token in Anwendungsanfragen an Ihre API übergeben. Je nachdem, welche Optionen Sie bei der Erstellung Ihres Richtlinienspeichers ausgewählt haben, erwartet Verified Permissions einen der beiden Token-Typen. Beide Typen enthalten Informationen über die Gruppenmitgliedschaft des Benutzers. Weitere Informationen zu Tokentypen in Amazon Cognito finden Sie unter Verwenden von Token mit Benutzerpools im Amazon Cognito Developer Guide.

Nachdem Sie einen Richtlinienspeicher erstellt haben, können Sie Richtlinien hinzufügen und erweitern. Beispielsweise können Sie Ihren Richtlinien neue Gruppen hinzufügen, wenn Sie sie Ihrem Benutzerpool hinzufügen. Da Ihr Richtlinienspeicher bereits weiß, wie Ihr Benutzerpool Gruppen in Tokens präsentiert, können Sie eine Reihe von Aktionen für jede neue Gruppe mit einer neuen Richtlinie zulassen.

Möglicherweise möchten Sie auch das gruppenbasierte Modell der Richtlinienbewertung um ein genaueres Modell erweitern, das auf Benutzereigenschaften basiert. Benutzerpool-Token enthalten zusätzliche Benutzerinformationen, die zu Autorisierungsentscheidungen beitragen können.

ID-Token

ID-Token stellen die Attribute eines Benutzers dar und bieten ein Höchstmaß an detaillierter Zugriffskontrolle. Um E-Mail-Adressen, Telefonnummern oder benutzerdefinierte Attribute wie Abteilung und Manager auszuwerten, werten Sie das ID-Token aus.

Zugriffstoken

Zugriffstoken stellen die Berechtigungen eines Benutzers mit OAuth 2.0-Bereichen dar. Um eine Autorisierungsebene hinzuzufügen oder Anfragen für zusätzliche Ressourcen einzurichten, bewerten Sie das Zugriffstoken. Sie können beispielsweise überprüfen, ob ein Benutzer zu den entsprechenden Gruppen gehört und über einen solchen Geltungsbereich verfügt`PetStore.read`, der generell den Zugriff auf die API autorisiert. Benutzerpools können Tokens mit Ressourcenservern und mit Token-Anpassungen zur Laufzeit benutzerdefinierte Bereiche hinzufügen.

Sehen Sie sich Arbeiten mit Identitätsquellen in Schemas und Richtlinien zum Beispiel Richtlinien an, die Ansprüche in ID- und Zugriffstoken verarbeiten.

Überlegungen zu API-verknüpften Policy-Stores

Wenn Sie in der Konsole „Verified Permissions“ einen API-verknüpften Richtlinienspeicher erstellen, erstellen Sie einen Test für eine spätere Produktionsbereitstellung. Bevor Sie zur Produktion übergehen, richten Sie eine feste Konfiguration für Ihre API und Ihren Benutzerpool ein. Berücksichtigen Sie die folgenden Faktoren:

API Gateway speichert Antworten im Cache

In API-verknüpften Richtlinienspeichern erstellt Verified Permissions einen Lambda-Autorisierer mit einer Autorisierungs-Caching-TTL von 120 Sekunden. Sie können diesen Wert anpassen oder das Caching in Ihrem Authorizer deaktivieren. In einem Autorisierer mit aktiviertem Caching gibt Ihr Autorisierer jedes Mal dieselbe Antwort zurück, bis die TTL abläuft. Dadurch kann die tatsächliche Lebensdauer von Benutzerpool-Token um eine Dauer verlängert werden, die der Caching-TTL der angeforderten Phase entspricht.

Amazon Cognito Cognito-Gruppen können wiederverwendet werden

Amazon Verified Permissions bestimmt die Gruppenmitgliedschaft von Benutzern aus dem Benutzerpool anhand des `cognito:groups` Antrags in der ID oder dem Zugriffstoken eines Benutzers. Der Wert dieses Anspruchs besteht aus einer Reihe von benutzerfreundlichen Namen

der Benutzerpoolgruppen, denen der Benutzer angehört. Sie können Benutzerpoolgruppen keinen eindeutigen Bezeichner zuordnen.

Benutzerpoolgruppen, die Sie löschen und mit demselben Namen neu erstellen, werden in Ihrem Richtlinienspeicher als dieselbe Gruppe angezeigt. Wenn Sie eine Gruppe aus einem Benutzerpool löschen, löschen Sie alle Verweise auf die Gruppe aus Ihrem Richtlinienspeicher.

Von der API abgeleiteter Namespace und Schema sind point-in-time

Verified Permissions erfasst Ihre API zu einem bestimmten Zeitpunkt: Ihre API wird nur abgefragt, wenn Sie Ihren Richtlinienspeicher erstellen. Wenn sich das Schema oder der Name Ihrer API ändert, müssen Sie Ihren Richtlinienspeicher und Ihren Lambda-Autorisierer aktualisieren oder einen neuen API-verknüpften Richtlinienspeicher erstellen. Verified Permissions leitet den Namespace für den Richtlinienspeicher vom Namen Ihrer API ab.

Lambda Lambda-Funktion hat keine VPC-Konfiguration

Die Lambda-Funktion, die Verified Permissions für Ihren API-Authorizer erstellt, ist nicht mit einer VPC verbunden. Standardmäßig. APIs, deren Netzwerkzugriff auf private VPCs beschränkt ist, können nicht mit der Lambda-Funktion kommunizieren, die Zugriffsanforderungen mit verifizierten Berechtigungen autorisiert.

Verified Permissions stellt Autorisierungsressourcen bereit in CloudFormation

Um einen API-verknüpften Richtlinienspeicher zu erstellen, müssen Sie sich bei der Verified Permissions-Konsole mit einem AWS Prinzipal mit hohen Rechten anmelden. Dieser Benutzer stellt einen AWS CloudFormation Stapel bereit, der Ressourcen aus mehreren zusammensetzt. AWS-Services Dieser Principal muss über die Berechtigung verfügen, Ressourcen in Verified Permissions IAM, Lambda und API Gateway hinzuzufügen und zu ändern. Es hat sich bewährt, diese Anmeldeinformationen nicht mit anderen Administratoren in Ihrer Organisation zu teilen.

Einen Überblick über die Ressourcen, die Verified Permissions erstellt, finden Übergang zur Produktion mit AWS CloudFormation Sie unter.

Übergang zur Produktion mit AWS CloudFormation

API-verknüpfte Richtlinienspeicher sind eine Möglichkeit, schnell ein Autorisierungsmodell für eine API Gateway zu erstellen. Sie sind so konzipiert, dass sie als Testumgebung für die Autorisierungskomponente Ihrer Anwendung dienen. Nachdem Sie Ihren Testrichtlinienspeicher erstellt haben, sollten Sie Zeit damit verbringen, die Richtlinien, das Schema und den Lambda-Authorizer zu verfeinern.

Möglicherweise passen Sie die Architektur Ihrer API an, sodass entsprechende Anpassungen an Ihrem Richtlinienspeicher-Schema und Ihren Richtlinien erforderlich sind. API-verknüpfte Richtlinienspeicher aktualisieren ihr Schema nicht automatisch über die API-Architektur — Verified Permissions fragt die API nur ab, wenn Sie einen Richtlinienspeicher erstellen. Wenn sich Ihre API ausreichend ändert, müssen Sie den Vorgang möglicherweise mit einem neuen Richtlinienspeicher wiederholen.

Wenn Ihre Anwendung und Ihr Autorisierungsmodell bereit für die Bereitstellung in der Produktion sind, integrieren Sie den API-verknüpften Richtlinienspeicher, den Sie entwickelt haben, in Ihre Automatisierungsprozesse. Als bewährte Methode empfehlen wir, das Richtlinienspeicherschema und die Richtlinien in eine AWS CloudFormation Vorlage zu exportieren, die Sie in anderen AWS-Konten Bereichen bereitstellen können. AWS-Regionen

Die Ergebnisse des API-verknüpften Richtlinienspeicher-Prozesses sind ein erster Richtlinienspeicher und ein Lambda-Autorisierer. Der Lambda-Autorisierer hat mehrere abhängige Ressourcen. Verified Permissions stellt diese Ressourcen in einem automatisch generierten Stack bereit. CloudFormation Für die Bereitstellung in der Produktion müssen Sie den Richtlinienspeicher und die Lambda-Autorisierungsressourcen in einer Vorlage zusammenfassen. Ein API-verknüpfter Richtlinienspeicher besteht aus den folgenden Ressourcen:

1. AWS::VerifiedPermissions::PolicyStore: Kopieren Sie Ihr Schema in das `SchemaDefinition` Objekt. "Escape-Zeichen als\".
2. AWS::VerifiedPermissions::IdentitySource: Kopieren Sie Werte aus der Ausgabe von GetIdentitySource aus Ihrem Testrichtlinienspeicher und ändern Sie sie nach Bedarf.
3. Eine oder mehrere der folgenden AWS::VerifiedPermissions::PolicyOptionen: Kopieren Sie Ihre Richtlinienerklärung in das `Definition` Objekt. "Escape-Zeichen als\".
4. AWS: :Lambda: :Function,: AWS:: :Role,IAM::: :Policy,AWS:IAM:: :Authorizer,AWS ApiGateway AWS::Lambda::Permission: Kopieren Sie die Vorlage aus dem Template-Tab des Stacks, der Verified Permissions bereitgestellt hat, als Sie Ihren Richtlinienspeicher erstellt haben.

Die folgende Vorlage ist ein Beispiel für einen Richtlinienspeicher. Sie können die Lambda-Authorizer-Ressourcen aus Ihrem vorhandenen Stack an diese Vorlage anhängen.

```
{
    "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"
            ]
        }
    }
}
```

Fehlerbehebung bei API-verknüpften Policyspeichern

Verwenden Sie die Informationen hier, um häufig auftretende Probleme beim Erstellen von mit der Amazon Verified Permissions API verknüpften Policy Stores zu diagnostizieren und zu beheben.

Themen

- Ich habe meine Richtlinie aktualisiert, aber die Autorisierungsentscheidung hat sich nicht geändert
- Ich habe den Lambda-Autorisierer an meine API angehängt, aber er generiert keine Autorisierungsanfragen
- Ich habe eine unerwartete Autorisierungsentscheidung erhalten und möchte die Autorisierungslogik überprüfen
- Ich möchte Logs von meinem Lambda-Authorizer finden
- Mein Lambda-Autorisierer existiert nicht
- Meine API befindet sich in einer privaten VPC und kann den Authorizer nicht aufrufen
- Ich möchte zusätzliche Benutzerattribute in meinem Autorisierungsmodell verarbeiten

- Ich möchte neue Aktionen, Aktionskontext-Attribute oder Ressourcenattribute hinzufügen

Ich habe meine Richtlinie aktualisiert, aber die Autorisierungsentscheidung hat sich nicht geändert

Standardmäßig konfiguriert Verified Permissions den Lambda-Autorisierer so, dass Autorisierungsentscheidungen 120 Sekunden lang zwischengespeichert werden. Versuchen Sie es nach zwei Minuten erneut, oder deaktivieren Sie den Cache auf Ihrem Authorizer. Weitere Informationen finden Sie unter Aktivieren von API-Caching zur Verbesserung der Reaktionsfähigkeit im Amazon API Gateway Developer Guide.

Ich habe den Lambda-Autorisierer an meine API angehängt, aber er generiert keine Autorisierungsanfragen

Um mit der Bearbeitung von Anfragen zu beginnen, müssen Sie die API-Stufe bereitstellen, an die Sie Ihren Autorisierer angehängt haben. Weitere Informationen finden Sie unter Bereitstellen einer REST-API im Amazon API Gateway Developer Guide.

Ich habe eine unerwartete Autorisierungsentscheidung erhalten und möchte die Autorisierungslogik überprüfen

Der API-verknüpfte Policy Store-Prozess erstellt eine Lambda-Funktion für Ihren Autorisierer. Verified Permissions integriert die Logik Ihrer Autorisierungsentscheidungen automatisch in die Autorisierungsfunktion. Nachdem Sie Ihren Richtlinienspeicher erstellt haben, können Sie zurückgehen, um die Logik in der Funktion zu überprüfen und zu aktualisieren.

Um Ihre Lambda-Funktion von der AWS CloudFormation Konsole aus zu finden, klicken Sie auf der Übersichtsseite Ihres neuen Policy-Stores auf die Schaltfläche Bereitstellung überprüfen.

Sie können Ihre Funktion auch in der AWS Lambda Konsole finden. Navigieren Sie zur Konsole in Ihrem Richtlinienspeicher und suchen Sie nach einem Funktionsnamen mit dem Präfix`AVPAuthorizerLambda`. AWS-Region Wenn Sie mehr als einen API-verknüpften Richtlinienspeicher erstellt haben, verwenden Sie die Uhrzeit der letzten Änderung Ihrer Funktionen, um sie mit der Erstellung des Richtlinienspeichers zu korrelieren.

Ich möchte Logs von meinem Lambda-Authorizer finden

Lambda-Funktionen sammeln Metriken und protokollieren ihre Aufrufergebnisse in Amazon. CloudWatch Um Ihre Logs zu überprüfen, suchen Sie Ihre Funktion in der Lambda-Konsole und

wählen Sie die Registerkarte Überwachen. Wählen Sie CloudWatch Protokolle anzeigen aus und überprüfen Sie die Einträge in der Protokollgruppe.

Weitere Informationen zu Lambda-Funktionsprotokollen finden Sie unter Using Amazon CloudWatch Logs with AWS Lambda im AWS Lambda Developer Guide.

Mein Lambda-Autorisierer existiert nicht

Nachdem Sie die Einrichtung eines API-verknüpften Richtlinienspeichers abgeschlossen haben, müssen Sie den Lambda-Authorizer an Ihre API anhängen. Wenn Sie Ihren Autorisierer in der API Gateway Gateway-Konsole nicht finden können, sind die zusätzlichen Ressourcen für Ihren Richtlinienspeicher möglicherweise ausgefallen oder noch nicht bereitgestellt. API-verknüpfte Policy-Stores stellen diese Ressourcen in einem Stapel bereit. AWS CloudFormation

Unter Verifizierte Berechtigungen wird am Ende des Erstellungsvorgangs ein Link mit der Bezeichnung Bereitstellung prüfen angezeigt. Wenn Sie diesen Bildschirm bereits verlassen haben, rufen Sie die CloudFormation Konsole auf und suchen Sie in den letzten Stacks nach einem Namen, dem das Präfix vorangestellt ist. `AVPAuthorizer-<policy store ID>` CloudFormation bietet wertvolle Informationen zur Fehlerbehebung in der Ausgabe einer Stack-Bereitstellung.

Hilfe zur Fehlerbehebung bei CloudFormation Stacks finden Sie unter Problembehandlung CloudFormation im AWS CloudFormation Benutzerhandbuch.

Meine API befindet sich in einer privaten VPC und kann den Authorizer nicht aufrufen

Verified Permissions unterstützt keinen Zugriff auf Lambda-Autorisierer über VPC-Endpunkte. Sie müssen einen Netzwerkpfad zwischen Ihrer API und der Lambda-Funktion öffnen, die als Autorisierer dient.

Ich möchte zusätzliche Benutzerattribute in meinem Autorisierungsmodell verarbeiten

Der API-verknüpfte Policy-Store-Prozess leitet Richtlinien für verifizierte Berechtigungen aus dem Anspruch der Gruppe in Benutzer-Tokens ab. Um Ihr Autorisierungsmodell zu aktualisieren und zusätzliche Benutzerattribute zu berücksichtigen, integrieren Sie diese Attribute in Ihre Richtlinien.

Sie können viele Ansprüche in ID- und Zugriffstoken aus Amazon Cognito Cognito-Benutzerpools den Richtlinienerklärungen für verifizierte Berechtigungen zuordnen. Beispielsweise haben die meisten Benutzer einen `email` Anspruch in ihrem ID-Token. Weitere Informationen zum Hinzufügen von Ansprüchen aus Ihrer Identitätsquelle zu Richtlinien finden Sie unterArbeiten mit Identitätsquellen in Schemas und Richtlinien.

Ich möchte neue Aktionen, Aktionskontext-Attribute oder Ressourcenattribute hinzufügen

Ein API-verknüpfter Richtlinienspeicher und der Lambda-Autorisierer, den er erstellt, sind eine Ressource. point-in-time Sie geben den Status Ihrer API zum Zeitpunkt der Erstellung wieder. Das Policy Store-Schema weist Aktionen weder Kontext-Attribute noch Attribute oder übergeordnete Elemente der `Application` Standardressource zu.

Wenn Sie Ihrer API Aktionen — Pfade und Methoden — hinzufügen, müssen Sie Ihren Richtlinienspeicher aktualisieren, damit er über die neuen Aktionen informiert ist. Sie müssen auch Ihren Lambda-Autorisierer aktualisieren, um Autorisierungsanfragen für die neuen Aktionen zu verarbeiten. Sie können mit einem neuen Richtlinienspeicher erneut beginnen oder Ihren vorhandenen Richtlinienspeicher aktualisieren.

Suchen Sie nach Ihrer Funktion, um Ihren vorhandenen Richtlinienspeicher zu aktualisieren. Untersuchen Sie die Logik in der automatisch generierten Funktion und aktualisieren Sie sie, um die neuen Aktionen, Attribute oder den Kontext zu verarbeiten. Bearbeiten Sie dann Ihr Schema so, dass es die neuen Aktionen und Attribute enthält.

Wechseln der Richtlinienspeicher für verifizierte Berechtigungen

AWS Management Console

So wechseln Sie Richtlinienspeicher oder erstellen zusätzliche Richtlinienspeicher

1. Öffnen Sie die Konsole Verified Permissions unter https://console.aws.amazon.com/verifiedpermissions/. Wählen Sie Ihren Richtlinienspeicher aus.
2. Wählen Sie im Navigationsbereich auf der linken Seite neben Aktueller Richtlinienspeicher die Option Wechseln aus.
3. Sie können zwischen vorhandenen Richtlinienspeichern wechseln oder zusätzliche Richtlinienspeicher erstellen.

   - Um Richtlinienspeicher zu wechseln, wählen Sie die Richtlinienspeicher-ID des Richtlinienspeichers aus, zu dem gewechselt werden soll.
   - Um einen neuen Richtlinienspeicher zu erstellen, wählen Sie Neuen Richtlinienspeicher erstellen aus. Folgen Sie den Anweisungen in Richtlinienspeicher für verifizierte Berechtigungen erstellen.

AWS CLI

So wechseln Sie Richtlinienspeicher oder erstellen zusätzliche Richtlinienspeicher

Der verwaltet AWS CLI keinen „Standard“-Richtlinienspeicher. Stattdessen verwenden die meisten AWS CLI Befehle die , `--policy-store-id` um anzugeben, welcher Richtlinienspeicher für jeden Befehl verwendet werden soll.

Verwenden Sie den create-policy-store Befehl , um einen neuen Richtlinienspeicher zu erstellen.

Löschen von Richtlinienspeichern für verifizierte Berechtigungen

AWS Management Console

So löschen Sie einen Richtlinienspeicher

1. Öffnen Sie die Verified Permissions-Konsole unter https://console.aws.amazon.com/verifiedpermissions/. Wählen Sie Ihren Richtlinienspeicher aus.
2. Wählen Sie im Navigationsbereich auf der linken Seite Settings aus.
3. Wählen Sie Diesen Richtlinienspeicher löschen aus.
4. Geben Sie `delete` in das Textfeld ein und wählen Sie Löschen aus.

AWS CLI

So löschen Sie einen Richtlinienspeicher

Sie können einen Richtlinienspeicher mithilfe der -`delete-policy-store`Operation löschen.

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

Dieser Befehl erzeugt keine Ausgabe, wenn er erfolgreich ist.

Speicherschema für Richtlinien von Amazon Verified Permissions

Ein Schema ist eine Deklaration der Struktur der von Ihrer Anwendung unterstützten Entitätstypen und der Aktionen, die Ihre Anwendung in Autorisierungsanfragen bereitstellen kann.

Weitere Informationen finden Sie unter Cedar-Schemaformat im Cedar Policy Language Reference Guide.

> **Note**
>
> Die Verwendung von Schemas in Verified Permissions ist optional, wird aber für Produktionssoftware dringend empfohlen. Wenn Sie eine neue Richtlinie erstellen, kann Verified Permissions das Schema verwenden, um die Entitäten und Attribute zu überprüfen, auf die im Bereich und in den Bedingungen verwiesen wird, um Tippfehler und Fehler in Richtlinien zu vermeiden, die zu verwirrendem Systemverhalten führen können. Wenn Sie die Richtlinienvalidierung aktivieren, müssen alle neuen Richtlinien dem Schema entsprechen.

AWS Management Console

Ein Schema erstellen

1. Öffnen Sie die Konsole „Verified Permissions“ unter https://console.aws.amazon.com/verifiedpermissions/. Wählen Sie Ihren Richtlinienspeicher aus.
2. Wählen Sie im Navigationsbereich auf der linken Seite Schema aus.
3. Wählen Sie Create schema (Schema erstellen) aus.

AWS CLI

Um ein neues Schema einzureichen oder ein vorhandenes Schema zu überschreiben, verwenden Sie den AWS CLI.

Sie können einen Richtlinienspeicher erstellen, indem Sie einen AWS CLI Befehl ausführen, der dem folgenden Beispiel ähnelt.

Stellen Sie sich ein Schema vor, das den folgenden Cedar-Inhalt enthält:

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

Sie müssen die JSON-Datei zunächst in eine einzeilige Zeichenfolge umwandeln und ihr eine Deklaration ihres Datentyps voranstellen:`cedarJson`. Im folgenden Beispiel wird der folgende Inhalt einer `schema.json` Datei verwendet, die die Escape-Version des JSON-Schemas enthält.

> **Note**
>
> Das Beispiel hier ist aus Gründen der Lesbarkeit mit einem Zeilenumbruch versehen. Sie müssen die gesamte Datei in einer einzigen Zeile haben, damit der Befehl sie akzeptiert.

```
{"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

Sie können mithilfe der `PutSchema` API einen Richtlinienspeicher erstellen. Weitere Informationen finden Sie PutSchemaim Referenzhandbuch zur Amazon Verified Permissions API.

Bearbeiten von Schemas im visuellen Modus

Wenn Sie in der Konsole für verifizierte Berechtigungen die Option Schema auswählen, werden im visuellen Modus die Entitätstypen und Aktionen angezeigt, aus denen Ihr Schema besteht. In dieser Ansicht auf oberster Ebene oder in den Details einer Entität können Sie Schema bearbeiten auswählen, um mit der Aktualisierung Ihres Schemas zu beginnen. Der visuelle Modus ist bei einigen Schemaformaten wie verschachtelten Datensätzen nicht verfügbar.

Der visuelle Schema-Editor beginnt mit einer Reihe von Diagrammen, die die Beziehungen zwischen den Entitäten in Ihrem Schema veranschaulichen. Wählen Sie Erweitern, um Ihre Ansicht der Entitätsbeziehungen Ihres Schemas zu maximieren.

Diagramm der Aktionen

In der Diagrammansicht Aktionen werden die Typen von Prinzipalen aufgeführt, die Sie in Ihrem Richtlinienspeicher konfiguriert haben, die Aktionen, für die sie berechtigt sind, und die Ressourcen, für die sie Aktionen ausführen können. Die Linien zwischen den Entitäten geben an, dass Sie eine Richtlinie erstellen können, die es einem Prinzipal ermöglicht, eine Aktion für eine Ressource durchzuführen. Wenn Ihr Aktionsdiagramm keine Beziehung zwischen zwei Entitäten anzeigt, müssen Sie diese Beziehung zwischen ihnen herstellen, bevor Sie sie in Richtlinien zulassen oder verweigern können. Wählen Sie eine Entität aus, um eine Übersicht über die Eigenschaften zu erhalten, und führen Sie einen Drilldown durch, um alle Details anzuzeigen. Wählen Sie Nach [Aktion | Ressourcentyp | Prinzipaltyp] filtern, um eine Entität in einer Ansicht mit nur ihren eigenen Verbindungen anzuzeigen.

Diagramm der Entitätstypen

Das Diagramm mit den Entitätstypen konzentriert sich auf die Beziehungen zwischen Prinzipalen und Ressourcen. Wenn Sie die komplexen verschachtelten Elternbeziehungen in Ihrem Schema verstehen möchten, sehen Sie sich dieses Diagramm an. Zeigen Sie mit der Maus auf eine Entität, um sich die übergeordneten Beziehungen anzusehen, die sie besitzt.

Unter den Diagrammen befinden sich Listenansichten der Entitätstypen und Aktionen in Ihrem Schema. Die Listenansicht ist nützlich, wenn Sie sofort die Details einer bestimmten Aktion oder eines bestimmten Entitätstyps anzeigen möchten. Wählen Sie eine Entität aus, um Details anzuzeigen.

Um ein Schema für verifizierte Berechtigungen im visuellen Modus zu bearbeiten

1. Öffnen Sie die Konsole Verified Permissions unter https://console.aws.amazon.com/verifiedpermissions/. Wählen Sie Ihren Richtlinienspeicher aus.
2. Wählen Sie im Navigationsbereich auf der linken Seite Schema aus.
3. Wählen Sie den Visuellen Modus. Überprüfen Sie die Entity-Relationship-Diagramme und planen Sie die Änderungen, die Sie an Ihrem Schema vornehmen möchten. Sie können optional nach einer Entität filtern, um deren individuelle Verbindungen zu anderen Entitäten zu untersuchen.
4. Wählen Sie Edit schema (Schema bearbeiten).
5. Geben Sie im Abschnitt Details einen Namespace für Ihr Schema ein.
6. Wählen Sie im Abschnitt Entitätstypen die Option Neuen Entitätstyp hinzufügen aus.
7. Geben Sie den Namen der Entität ein.
8. (Optional) Wählen Sie Übergeordnete Entitäten hinzufügen aus, um übergeordnete Entitäten hinzuzufügen, denen die neue Entität angehört. Um ein übergeordnetes Element zu entfernen, das der Entität hinzugefügt wurde, wählen Sie neben dem Namen der übergeordneten Entität die Option Entfernen aus.
9. Wählen Sie Attribut hinzufügen, um der Entität Attribute hinzuzufügen. Geben Sie den Attributnamen ein und wählen Sie den Attributtyp für jedes Attribut der Entität aus. Verified Permissions verwendet die angegebenen Attributwerte, wenn Richtlinien anhand des Schemas überprüft werden. Wählen Sie aus, ob jedes Attribut erforderlich ist. Um ein Attribut zu entfernen, das der Entität hinzugefügt wurde, wählen Sie neben dem Attribut die Option Entfernen aus.
10. Wählen Sie Entitätstyp hinzufügen, um die Entität zum Schema hinzuzufügen.
11. Wählen Sie im Abschnitt Aktionen die Option Neue Aktion hinzufügen aus.
12. Geben Sie den Namen der Aktion ein.

13. (Optional) Wählen Sie Ressource hinzufügen aus, um Ressourcentypen hinzuzufügen, für die die Aktion gilt. Um einen Ressourcentyp zu entfernen, der der Aktion hinzugefügt wurde, wählen Sie neben dem Namen des Ressourcentyps die Option Entfernen aus.
14. (Optional) Wählen Sie Prinzipal hinzufügen aus, um einen Prinzipaltyp hinzuzufügen, für den die Aktion gilt. Um einen Prinzipaltyp zu entfernen, der der Aktion hinzugefügt wurde, wählen Sie neben dem Namen des Prinzipaltyps die Option Entfernen aus.
15. Wählen Sie Attribut hinzufügen aus, um Attribute hinzuzufügen, die dem Kontext einer Aktion in Ihren Autorisierungsanfragen hinzugefügt werden können. Geben Sie den Attributnamen ein und wählen Sie den Attributtyp für jedes Attribut aus. Verified Permissions verwendet die angegebenen Attributwerte, wenn Richtlinien anhand des Schemas überprüft werden. Wählen Sie aus, ob jedes Attribut erforderlich ist. Um ein Attribut zu entfernen, das der Aktion hinzugefügt wurde, wählen Sie neben dem Attribut die Option Entfernen aus.
16. Wählen Sie Aktion hinzufügen aus.
17. Nachdem alle Entitätstypen und Aktionen zum Schema hinzugefügt wurden, wählen Sie Änderungen speichern aus.

Bearbeiten von Schemas im JSON-Modus

Um ein Schema für verifizierte Berechtigungen im JSON-Modus zu bearbeiten

1. Öffnen Sie die Konsole „Verified Permissions“ unter https://console.aws.amazon.com/verifiedpermissions/. Wählen Sie Ihren Richtlinienspeicher aus.
2. Wählen Sie im Navigationsbereich auf der linken Seite Schema aus.
3. Wählen Sie den JSON-Modus und dann Schema bearbeiten aus.
4. Geben Sie den Inhalt Ihres JSON-Schemas in das Feld Inhalt ein. Sie können Aktualisierungen Ihres Schemas erst speichern, wenn Sie alle Syntaxfehler behoben haben. Sie können Format JSON wählen, um die JSON-Syntax Ihres Schemas mit den empfohlenen Abständen und Einzügen zu formatieren.
5. Wählen Sie Änderungen speichern aus.

Ein Schema löschen

AWS Management Console

Um ein Schema für verifizierte Berechtigungen zu löschen

1. Öffnen Sie die Konsole Verified Permissions unter https://console.aws.amazon.com/verifiedpermissions/. Wählen Sie Ihren Richtlinienspeicher aus.
2. Wählen Sie im Navigationsbereich auf der linken Seite Schema aus.
3. Wählen Sie Schema löschen aus.

AWS CLI

Um ein Schema für verifizierte Berechtigungen zu löschen

Es gibt keinen Befehl zum Löschen eines Schemas. Sie können das Schema in einem Richtlinienspeicher löschen, indem Sie den `put-schema` Befehl mit einem leeren Schema im `cedarJson` Feld verwenden. Ein leeres Schema wird durch ein Paar geschweifter Klammern '{}' dargestellt.

```
$ aws verifiedpermissions put-schema \
    --policy-store-id PSEXAMPLEabcdefg111111 \
    --definition cedarJson='{}'{
    "policyStoreId": "PSEXAMPLEabcdefg111111",
    "namespaces": [],
    "createdDate": "2023-06-14T21:55:27.347581Z",
    "lastUpdatedDate": "2023-06-19T17:55:04.95944Z"
}
```

Richtlinienvalidierungsmodus für Amazon Verified Permissions

Sie können den Richtlinienvalidierungsmodus in Verified Permissions festlegen, um zu steuern, ob Richtlinienänderungen anhand des Schemas in Ihrem Richtlinienspeicher validiert werden.

> **Important**
>
> Wenn Sie die Richtlinienvalidierung aktivieren, werden alle Versuche, eine Richtlinie oder Richtlinienvorlage zu erstellen oder zu aktualisieren, anhand des Schemas im Richtlinienspeicher validiert. Verified Permissions lehnt die Anforderung ab, wenn die Validierung fehlschlägt.

AWS Management Console

So legen Sie den Richtlinienvalidierungsmodus für einen Richtlinienspeicher fest

1. Öffnen Sie die Konsole Verified Permissions unter https://console.aws.amazon.com/verifiedpermissions/. Wählen Sie Ihren Richtlinienspeicher aus.
2. Wählen Sie Settings (Einstellungen) aus.
3. Wählen Sie im Abschnitt Richtlinienvalidierungsmodus die Option Ändern aus.
4. Führen Sie eine der folgenden Aktionen aus:
   - Um die Richtlinienvalidierung zu aktivieren und durchzusetzen, dass alle Richtlinienänderungen anhand Ihres Schemas validiert werden müssen, wählen Sie das Optionsfeld Strikt (empfohlen).
   - Um die Richtlinienvalidierung für Richtlinienänderungen zu deaktivieren, wählen Sie das Optionsfeld Aus. Geben Sie ein, `confirm` um zu bestätigen, dass Aktualisierungen von Richtlinien nicht mehr anhand Ihres Schemas validiert werden.
5. Wählen Sie Änderungen speichern aus.

AWS CLI

So legen Sie den Validierungsmodus für einen Richtlinienspeicher fest

Sie können den Validierungsmodus für einen Richtlinienspeicher ändern, indem Sie die -UpdatePolicyStoreOperation verwenden und einen anderen Wert für den -ValidationSettingsParameter angeben.

```
$ 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"
    }
}
```

Weitere Informationen finden Sie unter Richtlinienvalidierung im Referenzhandbuch zur Cedar-Richtliniensprache .

Richtlinien für verifizierte Berechtigungen von Amazon

Eine Richtlinie ist eine Erklärung, die einem Principal entweder erlaubt oder verbietet, eine oder mehrere Aktionen an einer Ressource durchzuführen. Jede Richtlinie wird unabhängig von allen anderen Richtlinien bewertet. Weitere Informationen darüber, wie die Richtlinien von Cedar strukturiert und bewertet werden, finden Sie unter Cedar Policy Validation against Schema im Cedar Policy Language Reference Guide.

Important

Wenn Sie Cedar-Richtlinien verfassen, die sich auf Prinzipale, Ressourcen und Aktionen beziehen, können Sie die eindeutigen Identifikatoren definieren, die für jedes dieser Elemente verwendet werden. Wir empfehlen Ihnen dringend, die folgenden Best Practices zu befolgen:

- Verwenden Sie Werte wie Universally Unique Identifiers (UUIDs) für alle Prinzipal- und Ressourcen-Identifikatoren.

  Wenn beispielsweise ein Benutzer das Unternehmen `jane` verlässt und Sie später einer anderen Person die Verwendung des Namens gestatten, erhält dieser neue Benutzer automatisch Zugriff auf alles`jane`, was durch Richtlinien gewährt wird, auf die immer noch verwiesen wird. `User::"jane"` Cedar kann nicht zwischen dem neuen und dem alten Benutzer unterscheiden. Dies gilt sowohl für Prinzipal- als auch für Ressourcen-IDs. Verwenden Sie immer Identifikatoren, die garantiert einzigartig sind und niemals wiederverwendet werden, um sicherzustellen, dass Sie nicht versehentlich Zugriff gewähren, weil eine Richtlinie eine alte Kennung enthält.

  Wenn Sie eine UUID für eine Entität verwenden, empfehlen wir, dass Sie ihr den Kommentarbezeichner//und den „freundlichen“ Namen Ihrer Entität folgen. Dies trägt dazu bei, dass Ihre Richtlinien leichter verständlich sind. Zum Beispiel: principal == User: :"a1b2c3d4-e5f6-A1b2-c3d4-example11111",//alice
- Nehmen Sie keine personenbezogenen, vertraulichen oder sensiblen Informationen als Teil der eindeutigen Kennung für Ihre Prinzipale oder Ressourcen auf. Diese Kennungen sind in Protokolleinträgen enthalten, die in Trails geteilt werden. AWS CloudTrail

Formatierung von Entitäten in Amazon Verified Permissions

Amazon Verified Permissions verwendet die Richtliniensprache von Cedar, um Richtlinien zu erstellen. Die Syntax der Richtlinien und die unterstützten Datentypen entsprechen der Syntax und den Datentypen, die in den Themen Grundlegende Richtlinienkonstruktion in Cedar und Von Cedar unterstützte Datentypen im Referenzhandbuch zur Richtliniensprache von Cedar beschrieben sind. Es gibt jedoch Unterschiede zwischen Verified Permissions und Cedar bei der Formatierung von Entitäten, wenn eine Autorisierungsanfrage gestellt wird.

Die JSON-Formatierung von Entitäten in Verified Permissions unterscheidet sich in folgenden Punkten von Cedar:

- In Verified Permissions müssen alle Schlüssel-Wert-Paare eines JSON-Objekts in ein JSON-Objekt mit dem Namen eingeschlossen sein. `Record`
- Eine JSON-Liste in Verified Permissions muss in ein JSON-Schlüssel-Wert-Paar eingeschlossen werden, wobei der Schlüsselname `Set` und der Wert die ursprüngliche JSON-Liste von Cedar sind.
- Für `String`, `Long`, und `Boolean` Typnamen wird jedes Schlüssel-Wert-Paar aus Cedar in Verified Permissions durch ein JSON-Objekt ersetzt. Der Name des Objekts ist der ursprüngliche Schlüsselname. Innerhalb des JSON-Objekts gibt es ein Schlüssel-Wert-Paar, wobei der Schlüsselname der Typname des Skalarwerts (`String`, `Long`, oder`Boolean`) und der Wert der Wert aus der Cedar-Entität ist.
- Die Syntaxformatierung von Cedar-Entitäten und Verified Permissions-Entitäten unterscheidet sich in folgenden Punkten:

Cedar-Format	Format für verifizierte Berechtigungen
`uid`	`Identifier`
`type`	`EntityType`
`id`	`EntityId`
`attrs`	`Attributes`
`parents`	`Parents`

Das folgende Beispiel zeigt, wie Entitäten in einer Liste mithilfe von Cedar formatiert werden.

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

Das folgende Beispiel zeigt, wie dieselben Entitäten aus dem vorherigen Beispiel für eine Cedar-Liste in Verified Permissions formatiert werden.

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

Das folgende Beispiel zeigt, wie Cedar-Entitäten für die Auswertung einer Richtlinie in einer Autorisierungsanfrage formatiert werden.

```
[
    {
        "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": []
    }
]
```

Das folgende Beispiel zeigt, wie dieselben Entitäten aus dem vorherigen Cedar-Beispiel in Verified Permissions formatiert werden.

```
[
    {
        "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": []
    }
]
```

Statische Richtlinien für Amazon Verified Permissions erstellen

Sie können eine statische Cedar-Richtlinie erstellen, um es Prinzipalen zu ermöglichen oder zu verweigern, bestimmte Aktionen mit bestimmten Ressourcen für Ihre Anwendung auszuführen.

AWS Management Console

Um eine statische Richtlinie zu erstellen

1. Öffnen Sie die Konsole Verified Permissions unter https://console.aws.amazon.com/verifiedpermissions/. Wählen Sie Ihren Richtlinienspeicher aus.
2. Wählen Sie im Navigationsbereich auf der linken Seite Policies (Richtlinien).
3. Wählen Sie Richtlinie erstellen und dann Statische Richtlinie erstellen aus.
4. Wählen Sie im Abschnitt Richtlinienwirkung aus, ob die Richtlinie Zulassen oder Verbieten soll, wenn eine Anfrage mit der Richtlinie übereinstimmt.
5. Wählen Sie im Feld Geltungsbereich für Prinzipale den Geltungsbereich der Prinzipale aus, für die die Richtlinie gelten soll.
   - Wählen Sie Spezifischer Auftraggeber aus, um die Richtlinie auf einen bestimmten Prinzipal anzuwenden. Geben Sie den Entitätstyp und die Kennung für den Prinzipal an, der die in der Richtlinie angegebenen Aktionen ausführen darf oder nicht.
   - Wählen Sie Gruppe von Prinzipalen, um die Richtlinie auf eine Gruppe von Prinzipalen anzuwenden. Geben Sie den Namen der Hauptgruppe in das Feld Gruppe von Prinzipalen ein.
   - Wählen Sie Alle Prinzipale aus, um die Richtlinie auf alle Prinzipale in Ihrem Richtlinienspeicher anzuwenden.
6. Wählen Sie im Feld Umfang der Ressourcen den Umfang der Ressourcen aus, für die die Richtlinie gelten soll.
   - Wählen Sie Bestimmte Ressourcen aus, um die Richtlinie auf eine bestimmte Ressource anzuwenden. Geben Sie den Entitätstyp und die Kennung für die Ressource an, für die die Richtlinie gelten soll.
   - Wählen Sie Gruppe von Ressourcen, um die Richtlinie auf eine Gruppe von Ressourcen anzuwenden. Geben Sie den Namen der Ressourcengruppe in das Feld Ressourcengruppe ein.
   - Wählen Sie Alle Ressourcen aus, um die Richtlinie auf alle Ressourcen in Ihrem Richtlinienspeicher anzuwenden.
7. Wählen Sie im Abschnitt Geltungsbereich der Aktionen den Umfang der Ressourcen aus, für die die Richtlinie gelten soll.

- Wählen Sie Spezifische Gruppe von Aktionen aus, um die Richtlinie auf eine Reihe von Aktionen anzuwenden. Aktivieren Sie die Kontrollkästchen neben den Aktionen, um die Richtlinie anzuwenden.
- Wählen Sie Alle Aktionen aus, um die Richtlinie auf alle Aktionen in Ihrem Richtlinienspeicher anzuwenden.

8. Wählen Sie Weiter aus.
9. Überprüfen Sie im Abschnitt „Richtlinien“ Ihre Cedar-Richtlinie. Sie können Format wählen, um die Syntax Ihrer Richtlinie mit den empfohlenen Abständen und Einzügen zu formatieren. Weitere Informationen finden Sie im Cedar Policy Language Reference Guide unter Basic Policy Construction in Cedar.
10. Geben Sie im Abschnitt Details eine optionale Beschreibung der Richtlinie ein.
11. Wählen Sie Richtlinie erstellen aus.

AWS CLI

Um eine statische Richtlinie zu erstellen

Mithilfe des CreatePolicyVorgangs können Sie eine statische Richtlinie erstellen. Das folgende Beispiel erstellt eine einfache statische Richtlinie.

```
$ 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"
}
```

Statische Richtlinien von Amazon Verified Permissions bearbeiten

Sie können eine bestehende statische Cedar-Richtlinie in Ihrem Richtlinienspeicher bearbeiten. Sie können nur statische Richtlinien direkt aktualisieren. Sie können nur bestimmte Elemente einer statischen Richtlinie ändern:

- Die, `action` auf die in der Richtlinie verwiesen wird.
- Eine Bedingungsklausel, z. B. `when` und`unless`.

Sie können diese Elemente einer statischen Richtlinie nicht ändern:

- Änderung einer Richtlinie von einer statischen Richtlinie in eine mit einer Vorlage verknüpfte Richtlinie.
- Änderung der Wirkung einer statischen Richtlinie von `permit` oder. `forbid`
- Der, `principal` auf den eine statische Richtlinie verweist.
- Wird von einer statischen Richtlinie `resource` referenziert.

Um eine mit einer Vorlage verknüpfte Richtlinie zu ändern, müssen Sie stattdessen die Vorlage aktualisieren. Weitere Informationen finden Sie unter Bearbeiten von Richtlinienvorlagen.

AWS Management Console

Um eine statische Richtlinie zu bearbeiten

1. Öffnen Sie die Konsole Verified Permissions unter https://console.aws.amazon.com/verifiedpermissions/. Wählen Sie Ihren Richtlinienspeicher aus.
2. Wählen Sie im Navigationsbereich auf der linken Seite Policies (Richtlinien).
3. Wählen Sie das Optionsfeld neben der statischen Richtlinie, die Sie bearbeiten möchten, und wählen Sie dann Bearbeiten aus.
4. Aktualisieren Sie im Abschnitt Richtlinientext die Klausel `action` oder die Bedingungsklausel Ihrer statischen Richtlinie. Sie können den Richtlinieneffekt oder `principal` `resource` die Richtlinie nicht aktualisieren.
5. Wählen Sie Richtlinie aktualisieren.

> **Note**
>
> Wenn die Richtlinienüberprüfung im Richtlinienspeicher aktiviert ist, führt die Aktualisierung einer statischen Richtlinie dazu, dass Verified Permissions die Richtlinie anhand des Schemas im Richtlinienspeicher validiert. Wenn die aktualisierte statische Richtlinie die Validierung nicht besteht, schlägt der Vorgang fehl und das Update wird nicht gespeichert.

AWS CLI

Um eine statische Richtlinie zu bearbeiten

Sie können eine statische Richtlinie mithilfe des UpdatePolicyVorgangs bearbeiten. Im folgenden Beispiel wird eine einfache statische Richtlinie bearbeitet.

Im Beispiel wird die Datei verwendet`definition.txt`, um die Richtliniendefinition zu enthalten.

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

Der folgende Befehl verweist auf diese Datei.

```
$ 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"
    }
}
```

Richtlinien anzeigen

AWS Management Console

Um Ihre Richtlinien für verifizierte Berechtigungen einzusehen

1. Öffnen Sie die Konsole für verifizierte Berechtigungen unter https://console.aws.amazon.com/verifiedpermissions/. Wählen Sie Ihren Richtlinienspeicher aus.
2. Wählen Sie im Navigationsbereich auf der linken Seite Policies (Richtlinien). Alle Richtlinien, die Sie erstellt haben, werden angezeigt.
3. Wählen Sie das Textfeld Suchen aus, um Richtlinien nach Prinzipal oder Ressource zu filtern.
4. Wählen Sie das Optionsfeld neben einer Richtlinie aus, um Details zur Richtlinie anzuzeigen, z. B. wann die Richtlinie erstellt und aktualisiert wurde, und den Inhalt der Richtlinie.
5. Sie können eine Richtlinie löschen, indem Sie das Optionsfeld neben einer Richtlinie und dann Löschen auswählen. Wählen Sie Richtlinie löschen, um das Löschen der Richtlinie zu bestätigen.

AWS CLI

Um alle verfügbaren Richtlinien in einem Richtlinienspeicher aufzulisten

Sie können die Liste der Richtlinien mithilfe des GetPolicyVorgangs anzeigen. Im folgenden Beispiel wird eine Liste abgerufen, die eine statische Richtlinie und eine mit einer Vorlage verknüpfte Richtlinie enthält.

```
$ aws verifiedpermissions list-policies \
    --policy-store-id PSEXAMPLEabcdefg111111
{
    "Policies": [
        {
```

```
            "createdDate": "2023-05-17T18:38:31.359864+00:00",
            "definition": {
                "static": {
                    "Description": "Grant everyone of janeFriends UserGroup access
 to the vacationFolder Album"
                }
            },
            "lastUpdatedDate": "2023-05-18T16:15:04.366237+00:00",
            "policyId": "SPEXAMPLEabcdefg111111",
            "policyStoreId": "PSEXAMPLEabcdefg111111",
            "policyType": "STATIC",
            "resource": {
                "entityId": "publicFolder",
                "entityType": "Album"
            }
        },
        {
            "createdDate": "2023-05-22T18:57:53.298278+00:00",
            "definition": {
                "templateLinked": {
                    "policyTemplateId": "PTEXAMPLEabcdefg111111"
                }
            },
            "lastUpdatedDate": "2023-05-22T18:57:53.298278+00:00",
            "policyId": "TPEXAMPLEabcdefg111111",
            "policyStoreId": "PSEXAMPLEabcdefg111111",
            "policyType": "TEMPLATELINKED",
            "principal": {
                "entityId": "alice",
                "entityType": "User"
            },
            "resource": {
                "entityId": "VacationPhoto94.jpg",
                "entityType": "Photo"
            }
        }
    ]
}
```

Um die Details für eine einzelne Richtlinie anzuzeigen

Sie können die Details für eine Richtlinie mithilfe des GetPolicyVorgangs abrufen. Im folgenden Beispiel werden Details für eine mit einer Vorlage verknüpfte Richtlinie abgerufen.

```
$ aws verifiedpermissions get-policy \
    --policy-id TPEXAMPLEabcdefg111111
    --policy-store-id PSEXAMPLEabcdefg111111

{
    "arn": "arn:aws:verifiedpermissions::123456789012:policy/PSEXAMPLEabcdefg111111/
TPEXAMPLEabcdefg111111",
    "createdDate": "2023-03-15T16:03:07.620867Z",
    "lastUpdatedDate": "2023-03-15T16:03:07.620867Z",
    "policyDefinition": {
        "templatedPolicy": {
            "policyTemplateId": "PTEXAMPLEabcdefg111111",
            "principal": {
                "entityId": "alice",
                "entityType": "User"
            },
            "resource": {
                "entityId": "Vacation94.jpg",
                "entityType": "Photo"
            }
        }
    },
    "policyId": "TPEXAMPLEabcdefg111111",
    "policyStoreId": "PSEXAMPLEabcdefg111111",
    "policyType": "TEMPLATELINKED",
    "principal": {
        "entityId": "alice",
        "entityType": "User"
    },
    "resource": {
        "entityId": "Vacation94.jpg",
        "entityType": "Photo"
    }
}
```

Beispielrichtlinien für Amazon Verified Permissions

Die folgenden Richtlinienbeispiele für verifizierte Berechtigungen basieren auf dem Schema, das für die hypothetische Anwendung definiert wurde und im Abschnitt Beispielschema des Cedar Policy Language Reference Guide PhotoFlash beschrieben wird. Weitere Informationen zur Cedar-Policy-

Syntax finden Sie unter Basic Policy Construction in Cedar im Cedar Policy Language Reference Guide.

Beispiele für Richtlinien

- Ermöglicht den Zugriff auf einzelne Entitäten
- Ermöglicht den Zugriff auf Gruppen von Entitäten
- Ermöglicht den Zugriff für jede Entität
- Ermöglicht den Zugriff auf Attribute einer Entität (ABAC)
- Verweigert den Zugriff

Ermöglicht den Zugriff auf einzelne Entitäten

Dieses Beispiel zeigt, wie Sie eine Richtlinie erstellen könnten, die es dem Benutzer `alice` ermöglicht, das Foto anzusehen`VacationPhoto94.jpg`.

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

Ermöglicht den Zugriff auf Gruppen von Entitäten

Dieses Beispiel zeigt, wie Sie eine Richtlinie erstellen könnten, die es allen Mitgliedern der Gruppe `alice_friends` ermöglicht, das Foto anzusehen`VacationPhoto94.jpg`.

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

Dieses Beispiel zeigt, wie Sie eine Richtlinie erstellen könnten, die es dem Benutzer `alice` ermöglicht, jedes Foto im Album anzusehen`alice_vacation`.

```
permit(
  principal == User::"alice",
  action == Action::"view",
```

```
  resource in Album::"alice_vacation"
);
```

Dieses Beispiel zeigt, wie Sie eine Richtlinie erstellen könnten, die es dem Benutzer `alice` ermöglicht, jedes Foto im Album anzusehen, zu bearbeiten oder zu löschen`alice_vacation`.

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

Dieses Beispiel zeigt, wie Sie eine Richtlinie erstellen könnten, die dem Benutzer `alice` Berechtigungen für das Album `alice_vacation` gewährt. Dabei `admin` handelt es sich um eine in der Schemahierarchie definierte Gruppe, die die Berechtigungen zum Anzeigen, Bearbeiten und Löschen eines Fotos enthält.

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

Dieses Beispiel zeigt, wie Sie eine Richtlinie erstellen könnten, die dem Benutzer `alice` Berechtigungen für das Album `alice_vacation` gewährt. Dabei `viewer` handelt es sich um eine in der Schemahierarchie definierte Gruppe, die die Berechtigung zum Ansehen und Kommentieren eines Fotos enthält. Dem Benutzer `alice` wird die `edit` Berechtigung auch durch die zweite in der Richtlinie aufgeführte Aktion erteilt.

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

Ermöglicht den Zugriff für jede Entität

Dieses Beispiel zeigt, wie Sie eine Richtlinie erstellen könnten, die es jedem authentifizierten Prinzipal ermöglicht, das Album `alice_vacation` anzusehen.

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

Dieses Beispiel zeigt, wie Sie eine Richtlinie erstellen könnten, die es dem Benutzer `alice` ermöglicht, alle Alben im `jane` Konto aufzulisten, die Fotos in jedem Album aufzulisten und die Fotos im Konto anzusehen.

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

Dieses Beispiel zeigt, wie Sie eine Richtlinie erstellen könnten, die es dem Benutzer `alice` ermöglicht, beliebige Aktionen mit Ressourcen im Album durchzuführen`jane_vaction`.

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

Ermöglicht den Zugriff auf Attribute einer Entität (ABAC)

Die attributbasierte Zugriffskontrolle (ABAC) ist eine Autorisierungsstrategie, bei der Berechtigungen basierend auf Attributen definiert werden. Verifizierte Berechtigungen ermöglichen das Anhängen von Attributen an Prinzipale, Aktionen und Ressourcen. Auf diese Attribute kann dann in den Richtlinienklauseln `when` und `unless` -klauseln verwiesen werden, in denen die Attribute der Prinzipale, Aktionen und Ressourcen bewertet werden, die den Kontext der Anfrage bilden.

In den folgenden Beispielen werden die Attribute verwendet, die in der hypothetischen Anwendung definiert sind, die im Abschnitt Beispielschema des Cedar Policy Language Reference Guide PhotoFlash beschrieben wird.

Dieses Beispiel zeigt, wie Sie eine Richtlinie erstellen könnten, die es jedem Schulleiter in der `HardwareEngineering` Abteilung mit einem Stellenlevel von mindestens 5 ermöglicht, Fotos im Album anzusehen und aufzulisten. `device_prototypes`

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

Dieses Beispiel zeigt, wie Sie eine Richtlinie erstellen könnten, die es dem Benutzer `alice` ermöglicht, jede Ressource eines Dateityps anzuzeigen`JPEG`.

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

Aktionen haben Kontext-Attribute. Sie müssen diese Attribute in einer Autorisierungsanfrage übergeben. `context` Dieses Beispiel zeigt, wie Sie eine Richtlinie erstellen könnten, die es dem Benutzer `alice` ermöglicht, jede `readOnly` Aktion auszuführen. Sie können auch eine `appliesTo` Eigenschaft für Aktionen in Ihrem Schema festlegen. Dies gibt gültige Aktionen für eine Ressource an, wenn Sie sicherstellen möchten, dass Benutzer beispielsweise nur versuchen können, sich `ViewPhoto` für eine Ressource des Typs `PhotoFlash::Photo` zu autorisieren.

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

Eine bessere Methode, die Eigenschaften von Aktionen in Ihrem Schema festzulegen, besteht jedoch darin, sie in funktionalen Aktionsgruppen anzuordnen. Sie können beispielsweise

eine Aktion mit dem Namen einer Aktionsgruppe erstellen `ReadOnlyPhotoAccess` und festlegen`PhotoFlash::Action::"ViewPhoto"`, dass `ReadOnlyPhotoAccess` sie Mitglied dieser Aktion ist. Dieses Beispiel zeigt, wie Sie eine Richtlinie erstellen könnten, die Alice Zugriff auf die schreibgeschützten Aktionen in dieser Gruppe gewährt.

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

Dieses Beispiel zeigt, wie Sie eine Richtlinie erstellen könnten, die es allen Prinzipalen ermöglicht, jede Aktion mit Ressourcen durchzuführen, für die sie ein Attribut haben. `owner`

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

Dieses Beispiel zeigt, wie Sie eine Richtlinie erstellen könnten, die es jedem Prinzipal ermöglicht, jede Ressource anzuzeigen, wenn das `department` Attribut für den Prinzipal mit dem `department` Attribut der Ressource übereinstimmt.

> **Note**
>
> Wenn für eine Entität kein Attribut in einer Richtlinienbedingung erwähnt wird, wird die Richtlinie ignoriert, wenn eine Autorisierungsentscheidung getroffen wird und die Bewertung dieser Richtlinie für diese Entität fehlschlägt. Beispielsweise kann einem Prinzipal, der kein `department` Attribut hat, durch diese Richtlinie kein Zugriff auf eine Ressource gewährt werden.

```
permit(
```

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

Dieses Beispiel zeigt, wie Sie eine Richtlinie erstellen könnten, die es jedem Prinzipal ermöglicht, jede Aktion an einer Ressource auszuführen, wenn der Principal `owner` der Ressource ist ODER wenn der Principal Teil der `admins` Gruppe für die Ressource ist.

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

Verweigert den Zugriff

Wenn eine Richtlinie die Wirkung der Richtlinie vorsieht, schränkt sie Berechtigungen ein, anstatt sie zu gewähren. `forbid`

> **Important**
>
> Wenn bei der Autorisierung `permit` sowohl eine als auch eine `forbid` Richtlinie durchgesetzt werden, `forbid` hat die Vorrang.

In den folgenden Beispielen werden die in der hypothetischen Anwendung definierten Attribute verwendet, die im Abschnitt Beispielschema des Cedar Policy Language Reference Guide PhotoFlash beschrieben sind.

Dieses Beispiel zeigt, wie Sie eine Richtlinie erstellen könnten, die dem Benutzer `alice` verbietet, alle Aktionen außer `readOnly` für eine Ressource auszuführen.

```
forbid (
```

```
  principal == User::"alice",
  action,
  resource
)
unless {
  action.readOnly
};
```

Dieses Beispiel zeigt, wie Sie eine Richtlinie erstellen könnten, die den Zugriff auf alle Ressourcen verweigert, die über ein `private` Attribut verfügen, es sei denn, der Principal hat das `owner` Attribut für die Ressource.

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

Amazon Verified Permissions

Sie können in Verified Permissions Richtlinienvorlagen für Cedar erstellen, um eine Zugriffskontrollregel für Ihr System zu definieren. Richtlinienvorlagen sind Cedar-Richtlinien mit Platzhaltern für`principal`,`resource`, oder beides. Mit Richtlinienvorlagen kann eine Richtlinie einmal definiert und dann mehreren Prinzipalen und Ressourcen zugewiesen werden. Aktualisierungen der Richtlinienvorlage gelten für alle Prinzipale und Ressourcen, die die Vorlage verwenden. Weitere Informationen finden Sie unterVorlagen für Richtlinienim Cedar Policy Language Reference Guide.

Wir empfehlen die Verwendung von Richtlinienvorlagen, um Richtlinien zu erstellen, die in Ihrer gesamten Anwendung gemeinsam genutzt werden können. Sie könnten beispielsweise eine Richtlinienvorlage für einen Editor erstellen, der Lese-, Bearbeitungs- und Kommentarberechtigungen für den Prinzipal und die Ressource bereitstellt, die die Richtlinienvorlage verwenden.

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

Wenn ein Prinzipal als Bearbeiter für eine Ressource bestimmt ist, könnte Ihre Anwendung mithilfe der Vorlage eine Richtlinie instanziieren, um dem Prinzipal Berechtigungen zum Durchführen der Lese-, Bearbeitungs- und Kommentierungsaktionen für die Ressource zu erteilen.

Erstellen von Richtlinienvorlagen

AWS Management Console

So erstellen Sie eine Richtlinienvorlage

1. Öffnen Sie die Konsole Verified Permissions unter https://console.aws.amazon.com/verifiedpermissions/. Wählen Sie Ihren Richtlinienspeicher aus.
2. Wählen Sie im Navigationsbereich auf der linken Seite Richtlinienvorlagen aus.
3. Wählen Sie Richtlinienvorlage erstellen aus.
4. Geben Sie im Abschnitt Details eine Beschreibung der Richtlinienvorlage ein.

5. Verwenden Sie im Abschnitt Richtlinienvorlagentext Platzhalter `?principal` und , `?resource` um Richtlinien, die basierend auf dieser Vorlage erstellt wurden, zu erlauben, die von ihnen gewährten Berechtigungen anzupassen. Sie können Format auswählen, um die Syntax Ihrer Richtlinienvorlage mit dem empfohlenen Abstand und den empfohlenen Einrückungen zu formatieren.
6. Wählen Sie Richtlinienvorlage erstellen aus.

AWS CLI

So erstellen Sie eine Richtlinienvorlage

Sie können eine Richtlinienvorlage mithilfe der -CreatePolicyTemplateOperation erstellen. Im folgenden Beispiel wird eine Richtlinienvorlage mit einem Platzhalter für den Prinzipal erstellt.

Die Datei `template1.txt` enthält Folgendes.

```
"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"
}
```

Erstellen von vorlagenverknüpften Richtlinien

Sie können vorlagenverknüpfte Richtlinien erstellen, um einen Link zu einer Richtlinienvorlage herzustellen. Vorlagenverknüpfte Richtlinien bleiben mit ihren Richtlinienvorlagen verknüpft. Wenn Sie die Richtlinienanweisung in der Richtlinienvorlage ändern, verwenden alle mit dieser Vorlage

verknüpften Richtlinien automatisch die neue Anweisung für alle Autorisierungsentscheidungen, die ab diesem Zeitpunkt getroffen wurden.

AWS Management Console

So erstellen Sie eine vorlagenverknüpfte Richtlinie durch Instanziieren einer Richtlinienvorlage

1. Öffnen Sie die Konsole Verified Permissions unter https://console.aws.amazon.com/verifiedpermissions/. Wählen Sie Ihren Richtlinienspeicher aus.
2. Wählen Sie im Navigationsbereich auf der linken Seite Policies (Richtlinien).
3. Wählen Sie Richtlinie erstellen und dann Vorlagenverknüpfte Richtlinie erstellen aus.
4. Wählen Sie das Optionsfeld neben der zu verwendenden Richtlinienvorlage und dann Weiter aus.
5. Geben Sie den Prinzipal und die Ressource ein, die für diese spezifische Instance der vorlagenverknüpften Richtlinie verwendet werden sollen. Die angegebenen Werte werden im Vorschaufeld Richtlinienanweisung angezeigt.

   > **Note**
   >
   > Die Prinzipal- und Ressourcenwerte müssen dieselbe Formatierung wie statische Richtlinien haben. Um beispielsweise die `AdminUsers` Gruppe für den Prinzipal anzugeben, geben Sie ein`Group::"AdminUsers"`. Wenn Sie eingeben`AdminUsers`, wird ein Validierungsfehler angezeigt.

6. Wählen Sie Vorlagenverknüpfte Richtlinie erstellen aus.

   Die neue vorlagenverknüpfte Richtlinie wird unter Richtlinien angezeigt.

AWS CLI

So erstellen Sie eine vorlagenverknüpfte Richtlinie durch Instanziieren einer Richtlinienvorlage

Sie können eine vorlagenverknüpfte Richtlinie erstellen, die auf eine vorhandene Richtlinienvorlage verweist und Werte für alle Platzhalter angibt, die von der Vorlage verwendet werden.

Im folgenden Beispiel wird eine vorlagenverknüpfte Richtlinie erstellt, die eine Vorlage mit der folgenden Anweisung verwendet:

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

Außerdem wird die folgende `definition.txt` Datei verwendet, um den Wert für den `definition` Parameter anzugeben:

```
{
    "templateLinked": {
        "policyTemplateId": "pt-4651be67-c128-4d22-8e67-9b068980c631",
        "principal": {
            "entityType": "User",
            "entityId": "alice"
        }
    }
}
```

Die Ausgabe zeigt sowohl die Ressource, die sie aus der Vorlage erhält, als auch den Prinzipal, den sie aus dem Definitionsparameter erhält

```
$ 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": "User"
    },
    "resource": {
        "entityId": "VacationPhoto94.jpg",
        "entityType": "Photo"
    }
}
```

Bearbeiten von Richtlinienvorlagen

AWS Management Console

So bearbeiten Sie Ihre Richtlinienvorlagen

1. Öffnen Sie die Verified Permissions-Konsole unter https://console.aws.amazon.com/verifiedpermissions/. Wählen Sie Ihren Richtlinienspeicher aus.
2. Wählen Sie im Navigationsbereich auf der linken Seite Richtlinienvorlagen aus. Die Konsole zeigt alle Richtlinienvorlagen an, die Sie im aktuellen Richtlinienspeicher erstellt haben.
3. Wählen Sie das Optionsfeld neben einer Richtlinienvorlage, um Details zur Richtlinienvorlage anzuzeigen, z. B. wann die Richtlinienvorlage erstellt, aktualisiert und den Inhalt der Richtlinienvorlage.
4. Wählen Sie Bearbeiten, um Ihre Richtlinienvorlage zu bearbeiten. Aktualisieren Sie die Richtlinienbeschreibung und den Richtlinientext nach Bedarf und wählen Sie dann Richtlinienvorlage aktualisieren aus.
5. Sie können eine Richtlinienvorlage löschen, indem Sie das Optionsfeld neben einer Richtlinienvorlage und dann Löschen auswählen. Wählen Sie OK, um das Löschen der Richtlinienvorlage zu bestätigen.

AWS CLI

So aktualisieren Sie eine Richtlinienvorlage

Sie können eine statische Richtlinie mithilfe der -UpdatePolicyOperation erstellen. Im folgenden Beispiel wird die angegebene Richtlinienvorlage aktualisiert, indem der Richtlinientext durch eine neue Richtlinie ersetzt wird, die in einer Datei definiert ist.

Inhalt der Datei `template1.txt`:

```
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"
}
```

Beispiel für vorlagenverknüpfte Richtlinien für Beispielrichtlinienspeicher für Verified Permissions

Wenn Sie einen Richtlinienspeicher in Verified Permissions mit der Methode Beispielrichtlinienspeicher erstellen, wird Ihr Richtlinienspeicher mit vordefinierten Richtlinien, Richtlinienvorlagen und einem Schema für das ausgewählte Beispielprojekt erstellt. Die folgenden Beispiele für vorlagenverknüpfte Richtlinien für verifizierte Berechtigungen können mit den Beispielrichtlinienspeichern und ihren jeweiligen Richtlinien, Richtlinienvorlagen und Schemata verwendet werden.

PhotoFlash Beispiele für vorlagenverknüpfte Richtlinien

Dieses Beispiel zeigt, wie Sie eine vorlagenverknüpfte Richtlinie erstellen können, die die Richtlinienvorlage Gewähren von eingeschränktem Zugriff auf nicht private geteilte Fotos mit einem einzelnen Benutzer und einem einzelnen Foto verwendet.

> **Note**
>
> Die Zedar-Richtliniensprache betrachtet eine Entität als `in` selbst. Daher `principal in User::"Alice"` entspricht `principal == User::"Alice"`.

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

```
 );
```

In diesem Beispiel wird gezeigt, wie Sie eine vorlagenverknüpfte Richtlinie erstellen können, die die Richtlinienvorlage Beschränkten Zugriff auf nicht private freigegebene Fotos für einen einzelnen Benutzer und ein einzelnes Album gewähren verwendet.

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

Dieses Beispiel zeigt, wie Sie eine vorlagenverknüpfte Richtlinie erstellen können, die die Richtlinienvorlage Gewähren von eingeschränktem Zugriff auf nicht private geteilte Fotos mit einer Freundesgruppe und einem einzelnen Foto verwendet.

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

Dieses Beispiel zeigt, wie Sie eine vorlagenverknüpfte Richtlinie erstellen können, die die Richtlinienvorlage Beschränkten Zugriff auf nicht private geteilte Fotos mit einer Freundesgruppe und einem Album gewähren verwendet.

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

Dieses Beispiel zeigt, wie Sie eine vorlagenverknüpfte Richtlinie erstellen können, die die Richtlinienvorlage Vollzugriff auf nicht private geteilte Fotos mit einer Freundesgruppe und einem individuellen Foto gewähren verwendet.

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

```
 );
```

Dieses Beispiel zeigt, wie Sie eine vorlagenverknüpfte Richtlinie erstellen können, die die Richtlinienvorlage Benutzer von einem Konto blockieren verwendet.

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

DigitalPetStore

Der DigitalPetStore Beispielrichtlinienspeicher enthält keine Richtlinienvorlagen. Sie können die im Richtlinienspeicher enthaltenen Richtlinien anzeigen, indem Sie im Navigationsbereich auf der linken Seite Richtlinien auswählen, nachdem Sie den DigitalPetStore Beispielrichtlinienspeicher erstellt haben.

TinyToDo Beispiele für vorlagenverknüpfte Richtlinien

Dieses Beispiel zeigt, wie Sie eine vorlagenverknüpfte Richtlinie erstellen können, die die Richtlinienvorlage verwendet, die Betrachterzugriff für einen einzelnen Benutzer und eine Aufgabenliste gewährt.

```
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"
);
```

Dieses Beispiel zeigt, wie Sie eine vorlagenverknüpfte Richtlinie erstellen können, die die Richtlinienvorlage verwendet, die Editorzugriff für einen einzelnen Benutzer und eine einzelne Aufgabenliste gewährt.

```
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"
);
```

Verwenden von Amazon Verified Permissions mit Identitätsanbietern

Eine Identitätsquelle ist eine Darstellung eines externen Identitätsanbieters (IdP) in Amazon Verified Permissions. Identitätsquellen stellen Informationen von einem Benutzer bereit, der sich bei einem IdP authentifiziert hat, der eine Vertrauensbeziehung zu Ihrem Richtlinienspeicher unterhält. Wenn Ihre Anwendung eine Autorisierungsanfrage mit einem Token aus einer Identitätsquelle stellt, kann Ihr Richtlinienspeicher anhand von Benutzereigenschaften und Zugriffsberechtigungen Autorisierungsentscheidungen treffen. Identitätsquellen mit verifizierten Berechtigungen verbessern die Autorisierung durch eine direkte Verbindung zu Ihrem zentralen Identitätsspeicher und Authentifizierungsdienst.

Sie können OpenID Connect (OIDC) -Identitätsanbieter (IdPs) mit verifizierten Berechtigungen verwenden. Ihre Anwendung kann Autorisierungsanfragen mit OIDC-Identität (ID) generieren oder auf JSON-Webtoken (JWTs) zugreifen. Mit ID-Tokens liest Verified Permissions Benutzer-IDs und Attributansprüche als Prinzipale für die attributebasierte Zugriffskontrolle (ABAC). Bei Zugriffstoken liest Verified Permissions Benutzer-IDs als Hauptbenutzer und andere Ansprüche als Kontext. Mit beiden Tokentypen können Sie einen Anspruch quasi einer Prinzipalgruppe `groups` zuordnen und Richtlinien erstellen, die die rollenbasierte Zugriffskontrolle (RBAC) bewerten.

Sie können einen Amazon Cognito Cognito-Benutzerpool oder einen benutzerdefinierten OpenID Connect (OIDC) IdP als Identitätsquelle hinzufügen.

Themen

Arbeiten mit Amazon Cognito Cognito-Identitätsquellen

Verified Permissions arbeitet eng mit Amazon Cognito Cognito-Benutzerpools zusammen. Amazon Cognito JWTs haben eine vorhersehbare Struktur. Verified Permissions erkennt diese Struktur und zieht den größtmöglichen Nutzen aus den darin enthaltenen Informationen. Sie können beispielsweise ein Autorisierungsmodell für die rollenbasierte Zugriffskontrolle (RBAC) mit ID-Token oder Zugriffstoken implementieren.

Die Identitätsquelle eines neuen Amazon Cognito Cognito-Benutzerpools benötigt die folgenden Informationen:

- Das AWS-Region.
- Die Benutzerpool-ID.
- Der Benutzer-Entitätstyp, den Sie Ihrer Identitätsquelle zuordnen möchten, zum Beispiel`MyCorp::User`.
- Zum Beispiel der Gruppen-Entitätstyp, den Sie Ihrer Identitätsquelle zuordnen möchten`MyCorp::UserGroup`.
- (Optional) Die Client-IDs aus Ihrem Benutzerpool, die Sie autorisieren möchten, Anfragen an Ihren Richtlinienspeicher zu stellen.

Da Verified Permissions nur mit Amazon Cognito Cognito-Benutzerpools in demselben funktioniert AWS-Konto, können Sie keine Identitätsquelle in einem anderen Konto angeben. Verified Permissions legt beispielsweise das Entitätspräfix — die ID der Identitätsquelle, auf die Sie in Richtlinien verweisen müssen, die sich auf Benutzerpool-Prinzipale beziehen — auf die ID Ihres Benutzerpools fest. `us-west-2_EXAMPLE`

Ansprüche auf Benutzerpool-Tokens können Attribute, Bereiche, Gruppen, Client-IDs und benutzerdefinierte Daten enthalten. Amazon Cognito JWTs können in Verified Permissions eine Vielzahl von Informationen aufnehmen, die zu Autorisierungsentscheidungen beitragen können. Dazu zählen:

1. Benutzername und Gruppenansprüche mit einem Präfix `cognito:`
2. Benutzerdefinierte Benutzerattribute mit einem `custom: prefix`
3. Zur Laufzeit hinzugefügte benutzerdefinierte Ansprüche
4. OIDC-Standardansprüche wie und `sub` `email`

Wir behandeln diese Ansprüche ausführlich und erfahren, wie sie verwaltet werden, in den Richtlinien für verifizierte Berechtigungen, unter. Arbeiten mit Identitätsquellen in Schemas und Richtlinien

> **Important**
>
> Sie können Amazon Cognito Cognito-Token zwar vor ihrem Ablauf widerrufen, aber JWTs gelten als zustandslose Ressourcen, die eigenständig sind und über eine Signatur und Gültigkeit verfügen. Von Diensten, die dem JSON Web Token RFC 7519 entsprechen, wird erwartet, dass sie Token remote validieren und müssen sie nicht beim Emittenten validieren. Das bedeutet, dass verifizierte Berechtigungen Zugriff auf der Grundlage eines Tokens gewähren können, das für einen Benutzer gesperrt oder ausgestellt und später gelöscht wurde. Um dieses Risiko zu minimieren, empfehlen wir Ihnen, Ihre Token mit der kürzest möglichen Gültigkeitsdauer zu erstellen und Aktualisierungstoken zu widerrufen, wenn Sie die Autorisierung zur Fortsetzung der Sitzung eines Benutzers entfernen möchten.

Die Richtlinien von Cedar für Identitätsquellen in Benutzerpools in Verified Permissions verwenden eine spezielle Syntax für Anspruchsnamen, die andere Zeichen als alphanumerische Zeichen und Unterstriche () enthalten. _ Dazu gehören auch Ansprüche auf Benutzerpool-Präfixe, die ein `:` Zeichen wie `cognito:username` und enthalten. `custom:department` Um eine Richtlinienbedingung zu schreiben, die auf den `custom:department` Anspruch `cognito:username` oder verweist, schreiben Sie sie jeweils als `principal["cognito:username"]` und`principal["custom:department"]`.

> **Note**
>
> Wenn ein Token einen Anspruch mit dem `custom:` Präfix `cognito:` oder und einen Anspruchsnamen mit dem wörtlichen Wert `cognito` oder enthält`custom`, schlägt eine Autorisierungsanfrage mit a IsAuthorizedWithToken`ValidationException`fehl.

Dieses Beispiel zeigt, wie Sie eine Richtlinie erstellen könnten, die auf einige der Amazon Cognito Cognito-Benutzerpool-Ansprüche verweist, die mit einem Prinzipal verknüpft sind.

```
permit(
     principal == ExampleCo::User::"us-east-1_example|4fe90f4a-ref8d9-4033-
a750-4c8622d62fb6",
     action,
```

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

Weitere Informationen zur Zuordnung von Ansprüchen finden Sie unterZuordnen von ID-Token zum Schema. Weitere Informationen zur Autorisierung für Amazon Cognito-Benutzer finden Sie unter Autorisierung mit von Amazon verifizierten Berechtigungen im Amazon Cognito Developer Guide.

Arbeiten mit OIDC-Identitätsquellen

Sie können auch jeden kompatiblen OpenID Connect (OIDC) IdP als Identitätsquelle für einen Richtlinienspeicher konfigurieren. OIDC-Anbieter ähneln Amazon Cognito Cognito-Benutzerpools: Sie produzieren JWTs als Produkt der Authentifizierung. Um einen OIDC-Anbieter hinzuzufügen, müssen Sie eine Aussteller-URL angeben

Für eine neue OIDC-Identitätsquelle sind die folgenden Informationen erforderlich:

- Die URL des Ausstellers. Verifizierte Berechtigungen müssen in der Lage sein, einen `.well-known/openid-configuration` Endpunkt unter dieser URL zu erkennen.
- Der Tokentyp, den Sie in Autorisierungsanfragen verwenden möchten. In diesem Fall haben Sie Identitätstoken gewählt.
- Der Benutzer-Entitätstyp, den Sie Ihrer Identitätsquelle zuordnen möchten, zum Beispiel`MyCorp::User`.
- Zum Beispiel der Gruppen-Entitätstyp, den Sie Ihrer Identitätsquelle zuordnen möchten`MyCorp::UserGroup`.
- Ein Beispiel für ein ID-Token oder eine Definition der Ansprüche im ID-Token.
- Das Präfix, das Sie auf Benutzer- und Gruppenentitäts-IDs anwenden möchten. In der CLI und API können Sie dieses Präfix wählen. In Policy Stores, die Sie mit der Option „Mit API Gateway einrichten und einer Identitätsquelle einrichten“ oder „Geführte Einrichtung“ erstellen, weist Verified Permissions beispielsweise ein Präfix mit dem Namen des Ausstellers minus `https://` zu. `MyCorp::User::"auth.example.com|a1b2c3d4-5678-90ab-cdef-EXAMPLE11111"`

Bei der Autorisierung mit OIDC-Identitätsquellen werden dieselben API-Operationen verwendet wie bei Identitätsquellen für Benutzerpools: und Token. IsAuthorizedWithTokenBatchIsAuthorizedWith

Dieses Beispiel zeigt, wie Sie eine Richtlinie erstellen könnten, die Mitarbeitern der Buchhaltungsabteilung Zugriff auf Jahresabschlussberichte gewährt, die vertraulich eingestuft sind und sich nicht in einem Außenbüro befinden. Verified Permissions leitet diese Attribute aus den Ansprüchen im ID-Token des Prinzipals ab.

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

Validierung von Kunden und Zielgruppen

Wenn Sie einem Richtlinienspeicher eine Identitätsquelle hinzufügen, verfügt Verified Permissions über Konfigurationsoptionen, mit denen überprüft wird, ob ID und Zugriffstoken wie vorgesehen verwendet werden. Diese Überprüfung erfolgt bei der Verarbeitung von `IsAuthorizedWithToken` und `BatchIsAuthorizedWithToken` API-Anfragen. Das Verhalten unterscheidet sich zwischen ID- und Zugriffstoken sowie zwischen Amazon Cognito- und OIDC-Identitätsquellen. Bei Anbietern von Amazon Cognito Cognito-Benutzerpools kann Verified Permissions die Client-ID sowohl in ID- als auch in Zugriffstoken validieren. Bei OIDC-Anbietern kann Verified Permissions die Client-ID in ID-Token und die Zielgruppe in Zugriffstoken validieren.

Eine Client-ID ist eine Kennung, die beispielsweise einer OAuth- oder OIDC-Anwendung zugeordnet ist, die mit dem Anbieter konfiguriert wurde. `1example23456789` Eine Zielgruppe ist beispielsweise ein URL-Pfad, der mit der beabsichtigten vertrauenden Partei oder dem Ziel der Zielanwendung verknüpft ist. `https://myapplication.example.com` Der `aud` Anspruch bezieht sich nicht immer auf die Zielgruppe.

Verified Permissions führt die Überprüfung der Identitätsquelle, der Zielgruppe und des Kunden wie folgt durch:

Amazon Cognito

- Amazon Cognito Cognito-ID-Token haben einen `aud` Anspruch, der die App-Client-ID enthält. Zugriffstoken haben einen `client_id` Anspruch, der auch die App-Client-ID enthält.

Wenn Sie in Ihrer Identitätsquelle einen oder mehrere Werte für die Validierung von Client-Anwendungen eingeben, vergleicht Verified Permissions diese Liste von App-Client-IDs mit dem `aud` ID-Token-Anspruch oder dem `client_id` Zugriffstoken-Anspruch. Verified Permissions validiert keine Zielgruppen-URL einer vertrauenden Partei für Amazon Cognito Cognito-Identitätsquellen.

OIDC

Für OIDC-ID-Token gibt es einen `aud` Anspruch, der eine Liste von Client-IDs enthält. Zugriffstoken haben einen `aud` Anspruch, der die Zielgruppen-URL für das Token enthält. Zugriffstoken haben auch einen `client_id` Anspruch, der die beabsichtigte Client-ID enthält.

Sie können einen oder mehrere Werte für die Zielgruppenvalidierung bei einem OIDC-Anbieter eingeben. Wenn Sie einen Token-Typ eines ID-Tokens wählen, validiert Verified Permissions die Client-ID und überprüft, ob mindestens ein Mitglied der Client-IDs im `aud` Antrag einem Wert für die Zielgruppenvalidierung entspricht.

Verified Permissions validiert die Zielgruppe auf Zugriffstoken und überprüft, ob der `aud` Anspruch einem Zielgruppenvalidierungswert entspricht. Dieser Zugriffstoken-Wert stammt hauptsächlich aus dem `aud` Anspruch, kann aber auch aus dem `cid` `client_id` OR-Anspruch stammen, wenn kein `aud` Anspruch besteht. Erkundigen Sie sich bei Ihrem IdP nach dem richtigen Zielgruppen-Claim und Format.

Ein Beispiel für den Wert für die Zielgruppenvalidierung eines ID-Tokens ist`1example23456789`.

Ein Beispiel für einen Wert für die Zielgruppenvalidierung eines Zugriffstokens ist`https://myapplication.example.com`.

Clientseitige Autorisierung für JWTs

Möglicherweise möchten Sie JSON-Webtoken in Ihrer Anwendung verarbeiten und deren Ansprüche an Verified Permissions weiterleiten, ohne eine Identitätsquelle für den Richtlinienspeicher zu verwenden. Sie können Ihre Entitätsattribute aus einem JSON Web Token (JWT) extrahieren und in verifizierte Berechtigungen umwandeln.

Dieses Beispiel zeigt, wie Sie verifizierte Berechtigungen von einem OIDC-IdP aufrufen könnten[1].

```
async function authorizeUsingJwtToken(jwtToken) {

    const payload = await verifier.verify(jwtToken);
```

```
    var principalEntity = {
        entityType: "PhotoFlash::User", // the application needs to fill in the
 relevant user type
        entityId: payload["sub"], // the application need to use the claim that
 represents the user-id
    };
    var resourceEntity = {
        entityType: "PhotoFlash::Photo", //the application needs to fill in the
 relevant resource type
        entityId: "jane_photo_123.jpg", // the application needs to fill in the
 relevant resource id
    };
    var action = {
        actionType: "PhotoFlash::Action", //the application needs to fill in the
 relevant action id
        actionId: "GetPhoto", //the application needs to fill in the relevant action
 type
    };
    var entities = {
        entityList: [],
    };
    entities.entityList.push(...getUserEntitiesFromToken(payload));
    var policyStoreId = "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 === '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 need to use the claim that
 represents the user-id
    }
  };
  return [entityItem];
}
```

[1] Dieses Codebeispiel verwendet die Bibliothek aws-jwt-verify zur Überprüfung von JWTs, die mit OIDC-Compatible signiert wurden. IdPs

Identitätsquellen für Amazon Verified Permissions erstellen

Mit dem folgenden Verfahren wird einem vorhandenen Richtlinienspeicher eine Identitätsquelle hinzugefügt. Nachdem Sie Ihre Identitätsquelle hinzugefügt haben, müssen Sie Ihrem Schema Attribute hinzufügen.

Sie können auch eine Identitätsquelle erstellen, wenn Sie in der Konsole „Verifizierte Berechtigungen“ einen neuen Richtlinienspeicher erstellen. In diesem Prozess können Sie die Ansprüche in Ihren Identitätsquellen-Token automatisch in Entitätsattribute importieren. Wählen Sie die Option Geführte Einrichtung oder Einrichtung mit API Gateway und einem Identitätsanbieter. Mit diesen Optionen werden auch erste Richtlinien erstellt.

> **Note**
>
> Identitätsquellen sind im Navigationsbereich auf der linken Seite erst verfügbar, wenn Sie einen Richtlinienspeicher erstellt haben. Identitätsquellen, die Sie erstellen, sind dem aktuellen Richtlinienspeicher zugeordnet.

Sie können den Hauptentitätstyp weglassen, wenn Sie eine Identitätsquelle mit create-identity-source in AWS CLI oder CreateIdentity Source in der Verified Permissions API erstellen. Ein leerer Entitätstyp erstellt jedoch eine Identitätsquelle mit dem Entitätstyp. `AWS::Cognito` Dieser Entitätsname ist nicht mit dem Richtlinienspeicherschema kompatibel. Um Amazon Cognito Cognito-Identitäten in Ihr Policy Store-Schema zu integrieren, müssen Sie den Prinzipal-Entitätstyp auf eine unterstützte Policy Store-Entität festlegen.

Themen

- Amazon Cognito Cognito-Identitätsquelle
- OIDC-Identitätsquelle

Amazon Cognito Cognito-Identitätsquelle

AWS Management Console

So erstellen Sie eine Identitätsquelle für Amazon Cognito Cognito-Benutzerpools

1. Öffnen Sie die Konsole Verified Permissions unter https://console.aws.amazon.com/verifiedpermissions/. Wählen Sie Ihren Richtlinienspeicher aus.
2. Wählen Sie im Navigationsbereich auf der linken Seite Identitätsquellen aus.
3. Wählen Sie Identitätsquelle erstellen aus.
4. Wählen Sie in den Cognito-Benutzerpooldetails die Benutzerpool-ID für Ihre Identitätsquelle aus AWS-Region und geben Sie sie ein.
5. Wählen Sie in der Prinzipalkonfiguration einen Prinzipaltyp für die Identitätsquelle aus. Identitäten aus den verbundenen Amazon Cognito Cognito-Benutzerpools werden dem ausgewählten Prinzipaltyp zugeordnet.
6. Wählen Sie in der Gruppenkonfiguration die Option Cognito-Gruppe verwenden aus, wenn Sie den `cognito:groups` Benutzerpoolanspruch zuordnen möchten. Wählen Sie einen Entitätstyp, der dem Prinzipaltyp übergeordnet ist.
7. Wählen Sie unter Validierung von Client-Anwendungen aus, ob Client-Anwendungs-IDs validiert werden sollen.
   - Um die IDs der Client-Anwendungen zu überprüfen, wählen Sie Nur Token akzeptieren, deren Client-Anwendungs-IDs übereinstimmen. Wählen Sie für jede zu validierende Client-Anwendungs-ID neue Client-Anwendungs-ID hinzufügen aus. Um eine hinzugefügte Client-Anwendungs-ID zu entfernen, klicken Sie neben der Client-Anwendungs-ID auf Entfernen.
   - Wählen Sie Client-Anwendungs-IDs nicht validieren, wenn Sie die Client-Anwendungs-IDs nicht validieren möchten.
8. Wählen Sie Identitätsquelle erstellen aus.
9. Bevor Sie in Ihren Cedar-Richtlinien auf Attribute verweisen können, die Sie aus Identitäts- oder Zugriffstoken extrahieren, müssen Sie Ihr Schema aktualisieren, damit Cedar weiß, welche Art von Prinzipal Ihre Identitätsquelle erstellt. Diese Ergänzung zum Schema muss die Attribute enthalten, auf die Sie in Ihren Cedar-Richtlinien verweisen möchten. Weitere Informationen zur Zuordnung von Amazon Cognito-Tokenattributen zu Cedar-Hauptattributen finden Sie unterArbeiten mit Identitätsquellen in Schemas und Richtlinien.

Wenn Sie einen API-verknüpften Richtlinienspeicher erstellen, fragt Verified Permissions Ihren Benutzerpool nach Benutzerattributen ab und erstellt ein Schema, in dem Ihr Prinzipaltyp mit Benutzerpool-Attributen aufgefüllt wird.

AWS CLI

So erstellen Sie eine Identitätsquelle für Amazon Cognito Cognito-Benutzerpools

Sie können eine Identitätsquelle erstellen, indem Sie den Vorgang CreateIdentitySource verwenden. Im folgenden Beispiel wird eine Identitätsquelle erstellt, die auf authentifizierte Identitäten aus einem Amazon Cognito Cognito-Benutzerpool zugreifen kann.

Die folgende `config.txt` Datei enthält die Details des Amazon Cognito Cognito-Benutzerpools, der vom Parameter --configuration im `create-identity-source` Befehl verwendet werden kann.

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

Befehl:

```
$ 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"
}
```

Bevor Sie in Ihren Cedar-Richtlinien auf Attribute verweisen können, die Sie aus Identitäts- oder Zugriffstoken extrahieren, müssen Sie Ihr Schema aktualisieren, damit Cedar weiß, welche Art von Prinzipal Ihre Identitätsquelle erstellt. Diese Ergänzung zum Schema muss die Attribute enthalten, auf die Sie in Ihren Cedar-Richtlinien verweisen möchten. Weitere Informationen zur Zuordnung von Amazon Cognito-Tokenattributen zu Cedar-Hauptattributen finden Sie unterArbeiten mit Identitätsquellen in Schemas und Richtlinien.

Wenn Sie einen API-verknüpften Richtlinienspeicher erstellen, fragt Verified Permissions Ihren Benutzerpool nach Benutzerattributen ab und erstellt ein Schema, in dem Ihr Prinzipaltyp mit Benutzerpool-Attributen aufgefüllt wird.

Weitere Informationen zur Verwendung von Zugriffs- und Identitätstoken von Amazon Cognito für authentifizierte Benutzer in Verified Permissions finden Sie unter Authorization with Amazon Verified Permissions im Amazon Cognito Developer Guide.

OIDC-Identitätsquelle

AWS Management Console

So erstellen Sie eine OpenID Connect (OIDC) -Identitätsquelle

1. Öffnen Sie die Konsole „Verified Permissions“ unter https://console.aws.amazon.com/verifiedpermissions/. Wählen Sie Ihren Richtlinienspeicher aus.
2. Wählen Sie im Navigationsbereich auf der linken Seite Identitätsquellen aus.
3. Wählen Sie Identitätsquelle erstellen aus.
4. Wählen Sie Externer OIDC-Anbieter.
5. Geben Sie unter Aussteller-URL die URL Ihres OIDC-Ausstellers ein. Dies ist der Dienstendpunkt, der beispielsweise den Autorisierungsserver, Signaturschlüssel und andere Informationen über Ihren Anbieter bereitstellt. `https://auth.example.com` Ihre Aussteller-URL muss ein OIDC-Discovery-Dokument unter hosten. `/.well-known/openid-configuration`
6. Wählen Sie unter Tokentyp den Typ des OIDC JWT aus, den Ihre Anwendung zur Autorisierung einreichen soll. Weitere Informationen finden Sie unter Arbeiten mit Identitätsquellen in Schemas und Richtlinien.
7. Wählen Sie unter Benutzer- und Gruppenansprüche eine Benutzerentität und einen Benutzeranspruch für die Identitätsquelle aus. Die Benutzerentität ist eine Entität in Ihrem

Richtlinienspeicher, auf die Sie Benutzer Ihres OIDC-Anbieters verweisen möchten. Bei dem Benutzeranspruch handelt es sich in der Regel `sub` um einen Anspruch aus Ihrer ID oder Ihrem Zugriffstoken, das die eindeutige Kennung für die zu bewertende Entität enthält. Identitäten des verbundenen OIDC-IdP werden dem ausgewählten Prinzipaltyp zugeordnet.

8. Wählen Sie unter Benutzer- und Gruppenansprüche eine Gruppenentität und einen Gruppenanspruch für die Identitätsquelle aus. Die Gruppenentität ist der Benutzerentität übergeordnet. Gruppenansprüche werden dieser Entität zugeordnet. Bei dem Gruppenanspruch handelt es sich in der Regel `groups` um einen Anspruch aus Ihrer ID oder Ihrem Zugriffstoken, der eine Zeichenfolge, JSON oder eine durch Leerzeichen getrennte Zeichenfolge mit Benutzergruppennamen für die auszuwertende Entität enthält. Identitäten des verbundenen OIDC-IdP werden dem ausgewählten Prinzipaltyp zugeordnet.
9. Geben Sie unter Zielgruppenvalidierung die Client-IDs oder Zielgruppen-URLs ein, die Ihr Richtlinienspeicher gegebenenfalls in Autorisierungsanfragen akzeptieren soll.
10. Wählen Sie „Identitätsquelle erstellen“.
11. Aktualisieren Sie Ihr Schema, damit Cedar weiß, welchen Prinzipaltyp Ihre Identitätsquelle erstellt. Dieser Zusatz zum Schema muss die Attribute enthalten, auf die Sie in Ihren Cedar-Richtlinien verweisen möchten. Weitere Informationen zur Zuordnung von Amazon Cognito-Tokenattributen zu Cedar-Hauptattributen finden Sie unter Arbeiten mit Identitätsquellen in Schemas und Richtlinien.

    Wenn Sie einen API-verknüpften Richtlinienspeicher erstellen, fragt Verified Permissions Ihren Benutzerpool nach Benutzerattributen ab und erstellt ein Schema, in dem Ihr Prinzipaltyp mit Benutzerpool-Attributen aufgefüllt wird.

AWS CLI

Um eine OIDC-Identitätsquelle zu erstellen

Sie können eine Identitätsquelle mithilfe der Operation CreateIdentitySource erstellen. Im folgenden Beispiel wird eine Identitätsquelle erstellt, die auf authentifizierte Identitäten aus einem Amazon Cognito Cognito-Benutzerpool zugreifen kann.

Die folgende `config.txt` Datei enthält die Details eines OIDC-IdP zur Verwendung durch den `--configuration` Parameter des Befehls. `create-identity-source` In diesem Beispiel wird eine OIDC-Identitätsquelle für ID-Token erstellt.

```
{
```

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

Die folgende `config.txt` Datei enthält die Details eines OIDC-IdP zur Verwendung durch den `--configuration` Parameter des Befehls. `create-identity-source` In diesem Beispiel wird eine OIDC-Identitätsquelle für Zugriffstoken erstellt.

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

Befehl:

```
$ 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"
}
```

Bevor Sie in Ihren Cedar-Richtlinien auf Attribute verweisen können, die Sie aus Identitäts- oder Zugriffstoken extrahieren, müssen Sie Ihr Schema aktualisieren, damit Cedar den Prinzipaltyp kennt, den Ihre Identitätsquelle erstellt. Diese Ergänzung zum Schema muss die Attribute enthalten, auf die Sie in Ihren Cedar-Richtlinien verweisen möchten. Weitere Informationen zur Zuordnung von Amazon Cognito-Tokenattributen zu Cedar-Hauptattributen finden Sie unterArbeiten mit Identitätsquellen in Schemas und Richtlinien.

Wenn Sie einen API-verknüpften Richtlinienspeicher erstellen, fragt Verified Permissions Ihren Benutzerpool nach Benutzerattributen ab und erstellt ein Schema, in dem Ihr Prinzipaltyp mit Benutzerpool-Attributen aufgefüllt wird.

Identitätsquellen für Amazon Verified Permissions bearbeiten

Sie können einige Parameter Ihrer Identitätsquelle bearbeiten, nachdem Sie sie erstellt haben. Wenn Ihr Richtlinienspeicher-Schema Ihren Identitätsquellenattributen entspricht, beachten Sie, dass Sie Ihr Schema separat aktualisieren müssen, um die Änderungen widerzuspiegeln, die Sie an Ihrer Identitätsquelle vornehmen.

Themen

- Identitätsquelle für Amazon Cognito Cognito-Benutzerpools
- OpenID Connect (OIDC) -Identitätsquelle

Identitätsquelle für Amazon Cognito Cognito-Benutzerpools

AWS Management Console

So aktualisieren Sie die Identitätsquelle eines Amazon Cognito Cognito-Benutzerpools

1. Öffnen Sie die Konsole Verified Permissions unter https://console.aws.amazon.com/verifiedpermissions/. Wählen Sie Ihren Richtlinienspeicher aus.

2. Wählen Sie im Navigationsbereich auf der linken Seite Identitätsquellen aus.
3. Wählen Sie die ID der Identitätsquelle aus, die Sie bearbeiten möchten.
4. Wählen Sie Bearbeiten aus.
5. Wählen Sie in den Cognito-Benutzerpooldetails die Benutzerpool-ID für Ihre Identitätsquelle aus AWS-Region und geben Sie sie ein.
6. In den Prinzipaldetails können Sie den Prinzipaltyp für die Identitätsquelle aktualisieren. Identitäten aus den verbundenen Amazon Cognito Cognito-Benutzerpools werden dem ausgewählten Prinzipaltyp zugeordnet.
7. Wählen Sie in der Gruppenkonfiguration die Option Cognito-Gruppe verwenden aus, wenn Sie den `cognito:groups` Benutzerpoolanspruch zuordnen möchten. Wählen Sie einen Entitätstyp, der dem Prinzipaltyp übergeordnet ist.
8. Wählen Sie unter Validierung von Client-Anwendungen aus, ob Client-Anwendungs-IDs validiert werden sollen.

   - Um die IDs der Client-Anwendungen zu überprüfen, wählen Sie Nur Token akzeptieren, deren Client-Anwendungs-IDs übereinstimmen. Wählen Sie für jede zu validierende Client-Anwendungs-ID neue Client-Anwendungs-ID hinzufügen aus. Um eine hinzugefügte Client-Anwendungs-ID zu entfernen, klicken Sie neben der Client-Anwendungs-ID auf Entfernen.
   - Wählen Sie Client-Anwendungs-IDs nicht validieren, wenn Sie die Client-Anwendungs-IDs nicht validieren möchten.
9. Wählen Sie Änderungen speichern aus.
10. Wenn Sie den Prinzipaltyp für die Identitätsquelle geändert haben, müssen Sie Ihr Schema aktualisieren, damit es den aktualisierten Prinzipaltyp korrekt wiedergibt.

Sie können eine Identitätsquelle löschen, indem Sie das Optionsfeld neben einer Identitätsquelle auswählen und dann Identitätsquelle löschen auswählen. Geben Sie `delete` etwas in das Textfeld ein und wählen Sie dann Identitätsquelle löschen aus, um das Löschen der Identitätsquelle zu bestätigen.

AWS CLI

So aktualisieren Sie die Identitätsquelle eines Amazon Cognito Cognito-Benutzerpools

Sie können eine Identitätsquelle aktualisieren, indem Sie den Vorgang UpdateIdentitySource verwenden. Im folgenden Beispiel wird die angegebene Identitätsquelle aktualisiert, sodass sie einen anderen Amazon Cognito Cognito-Benutzerpool verwendet.

Die folgende `config.txt` Datei enthält die Details des Amazon Cognito Cognito-Benutzerpools, der vom Parameter --configuration im `create-identity-source` Befehl verwendet werden kann.

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

Befehl:

```
$ 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"
}
```

Wenn Sie den Prinzipaltyp für die Identitätsquelle ändern, müssen Sie Ihr Schema aktualisieren, damit es den aktualisierten Prinzipaltyp korrekt wiedergibt.

OpenID Connect (OIDC) -Identitätsquelle

AWS Management Console

Um eine OIDC-Identitätsquelle zu aktualisieren

1. Öffnen Sie die Konsole Verified Permissions unter https://console.aws.amazon.com/verifiedpermissions/. Wählen Sie Ihren Richtlinienspeicher aus.
2. Wählen Sie im Navigationsbereich auf der linken Seite Identitätsquellen aus.
3. Wählen Sie die ID der Identitätsquelle aus, die Sie bearbeiten möchten.

4. Wählen Sie Bearbeiten aus.
5. Ändern Sie in den OIDC-Anbieterdetails die Aussteller-URL nach Bedarf.
6. Ändern Sie unter Tokenansprüche den Schemaattributen zuordnen die Verknüpfungen zwischen Benutzer- und Gruppenansprüchen sowie den Entitätstypen des Richtlinienspeichers nach Bedarf. Nachdem Sie die Entitätstypen geändert haben, müssen Sie Ihre Richtlinien und Schemaattribute aktualisieren, damit sie für die neuen Entitätstypen gelten.
7. Fügen Sie in der Zielgruppenvalidierung Zielgruppenwerte hinzu oder entfernen Sie sie, die Sie erzwingen möchten.
8. Wählen Sie Änderungen speichern aus.

Sie können eine Identitätsquelle löschen, indem Sie das Optionsfeld neben einer Identitätsquelle auswählen und dann Identitätsquelle löschen auswählen. Geben Sie `delete` etwas in das Textfeld ein und wählen Sie dann Identitätsquelle löschen aus, um das Löschen der Identitätsquelle zu bestätigen.

AWS CLI

Um eine OIDC-Identitätsquelle zu aktualisieren

Sie können eine Identitätsquelle aktualisieren, indem Sie den Vorgang UpdateIdentitySource verwenden. Im folgenden Beispiel wird die angegebene Identitätsquelle aktualisiert, sodass sie einen anderen OIDC-Anbieter verwendet.

Die folgende `config.txt` Datei enthält die Details des Amazon Cognito Cognito-Benutzerpools, der vom Parameter --configuration im `create-identity-source` Befehl verwendet werden kann.

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

```
            "groupClaim": "groups",
            "groupEntityType": "MyCorp::UserGroup"
        }
    }
}
```

Befehl:

```
$ 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"
}
```

Wenn Sie den Prinzipaltyp für die Identitätsquelle ändern, müssen Sie Ihr Schema aktualisieren, damit es den aktualisierten Prinzipaltyp korrekt wiedergibt.

Arbeiten mit Identitätsquellen in Schemas und Richtlinien

Möglicherweise möchten Sie einem Richtlinienspeicher eine Identitätsquelle hinzufügen und Anbieteransprüche Ihrem Policy-Store-Schema zuordnen. Sie können diesen Vorgang automatisieren oder Ihr Schema manuell aktualisieren. Dieser Abschnitt des Benutzerhandbuchs enthält die folgenden Informationen:

- Wann können Sie Attribute automatisch in ein Policy-Store-Schema eintragen
- So verwenden Sie Amazon Cognito- und OIDC-Token-Ansprüche in Ihren Richtlinien für verifizierte Berechtigungen
- Wie erstelle ich manuell ein Schema für eine Identitätsquelle

API-verknüpfte Richtlinienspeicher und Richtlinienspeicher mit einer Identitätsquelle mithilfe von Guided Setup erfordern keine manuelle Zuordnung von Identitätstokenattributen (ID) zum Schema. Sie können verifizierte Berechtigungen mit den Attributen in Ihrem Benutzerpool oder mit OIDC-Token angeben und ein Schema erstellen, das mit Benutzerattributen gefüllt ist. Bei der Autorisierung von ID-Tokens ordnet Verified Permissions Ansprüche Attributen einer Prinzipalentität zu. Unter den

folgenden Bedingungen müssen Sie Amazon Cognito Cognito-Token möglicherweise manuell Ihrem Schema zuordnen:

- Sie haben einen leeren Richtlinienspeicher oder einen Richtlinienspeicher aus einem Beispiel erstellt.
- Sie möchten die Verwendung von Zugriffstoken über die rollenbasierte Zugriffskontrolle (RBAC) hinaus erweitern.
- Sie erstellen Richtlinienspeicher mit der REST-API für verifizierte Berechtigungen, einem AWS SDK oder dem. AWS CDK

Um Amazon Cognito oder einen OIDC-Identitätsanbieter (IdP) als Identitätsquelle in Ihrem Richtlinienspeicher für verifizierte Berechtigungen zu verwenden, müssen Sie Anbieterattribute in Ihrem Schema haben. Wenn Sie Ihren Richtlinienspeicher so erstellt haben, dass Ihr Schema automatisch anhand der Anbieterinformationen in einem ID-Token aufgefüllt wird, sind Sie bereit, Richtlinien zu schreiben. Wenn Sie einen Richtlinienspeicher ohne Schema für Ihre Identitätsquelle erstellen, müssen Sie dem Schema Anbieterattribute hinzufügen. Ihr Schema muss den Entitäten entsprechen, die Anbieter-Tokens in IsAuthorizedWithTokenBatchIsAuthorizedWithToken-API-Anfragen erstellen. Anschließend können Sie Richtlinien mithilfe von Attributen aus dem Anbietertoken schreiben.

Weitere Informationen zur Verwendung von Amazon Cognito ID und Zugriffstoken für authentifizierte Benutzer in Verified Permissions finden Sie unter Authorization with Amazon Verified Permissions im Amazon Cognito Developer Guide.

Themen

Wissenswertes über Schema-Mapping

Die Attributzuweisung unterscheidet sich zwischen Tokentypen

Bei der Autorisierung von Zugriffstoken ordnet Verified Permissions Ansprüche dem Kontext zu. Bei der Autorisierung von ID-Tokens ordnet Verified Permissions Ansprüche den Hauptattributen

zu. Bei Richtlinienspeichern, die Sie in der Verified Permissions-Konsole erstellen, haben Sie nur leere Richtlinienspeicher und Beispiel-Richtlinienspeicher, sodass Sie keine Identitätsquelle haben und Sie Ihr Schema mit Benutzerpool-Attributen für die ID-Token-Autorisierung auffüllen müssen. Die Autorisierung mit Zugriffstoken basiert auf der rollenbasierten Zugriffskontrolle (RBAC) mit Gruppenmitgliedschaftsansprüchen und ordnet andere Ansprüche nicht automatisch dem Richtlinienspeicherschema zu.

Identitätsquellenattribute sind nicht erforderlich

Wenn Sie in der Konsole „Verified Permissions“ eine Identitätsquelle erstellen, werden keine Attribute als erforderlich markiert. Dadurch wird verhindert, dass fehlende Ansprüche zu Validierungsfehlern bei Autorisierungsanfragen führen. Sie können Attribute nach Bedarf auf erforderlich setzen, sie müssen jedoch in allen Autorisierungsanfragen vorhanden sein.

RBAC benötigt keine Attribute im Schema

Schemas für Identitätsquellen hängen von den Entitätszuordnungen ab, die Sie beim Hinzufügen Ihrer Identitätsquelle vornehmen. Eine Identitätsquelle ordnet einen Anspruch einem Benutzerentiätstyp und einen Anspruch einem Gruppen-Entitätstyp zu. Diese Entitätszuordnungen sind der Kern einer Identitätsquellenkonfiguration. Mit diesen Mindestinformationen können Sie Richtlinien schreiben, die Autorisierungsaktionen für bestimmte Benutzer und bestimmte Gruppen, denen Benutzer möglicherweise angehören, in einem Modell der rollenbasierten Zugriffskontrolle (RBAC) ausführen. Durch das Hinzufügen von Token-Ansprüchen zum Schema wird der Autorisierungsbereich Ihres Richtlinienspeichers erweitert. Benutzerattribute aus ID-Tokens enthalten Informationen über Benutzer, die zur ABAC-Autorisierung (attribute-Based Access Control) beitragen können. Kontextattribute von Zugriffstoken enthalten Informationen wie OAuth 2.0-Bereiche, die zusätzliche Informationen zur Zugriffskontrolle von Ihrem Anbieter bereitstellen können, aber zusätzliche Schemaänderungen erfordern.

Die Optionen Mit API Gateway und einer Identitätsquelle einrichten und Geführte Einrichtung in der Konsole Verified Permissions weisen dem Schema ID-Token-Ansprüche zu. Dies ist bei Ansprüchen auf Zugriffstoken nicht der Fall. Um Ihrem Schema Ansprüche auf Zugriffstoken hinzuzufügen, die nicht zu Gruppen gehören, müssen Sie Ihr Schema im JSON-Modus bearbeiten und CommonTypes-Attribute hinzufügen. Weitere Informationen finden Sie unter Zugriffstoken zuordnen.

OIDC-Gruppen behaupten, dass mehrere Formate unterstützt werden

Wenn Sie einen OIDC-Anbieter hinzufügen, können Sie den Namen des Gruppenanspruchs in ID- oder Zugriffstoken auswählen, den Sie der Gruppenmitgliedschaft eines Benutzers in Ihrem

Richtlinienspeicher zuordnen möchten. Verifizierte Berechtigungen erkennen Gruppenansprüche in den folgenden Formaten an:

1. Zeichenfolge ohne Leerzeichen: `"groups": "MyGroup"`
2. Durch Leerzeichen getrennte Liste:. `"groups": "MyGroup1 MyGroup2 MyGroup3"` Jede Zeichenfolge ist eine Gruppe.
3. JSON-Liste (durch Kommas getrennt): `"groups": ["MyGroup1", "MyGroup2", "MyGroup3"]`

> **Note**
>
> Verified Permissions interpretiert jede Zeichenfolge in einem durch Leerzeichen getrennten Gruppenanspruch als separate Gruppe. Um einen Gruppennamen mit einem Leerzeichen als einzelne Gruppe zu interpretieren, ersetzen oder entfernen Sie das Leerzeichen im Anspruch. Formatieren Sie beispielsweise eine Gruppe mit dem Namen `My GroupMyGroup`.

Wählen Sie einen Tokentyp

Wie Ihr Richtlinienspeicher mit Ihrer Identitätsquelle zusammenarbeitet, hängt von einer wichtigen Entscheidung bei der Konfiguration der Identitätsquelle ab: ob Sie ID- oder Zugriffstoken verarbeiten. Bei einem Amazon Cognito Cognito-Identitätsanbieter haben Sie die Wahl zwischen dem Tokentyp, wenn Sie einen API-verknüpften Richtlinienspeicher erstellen. Wenn Sie einen API-verknüpften Richtlinienspeicher erstellen, müssen Sie wählen, ob Sie die Autorisierung für ID- oder Zugriffstoken einrichten möchten. Diese Informationen wirken sich auf die Schemaattribute aus, die Verified Permissions auf Ihren Richtlinienspeicher anwendet, und auf die Syntax des Lambda-Autorisierers für Ihre API-Gateway-API. Bei einem OIDC-Anbieter müssen Sie beim Hinzufügen der Identitätsquelle einen Tokentyp auswählen. Sie können zwischen ID und Zugriffstoken wählen, und Ihre Wahl schließt die Verarbeitung des nicht ausgewählten Tokentyps in Ihrem Richtlinienspeicher aus. Insbesondere, wenn Sie von der automatischen Zuordnung von ID-Token-Ansprüchen zu Attributen in der Verified Permissions-Konsole profitieren möchten, sollten Sie sich frühzeitig für den Tokentyp entscheiden, den Sie verarbeiten möchten, bevor Sie Ihre Identitätsquelle erstellen. Das Ändern des Tokentyps erfordert einen erheblichen Aufwand, um Ihre Richtlinien und Ihr Schema umzugestalten. In den folgenden Themen wird die Verwendung von ID- und Zugriffstoken mit Richtlinienspeichern beschrieben.

Der Cedar-Parser benötigt für einige Zeichen Klammern

Richtlinien verweisen normalerweise auf Schemaattribute in einem Format wie`principal.username`. Bei den meisten nicht-alphanumerischen Zeichen wie`:`, oder`.`, `/` die in Namen von Token-Ansprüchen vorkommen können, kann Verified Permissions einen Bedingungswert wie oder nicht analysieren. `principal.cognito:groups` `context.ip-address` Stattdessen müssen Sie diese Bedingungen mit Klammernotation im jeweiligen Format `principal["cognito:username"]` oder `context["ip-address"]` formatieren. Der Unterstrich _ ist ein gültiges Zeichen in Anspruchsnamen und die einzige Ausnahme von dieser Anforderung, die nicht alphanumerisch ist.

Ein teilweises Beispielschema für ein Hauptattribut dieses Typs sieht wie folgt aus:

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

Ein teilweises Beispielschema für ein Kontextattribut dieses Typs sieht wie folgt aus:

```
"GetOrder": {
   "memberOf": [],
   "appliesTo": {
      "resourceTypes": [
         "Order"
      ],
      "context": {
         "type": "Record",
         "attributes": {
```

```
            "ip-address": {
               "required": false,
               "type": "String"
            }
   }
   },
      "principalTypes": [
         "User"
      ]
   }
}
```

Eine Beispielrichtlinie für Attribute, die anhand dieses Schemas validiert werden, sieht wie folgt aus:

```
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.*"
};
```

Zuordnen von ID-Token zum Schema

Verified Permissions verarbeitet ID-Token-Ansprüche als Attribute des Benutzers: seine Namen und Titel, seine Gruppenzugehörigkeit, seine Kontaktinformationen. ID-Token sind in einem ABAC-Autorisierungsmodell (attribute-based access control) am nützlichsten. Wenn Sie möchten, dass Verified Permissions den Zugriff auf Ressourcen basierend darauf analysiert, wer die Anfrage stellt, wählen Sie ID-Token als Identitätsquelle.

Amazon Cognito Cognito-ID-Token

Amazon Cognito ID-Token funktionieren mit den meisten OIDC-Rely-Party-Bibliotheken. Sie erweitern die Funktionen von OIDC um zusätzliche Ansprüche. Ihre Anwendung kann den Benutzer mit API-Authentifizierungsoperationen für Amazon Cognito Cognito-Benutzerpools oder mit der gehosteten Benutzerpool-Benutzeroberfläche authentifizieren. Weitere Informationen finden Sie unter Using the API and Endpoints im Amazon Cognito Developer Guide.

Nützliche Angaben in Amazon Cognito Cognito-ID-Tokens

cognito:username und *preferred_username*

Varianten des Benutzernamens.

sub

Die eindeutige Benutzerkennung (UUID) des Benutzers

Ansprüche mit einem Präfix *custom:*

Ein Präfix für benutzerdefinierte Benutzerpool-Attribute wie`custom:employmentStoreCode`.

Standardansprüche

Standardansprüche von OIDC wie `email` und. `phone_number` Weitere Informationen finden Sie unter Standardansprüche in OpenID Connect Core 1.0, in denen Errata Set 2 enthalten ist.

cognito:groups

Gruppenmitgliedschaften eines Benutzers. In einem Autorisierungsmodell, das auf der rollenbasierten Zugriffskontrolle (RBAC) basiert, beschreibt dieser Anspruch die Rollen, die Sie in Ihren Richtlinien bewerten können.

Vorübergehende Ansprüche

Ansprüche, die nicht Eigentum des Benutzers sind, aber zur Laufzeit durch einen Lambda-Trigger vor der Token-Generierung aus dem Benutzerpool hinzugefügt werden. Vorübergehende Ansprüche ähneln Standardansprüchen, liegen aber außerhalb des Standards, z. B. `tenant` oder. `department`

In Richtlinien, die auf Amazon Cognito-Attribute verweisen, die über ein `:` Trennzeichen verfügen, verweisen Sie auf die Attribute im Format`principal["cognito:username"]`. Der Rollenanspruch `cognito:groups` stellt eine Ausnahme von dieser Regel dar. Verified Permissions ordnet den Inhalt dieses Anspruchs den übergeordneten Entitäten der Benutzerentität zu.

Weitere Informationen zur Struktur von ID-Token aus Amazon Cognito-Benutzerpools finden Sie unter Verwenden des ID-Tokens im Amazon Cognito Developer Guide.

Das folgende Beispiel für ein ID-Token hat jeden der vier Attributtypen. Es umfasst den Amazon Cognito-spezifischen Antrag`cognito:username`, den benutzerdefinierten Antrag, den

Standardantrag `custom:employmentStoreCode` und den `email` vorübergehenden Anspruch. `tenant`

```
{
    "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"
}
```

Wenn Sie mit Ihrem Amazon Cognito Cognito-Benutzerpool eine Identitätsquelle erstellen, geben Sie den Typ der Prinzipalentität an, mit `IsAuthorizedWithToken` der Verified Permissions in Autorisierungsanfragen generiert. Ihre Richtlinien können dann im Rahmen der Bewertung dieser Anfrage die Attribute dieses Prinzipals testen. Ihr Schema definiert den Prinzipaltyp und die Attribute für eine Identitätsquelle, und dann können Sie in Ihren Cedar-Richtlinien darauf verweisen.

Sie geben auch den Typ der Gruppenentität an, den Sie aus dem Anspruch der ID-Token-Gruppen ableiten möchten. In Autorisierungsanfragen ordnet Verified Permissions jedes Mitglied des Gruppenanspruchs diesem Gruppen-Entitätstyp zu. In Richtlinien können Sie auf diese Gruppenentität als Principal verweisen.

Das folgende Beispiel zeigt, wie Sie die Attribute aus dem Beispiel-Identitätstoken in Ihrem Verified Permissions-Schema wiedergeben können. Weitere Informationen zur Bearbeitung Ihres Schemas finden Sie unterBearbeiten von Schemas im JSON-Modus. Wenn Ihre Identitätsquellenkonfiguration

den Prinzipaltyp angibt`User`, können Sie etwas Ähnliches wie das folgende Beispiel hinzufügen, um diese Attribute für Cedar verfügbar zu machen.

```
"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
         }
      }
   }
}
```

Nachdem Sie Ihr Schema aktualisiert haben, um die Amazon Cognito-Attribute widerzuspiegeln, können Sie Richtlinien erstellen, die auf die Attribute verweisen.

```
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-Token

Die Arbeit mit ID-Token von einem OIDC-Anbieter entspricht weitgehend der Arbeit mit Amazon Cognito Cognito-ID-Token. Der Unterschied liegt in den Behauptungen. Ihr IdP kann Standard-OIDC-Attribute oder ein benutzerdefiniertes Schema aufweisen. Wenn Sie in der Konsole „Verified Permissions“ einen neuen Richtlinienspeicher erstellen, können Sie eine OIDC-Identitätsquelle mit einem Beispiel-ID-Token hinzufügen oder Tokenansprüche manuell Benutzerattributen zuordnen. Da Verified Permissions das Attributschema Ihres IdP nicht kennt, müssen Sie diese Informationen angeben.

Weitere Informationen finden Sie unter Richtlinienspeicher für verifizierte Berechtigungen erstellen.

Im Folgenden finden Sie ein Beispielschema für einen Richtlinienspeicher mit einer OIDC-Identitätsquelle.

```
"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"
         }
      }
   }
}
```

Die folgende Richtlinie gilt für Mitglieder einer Gruppe in Ihrem OIDC-Anbieter.

```
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*"
};
```

Zugriffstoken zuordnen

Verified Permissions verarbeitet Ansprüche auf Zugriffstoken, die nicht von Gruppen als Attribute der Aktion oder als Kontext-Attribute beansprucht werden. Neben der Gruppenmitgliedschaft können die Zugriffstoken Ihres IdP Informationen über den API-Zugriff enthalten. Zugriffstoken sind in Autorisierungsmodellen nützlich, die eine rollenbasierte Zugriffskontrolle (RBAC) verwenden. Autorisierungsmodelle, die auf anderen Zugriffstoken-Ansprüchen als der Gruppenmitgliedschaft basieren, erfordern zusätzlichen Aufwand bei der Schemakonfiguration.

Zuordnen von Amazon Cognito Cognito-Zugriffstoken

Amazon Cognito Cognito-Zugriffstoken haben Ansprüche, die für die Autorisierung verwendet werden können:

Nützliche Angaben in Amazon Cognito Cognito-Zugriffstoken

`client_id`

Die ID der Client-Anwendung einer vertrauenden OIDC-Partei. Anhand der Client-ID kann Verified Permissions überprüfen, ob die Autorisierungsanfrage von einem autorisierten Client für den Richtlinienspeicher stammt. Bei der machine-to-machine (M2M-) Autorisierung autorisiert das anfordernde System eine Anfrage mit einem geheimen Client-Schlüssel und stellt die Client-ID und den Geltungsbereich als Autorisierungsnachweis zur Verfügung.

`scope`

Die OAuth 2.0-Bereiche, die die Zugriffsberechtigungen des Inhabers des Tokens darstellen.

`cognito:groups`

Die Gruppenmitgliedschaften eines Benutzers. In einem Autorisierungsmodell, das auf der rollenbasierten Zugriffskontrolle (RBAC) basiert, beschreibt dieser Anspruch die Rollen, die Sie in Ihren Richtlinien bewerten können.

Vorübergehende Ansprüche

Ansprüche, bei denen es sich nicht um eine Zugriffsberechtigung handelt, die aber zur Laufzeit durch einen Lambda-Trigger vor der Token-Generierung im Benutzerpool hinzugefügt werden. Vorübergehende Ansprüche ähneln Standardansprüchen, liegen aber außerhalb des Standards, z. B. `tenant` oder. `department` Die Anpassung von Zugriffstoken erhöht Ihre AWS Rechnung um zusätzliche Kosten.

Weitere Informationen zur Struktur von Zugriffstoken aus Amazon Cognito-Benutzerpools finden Sie unter Verwenden des Zugriffstokens im Amazon Cognito Developer Guide.

Ein Amazon Cognito Cognito-Zugriffstoken wird einem Kontextobjekt zugeordnet, wenn es an Verified Permissions übergeben wird. Auf Attribute des Zugriffstokens kann mit verwiesen werden. `context.token.`*`attribute_name`* Das folgende Beispiel für ein Zugriffstoken umfasst `client_id` sowohl die `scope` Ansprüche als auch.

```
{
    "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"
}
```

Das folgende Beispiel zeigt, wie Sie die Attribute aus dem Beispielzugriffstoken in Ihrem Verified Permissions-Schema wiedergeben können. Weitere Informationen zur Bearbeitung Ihres Schemas finden Sie unterBearbeiten von Schemas im JSON-Modus.

```
{
   "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"
         }
      }
   }
}
```

Nachdem Sie Ihr Schema aktualisiert haben, um die Amazon Cognito-Attribute widerzuspiegeln, können Sie Richtlinien erstellen, die auf die Attribute verweisen.

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

Zuordnung von OIDC-Zugriffstoken

Die meisten Zugriffstoken von externen OIDC-Anbietern stimmen eng mit den Zugriffstoken von Amazon Cognito überein. Ein OIDC-Zugriffstoken wird einem Kontextobjekt zugeordnet, wenn es an Verified Permissions übergeben wird. Auf Attribute des Zugriffstokens kann mit verwiesen werden. `context.token.`*`attribute_name`* Das folgende Beispiel für ein OIDC-Zugriffstoken enthält Beispiele für Basisansprüche.

```
{
    "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"
}
```

Das folgende Beispiel zeigt, wie Sie die Attribute aus dem Beispiel-Zugriffstoken in Ihrem Verified Permissions-Schema wiedergeben können. Weitere Informationen zur Bearbeitung Ihres Schemas finden Sie unterBearbeiten von Schemas im JSON-Modus.

```
{
   "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"
      }
   }
}
}
```

Nachdem Sie Ihr Schema aktualisiert haben, um die IdP-Attribute widerzuspiegeln, können Sie Richtlinien erstellen, die auf die Attribute verweisen.

```
permit(
    principal,
```

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

Alternative Schreibweise für durch Doppelpunkte getrennte Amazon Cognito-Ansprüche

Zu dem Zeitpunkt, als Verified Permissions gestartet wurde, beanspruchte das empfohlene Schema für Amazon Cognito Cognito-Token „Gefällt mir“ `cognito:groups` und diese durch Doppelpunkte getrennten Zeichenketten wurden `custom:store` konvertiert, um das `.` Zeichen als Hierarchie-Trennzeichen zu verwenden. Dieses Format wird als Punktnotation bezeichnet. Beispielsweise `cognito:groups` wurde `principal.cognito.groups` in Ihren Richtlinien ein Verweis auf aufgenommen. Sie können dieses Format zwar weiterhin verwenden, wir empfehlen Ihnen jedoch, Ihr Schema und Ihre Richtlinien in Klammern zu erstellen. In diesem Format `cognito:groups` wird ein Verweis auf `principal["cognito:groups"]` in Ihren Richtlinien enthalten. Automatisch generierte Schemas für Benutzerpool-ID-Token aus der Verified Permissions-Konsole verwenden Klammern.

Sie können weiterhin die Punktnotation in manuell erstellten Schemas und Richtlinien für Amazon Cognito Cognito-Identitätsquellen verwenden. Sie können die Punktnotation mit `:` oder anderen nicht alphanumerischen Zeichen in Schemas oder Richtlinien für keinen anderen Typ von OIDC-IdP verwenden.

Ein Schema für die Punktnotation verschachtelt jede Instanz eines `:` Zeichens als untergeordnetes Element der `cognito` oder `custom` ersten Phrase, wie im folgenden Beispiel gezeigt:

```
"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
      }
   }
}
```

Mit einem Schema in diesem Format können Sie eine Richtlinie mit Punktnotation wie im folgenden Beispiel erstellen:

```
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"
};
```

Entwerfen eines Autorisierungsmodells für Ihre Anwendung

Wenn Sie sich darauf vorbereiten, den Service Amazon Verified Permissions in einer Softwareanwendung zu nutzen, kann es schwierig sein, als ersten Schritt sofort mit dem Verfassen von Richtlinienerklärungen zu beginnen. Dies wäre vergleichbar mit dem Beginn der Entwicklung anderer Teile einer Anwendung, indem Sie SQL-Anweisungen oder API-Spezifikationen schreiben, bevor Sie vollständig entscheiden, was die Anwendung tun soll. Stattdessen sollten Sie mit einer Benutzererfahrung beginnen und sich ein klares Bild davon machen, was Endbenutzer bei der Verwaltung von Berechtigungen in der Anwendungsbenutzeroberfläche sehen sollten. Gehen Sie dann von dieser Erfahrung aus, um zu einem Implementierungsansatz zu gelangen.

Während Sie diese Arbeit erledigen, werden Sie feststellen, dass Sie sich Fragen stellen wie:

- Was sind meine Ressourcen? Haben sie Beziehungen zueinander? Befinden sich Dateien beispielsweise in einem Ordner?
- Welche Aktionen können Principals für jede Ressource ausführen?
- Wie erwerben Principals diese Berechtigungen?
- Möchten Sie, dass Ihre Endbenutzer aus vordefinierten Berechtigungen wie „Admin“, „Operator“ oder „“ wählen können, oder sollten sie Ad-hoc-Richtlinienerklärungen erstellen? ReadOnly Oder beides?
- Sollten Berechtigungen ressourcenübergreifend vererbt werden, z. B. Dateien, die Berechtigungen von einem übergeordneten Ordner erben?
- Welche Arten von Abfragen sind erforderlich, um die Benutzererfahrung zu verbessern? Müssen Sie beispielsweise alle Ressourcen auflisten, auf die ein Principal zugreifen kann, um die Startseite dieses Benutzers zu rendern?
- Können sich Benutzer versehentlich selbst von ihren eigenen Ressourcen ausschließen? Muss das vermieden werden?

Das Endergebnis dieser Übung wird als Autorisierungsmodell bezeichnet. Es definiert die Prinzipien, Ressourcen und Aktionen und wie sie zueinander in Beziehung stehen. Für die Erstellung dieses Modells sind keine besonderen Kenntnisse über Cedar oder den Dienst Verified Permissions erforderlich. Stattdessen ist es, wie jede andere, in erster Linie eine Übung zur Gestaltung der Benutzererfahrung und kann sich in Artefakten wie Schnittstellenmodellen, logischen Diagrammen und einer allgemeinen Beschreibung dessen, wie Berechtigungen beeinflussen, was Benutzer im Produkt sehen, äußern. Cedar ist so konzipiert, dass es flexibel genug ist, um den Kunden bei einem

Modell entgegenzukommen, anstatt das Modell zu zwingen, sich unnatürlich zu verbiegen, um der Implementierung eines Cedar zu entsprechen. Daher ist ein klares Verständnis der gewünschten Benutzererfahrung der beste Weg, um zu einem optimalen Modell zu gelangen.

Dieser Abschnitt enthält allgemeine Hinweise zur Vorgehensweise bei der Entwurfsübung, worauf Sie achten sollten, und eine Sammlung von bewährten Methoden für die erfolgreiche Verwendung verifizierter Berechtigungen.

Denken Sie daran, zusätzlich zu den hier vorgestellten Richtlinien auch die bewährten Verfahren im Referenzhandbuch für Richtlinien von Cedar zu berücksichtigen.

Themen

Es gibt kein kanonisches „richtiges“ Modell

Wenn Sie ein Autorisierungsmodell entwerfen, gibt es keine einzige, eindeutig korrekte Antwort. Verschiedene Anwendungen können effektiv unterschiedliche Autorisierungsmodelle für ähnliche Konzepte verwenden, was in Ordnung ist. Betrachten Sie beispielsweise die Darstellung des Dateisystems eines Computers. Wenn Sie eine Datei in einem Unix-ähnlichen Betriebssystem erstellen, erbt sie nicht automatisch Berechtigungen aus dem übergeordneten Ordner. Im Gegensatz dazu erben Dateien in vielen anderen Betriebssystemen und den meisten Online-Services zur Dateifreigabe Berechtigungen von ihrem übergeordneten Ordner. Beide Optionen sind je nach den Umständen gültig, für die die Anwendung optimiert wird.

Die Richtigkeit einer Autorisierungslösung ist nicht absoluter Natur, sollte aber in Bezug darauf überprüft werden, wie sie die gewünschte Erfahrung bietet und ob sie ihre Ressourcen so schützt, wie sie es erwarten. Wenn Ihr Autorisierungsmodell dies erfüllt, ist es erfolgreich.

Aus diesem Grund ist es die hilfreichste Voraussetzung für die Erstellung eines effektiven Autorisierungsmodells, Ihr Design mit der gewünschten Benutzererfahrung zu beginnen.

Konzentrieren Sie sich auf Ihre Ressourcen, die über den API-Betrieb hinausgehen

In den meisten kundenorientierten Anwendungen orientieren sich die Berechtigungen an den Ressourcen, die von der Anwendung unterstützt werden. Beispielsweise kann eine Filesharing-Anwendung Berechtigungen als Aktionen darstellen, die für eine Datei oder einen Ordner ausgeführt werden können. Dies ist ein gutes, einfaches Modell, das die zugrunde liegende Implementierung und die Backend-API-Operationen abstrahiert.

Im Gegensatz dazu entwerfen andere Arten von Anwendungen, insbesondere Webdienste, häufig Berechtigungen rund um die API-Operationen selbst. Wenn ein Webdienst beispielsweise eine API mit dem Namen bereitstelltcreateThing(), könnte das Autorisierungsmodell eine entsprechende Berechtigung definieren, oder eine action in Cedar benanntecreateThing. Dies funktioniert in vielen Situationen und macht es einfach, die Berechtigungen zu verstehen. Um den createThing Vorgang aufzurufen, benötigen Sie die createThing Aktionsberechtigung. Scheint einfach, oder?

Sie werden feststellen, dass der Prozess „Erste Schritte“ in der Verified Permissions-Konsole die Option beinhaltet, Ihre Ressourcen und Aktionen direkt über eine API zu erstellen. Dies ist eine nützliche Grundlage: eine direkte Zuordnung zwischen Ihrem Richtlinienspeicher und der API, für die er autorisiert.

Dieser API-orientierte Ansatz kann jedoch nicht optimal sein, da APIs lediglich ein Proxy für das sind, was Ihre Kunden wirklich zu schützen versuchen: die zugrunde liegenden Daten und Ressourcen. Wenn mehrere APIs den Zugriff auf dieselben Ressourcen steuern, kann es für Administratoren schwierig sein, sich Gedanken über die Pfade zu diesen Ressourcen zu machen und den Zugriff entsprechend zu verwalten.

Stellen Sie sich zum Beispiel ein Benutzerverzeichnis vor, das die Mitglieder einer Organisation enthält. Benutzer können in Gruppen organisiert werden, und eines der Sicherheitsziele besteht darin, die Entdeckung von Gruppenmitgliedschaften durch Unbefugte zu verhindern. Der Dienst, der dieses Benutzerverzeichnis verwaltet, bietet zwei API-Operationen:

- `listMembersOfGroup`
- `listGroupMembershipsForUser`

Kunden können jeden dieser Vorgänge verwenden, um die Gruppenmitgliedschaft zu ermitteln. Daher muss der Berechtigungsadministrator daran denken, den Zugriff auf beide Vorgänge zu koordinieren. Dies wird noch komplizierter, wenn Sie sich später dafür entscheiden, einen neuen API-Vorgang hinzuzufügen, um zusätzliche Anwendungsfälle zu adressieren, wie z. B. die folgenden.

- `isUserInGroups`(eine neue API, um schnell zu testen, ob ein Benutzer zu einer oder mehreren Gruppen gehört)

Aus Sicherheitsgründen eröffnet diese API einen dritten Weg zur Erkennung von Gruppenmitgliedschaften, wodurch die sorgfältig ausgearbeiteten Berechtigungen des Administrators beeinträchtigt werden.

Wir empfehlen, die API-Semantik zu ignorieren und sich stattdessen auf die zugrunde liegenden Daten und Ressourcen und deren Zuordnungsvorgänge zu konzentrieren. Die Anwendung dieses Ansatzes auf das Beispiel der Gruppenmitgliedschaft würde zu einer abstrakten Berechtigung führen, wie z. B.`viewGroupMembership`, dass jede der drei API-Operationen eine Abfrage durchführen muss.

API-Name	Berechtigungen
`listMembersOfGroup`	erfordert eine `viewGroupMembership` Genehmigung für die Gruppe
`listGroupMembershipsForUser`	erfordert `viewGroupMembership` die Erlaubnis des Benutzers
`isUserInGroups`	erfordert `viewGroupMembership` die Erlaubnis des Benutzers

Durch die Definition dieser einen Berechtigung kontrolliert der Administrator erfolgreich den Zugriff auf die Erkennung von Gruppenmitgliedschaften — jetzt und für immer. Als Kompromiss muss nun jeder API-Vorgang die möglicherweise mehreren erforderlichen Berechtigungen dokumentieren, und der Administrator muss diese Dokumentation bei der Erstellung von Berechtigungen heranziehen.

Dies kann ein gültiger Kompromiss sein, wenn es notwendig ist, um Ihre Sicherheitsanforderungen zu erfüllen.

Die kombinierte Autorisierung ist normal

Eine kombinierte Autorisierung liegt vor, wenn für eine einzelne Benutzeraktivität, z. B. das Klicken auf eine Schaltfläche in der Benutzeroberfläche Ihrer Anwendung, mehrere individuelle Autorisierungsabfragen erforderlich sind, um festzustellen, ob diese Aktivität zulässig ist. Beispielsweise sind für das Verschieben einer Datei in ein neues Verzeichnis in einem Dateisystem möglicherweise drei verschiedene Berechtigungen erforderlich: die Fähigkeit, eine Datei aus dem Quellverzeichnis zu löschen, die Möglichkeit, dem Zielverzeichnis eine Datei hinzuzufügen, und möglicherweise die Möglichkeit, die Datei selbst zu berühren (abhängig von der Anwendung).

Wenn Sie mit der Entwicklung eines Autorisierungsmodells noch nicht vertraut sind, denken Sie vielleicht, dass jede Autorisierungsentscheidung in einer einzigen Autorisierungsabfrage lösbar sein muss. Dies kann jedoch zu übermäßig komplexen Modellen und verworrenen Grundsatzerklärungen führen. In der Praxis kann die Verwendung zusammengesetzter Autorisierungen hilfreich sein, um ein einfacheres Autorisierungsmodell zu erstellen. Ein gutes Autorisierungsmodell zeichnet sich unter anderem dadurch aus, dass Ihre zusammengesetzten Operationen, z. B. das Verschieben einer Datei, durch eine intuitive Aggregation von Primitiven dargestellt werden können, wenn Sie einzelne Aktionen ausreichend zerlegt haben.

Eine weitere Situation, in der eine kombinierte Autorisierung auftritt, ist, wenn mehrere Parteien an der Erteilung einer Genehmigung beteiligt sind. Stellen Sie sich ein Organisationsverzeichnis vor, in dem Benutzer Mitglieder von Gruppen sein können. Ein einfacher Ansatz besteht darin, dem Gruppenbesitzer die Erlaubnis zu erteilen, beliebige Personen hinzuzufügen. Was ist jedoch, wenn Sie möchten, dass Ihre Benutzer zuerst dem Hinzufügen zustimmen? Dadurch wird eine Handshake-Vereinbarung eingeführt, in der sowohl der Benutzer als auch die Gruppe der Mitgliedschaft zustimmen müssen. Um dies zu erreichen, können Sie eine weitere, an den Benutzer gebundene Berechtigung einführen, die festlegt, ob der Benutzer zu einer beliebigen Gruppe oder zu einer bestimmten Gruppe hinzugefügt werden kann. Wenn ein Aufrufer anschließend versucht, Mitglieder zu einer Gruppe hinzuzufügen, muss die Anwendung beide Seiten der Berechtigungen erzwingen: Der Anrufer muss berechtigt sein, Mitglieder zu der angegebenen Gruppe hinzuzufügen, und dass der einzelne hinzugefügte Benutzer über die erforderlichen Berechtigungen verfügt. WannN-wie es Händedrucke gibt, ist es üblich zu beobachtenNKombinierte Autorisierungsanfragen, um jeden Teil der Vereinbarung durchzusetzen.

Wenn Sie vor einer Entwurfsherausforderung stehen, bei der mehrere Ressourcen involviert sind und unklar ist, wie die Berechtigungen modelliert werden sollen, kann dies ein Zeichen dafür sein, dass Sie ein kombiniertes Autorisierungsszenario haben. In diesem Fall könnte eine Lösung gefunden werden, indem der Vorgang in mehrere, individuelle Autorisierungsprüfungen aufgeteilt wird.

Überlegungen zu Mehrmandantenfähigkeit

Möglicherweise möchten Sie Anwendungen für die Verwendung durch mehrere Kunden entwickeln – Unternehmen, die Ihre Anwendung nutzen, oder Mandanten – und sie in Amazon Verified Permissions integrieren. Bevor Sie Ihr Autorisierungsmodell entwickeln, entwickeln Sie eine Multi-Tenant-Strategie. Sie können die Richtlinien Ihrer Kunden in einem gemeinsam genutzten Richtlinienspeicher verwalten oder jedem einen Richtlinienspeicher pro Mandanten zuweisen.

1. Ein gemeinsam genutzter Richtlinienspeicher

   Alle Mandanten teilen sich einen einzelnen Richtlinienspeicher. Die Anwendung sendet alle Autorisierungsanforderungen an den freigegebenen Richtlinienspeicher.

2. Richtlinienspeicher pro Mandanten

   Jeder Mandant verfügt über einen dedizierten Richtlinienspeicher. Die Anwendung fragt verschiedene Richtlinienspeicher nach einer Autorisierungsentscheidung ab, je nachdem, welcher Mandant die Anforderung stellt.

Beide Strategien erzeugen ein relativ höheres Volumen an Autorisierungsanforderungen, die sich auf Ihre AWS Rechnung auswirken könnten. Wie sollten Sie dann Ihren Ansatz entwerfen? Im Folgenden finden Sie allgemeine Bedingungen, die zu Ihrer Multi-Tenancy-Autorisierungsstrategie für Verified Permissions beitragen können.

Isolation von Mandantenrichtlinien

Die Isolierung der Richtlinien jedes Mandanten von den anderen ist wichtig, um Mandantendaten zu schützen. Wenn jeder Mandant seinen eigenen Richtlinienspeicher hat, verfügt er über seinen eigenen isolierten Richtliniensatz.

Autorisierungsablauf

Sie können einen Mandanten, der eine Autorisierungsanforderung stellt, mit einer Richtlinienspeicher-ID in der Anforderung identifizieren, mit Richtlinienspeichern pro Mandanten.

Bei einem freigegebenen Richtlinienspeicher verwenden alle Anforderungen dieselbe Richtlinienspeicher-ID.

Vorlagen und Schemaverwaltung

Ihre Richtlinienvorlagen und ein Richtlinienspeicherschema fügen in jedem Richtlinienspeicher einen zusätzlichen Design- und Wartungsaufwand hinzu.

Verwaltung globaler Richtlinien

Möglicherweise möchten Sie einige global eRichtlinien auf jeden Mandanten anwenden. Der Aufwand für die Verwaltung globaler Richtlinien variiert je nach gemeinsam genutzten und pro Mandanten verwendeten Richtlinienspeichermodellen.

Mandanten außerhalb des Onboardings

Einige Mandanten tragen Elemente zu Ihrem Schema und Ihren Richtlinien bei, die für ihren Fall spezifisch sind. Wenn ein Mandant in Ihrer Organisation nicht mehr aktiv ist und Sie seine Daten entfernen möchten, variiert der Aufwand je nach Isolationsgrad anderer Mandanten.

Service-Ressourcenkontingente

Verified Permissions verfügt über Kontingente für Ressourcen und Anforderungsrate, die sich auf Ihre Entscheidung für mehrere Mandanten auswirken könnten. Weitere Informationen zu Kontingenten finden Sie unter Kontingente für Ressourcen.

Vergleich von freigegebenen Richtlinienspeichern und Richtlinienspeichern pro Mandanten

Jede Überlegung erfordert ein eigenes Maß an Zeit und Ressourcennutzung in gemeinsam genutzten und mandantenspezifischen Richtlinienspeichermodellen.

Überlegungen	Aufwandsebene in einem freigegebenen Richtlini enspeicher	Aufwandsebene in Richtlini enspeichern pro Mandanten
Isolation von Mandanten richtlinien	Mittel. Must include tenant identifiers in policies and authorization requests.	Niedrig. Isolation is default behavior. Tenant-specific policies are inaccessible to other tenants.

Autorisierungsablauf	Niedrig. All queries target one policy store.	Mittel. Must maintain mappings between each tenant and their policy store ID.
Vorlagen und Schemaverwaltung	Niedrig. Must make one schema work for all tenants.	Hoch. Schemas and templates might be less complex individually, but changes require more coordination and complexity.
Verwaltung globaler Richtlinien	Niedrig. All policies are global and can be centrally updated.	Hoch. You must add global policies to each policy store in onboarding. Replicate global policy updates between many policy stores.
Mandanten außerhalb des Onboardings	Mittel. Must identify and delete only tenant-specific policies.	Niedrig. Delete the policy store.
Service-Ressourcenkontingente	Hoch. Tenants share resource quotas that affect policy stores like schema size, policy size per resource, and identity sources per policy store.	Niedrig. Each tenant has dedicated resource quotas.

So wählen Sie aus

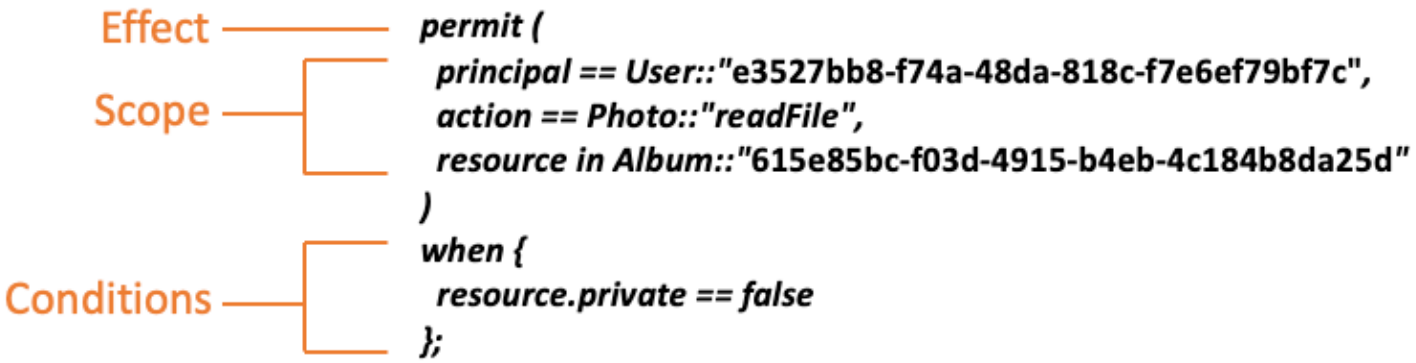
Jede Multi-Tenant-Anwendung unterscheidet sich. Vergleichen Sie sorgfältig die beiden Ansätze und ihre Überlegungen, bevor Sie eine architektonische Entscheidung treffen.

Wenn Ihre Anwendung keine mandantenspezifischen Richtlinien erfordert und eine einzelne Identitätsquelle verwendet, ist ein gemeinsam genutzter Richtlinienspeicher für alle Mandanten wahrscheinlich die effektivste Lösung. Dies führt zu einem einfacheren Autorisierungsablauf und einer globalen Richtlinienverwaltung. Das Abbinden eines Mandanten mit einem freigegebenen Richtlinienspeicher erfordert weniger Aufwand, da die Anwendung keine mandantenspezifischen Richtlinien löschen muss.

Wenn Ihre Anwendung jedoch viele mandantenspezifische Richtlinien erfordert oder mehrere Identitätsquellen verwendet, sind Richtlinienspeicher pro Mandanten wahrscheinlich am effektivsten. Sie können den Zugriff auf Mandantenrichtlinien mit IAM Richtlinien steuern, die jedem Richtlinienspeicher Berechtigungen pro Mandanten gewähren. Beim Off-Onboarding eines Mandanten müssen Sie seinen Richtlinienspeicher löschen. In einer shared-policy-store Umgebung müssen Sie Tenant-spezifische Richtlinien finden und löschen.

Wenn möglich, füllen Sie den Geltungsbereich der Richtlinie aus

Der Geltungsbereich der Police ist der Teil einer Cedar-Grundsatzerklärung nach dem`permit`oder`forbid`Schlüsselwörter und zwischen den öffnenden Klammern.



Wir empfehlen Ihnen, die Werte einzufügen`principal`und`resource`wann immer möglich. Dadurch können Verified Permissions die Richtlinien für einen effizienteren Abruf indexieren und somit die Leistung verbessern. Wenn Sie vielen verschiedenen Prinzipalen oder Ressourcen dieselben Berechtigungen gewähren müssen, empfehlen wir, eine Richtlinienvorlage zu verwenden und diese an jedes Prinzipal- und Ressourcenpaar anzuhängen.

Vermeiden Sie es, eine umfangreiche Richtlinie zu erstellen, die Listen von Prinzipalen und Ressourcen in einer`when`Klausel. Dies wird wahrscheinlich dazu führen, dass Sie auf Skalierbarkeitsgrenzen oder betriebliche Herausforderungen stoßen. Um beispielsweise einen einzelnen Benutzer zu einer großen Liste innerhalb einer Richtlinie hinzuzufügen oder daraus zu entfernen, müssen Sie die gesamte Richtlinie lesen, die Liste bearbeiten, die neue Richtlinie vollständig schreiben und Parallelitätsfehler beheben, wenn ein Administrator die Änderungen eines anderen überschreibt. Im Gegensatz dazu ist das Hinzufügen oder Entfernen eines Benutzers durch die Verwendung vieler detaillierter Berechtigungen so einfach wie das Hinzufügen oder Entfernen der einzelnen Richtlinie, die für ihn gilt.

Jede Ressource lebt in einem Container

Wenn Sie ein Autorisierungsmodell entwerfen, muss jede Aktion einer bestimmten Ressource zugeordnet werden. Mit einer Aktion wie`viewFile`, die Ressource, auf die Sie es anwenden können, ist intuitiv: eine einzelne Datei oder vielleicht eine Sammlung von Dateien in einem Ordner. Allerdings ist eine Operation wie`createFile`ist weniger intuitiv. Für welche Ressource gilt sie bei der Modellierung der Fähigkeit, eine Datei zu erstellen? Es kann nicht die Datei selbst sein, weil die Datei noch nicht existiert.

Dies ist ein Beispiel für das allgemeine Problem der Ressourcenerstellung. Die Erstellung von Ressourcen ist ein Bootstrapping-Problem. Es muss eine Möglichkeit geben, dass etwas die Erlaubnis erhält, Ressourcen zu erstellen, auch wenn noch keine Ressourcen vorhanden sind. Die Lösung besteht darin, zu erkennen, dass jede Ressource in einem Container existieren muss und dass der Container selbst als Ankerpunkt für Berechtigungen fungiert. Wenn beispielsweise ein Ordner bereits im System vorhanden ist, kann die Fähigkeit, eine Datei zu erstellen, als eine Berechtigung für diesen Ordner modelliert werden, da dies der Ort ist, an dem Berechtigungen erforderlich sind, um die neue Ressource zu instanziieren.

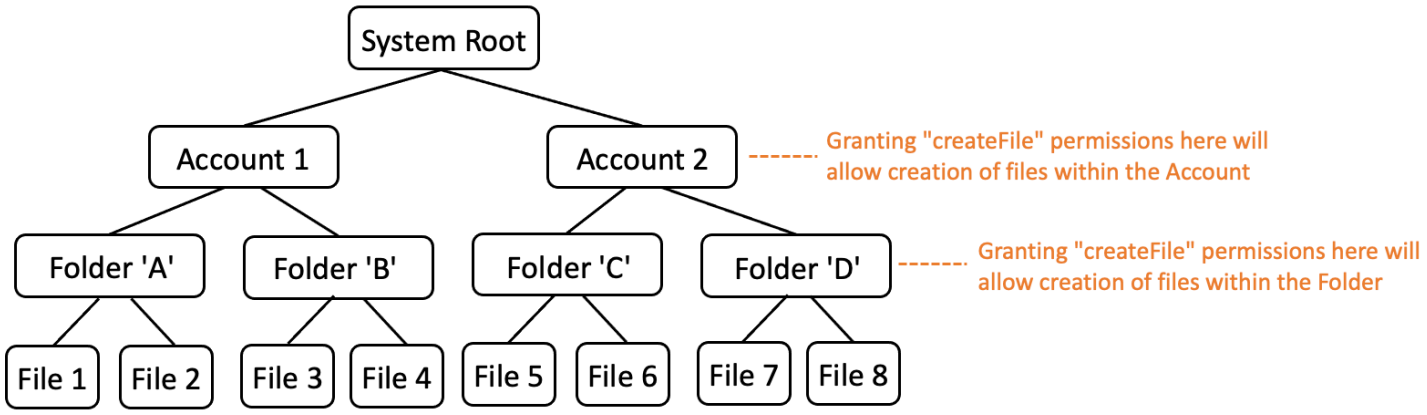
```
permit (
    principal == User::"6688f676-1aa9-456a-acf4-228340b54e9d",
    action == Action::"createFile",
    resource == Folder::"c863f89b-461f-4fc2-b638-e5fa5f79a48b"
);
```

Was aber, wenn kein Ordner existiert? Vielleicht ist dies ein brandneues Kundenkonto in einer Anwendung, für die noch keine Ressourcen vorhanden sind. In dieser Situation gibt es immer noch einen Kontext, der intuitiv verstanden werden kann, wenn man fragt: Wo kann der Kunde neue Dateien erstellen? Sie möchten nicht, dass sie Dateien in einem beliebigen Kundenkonto erstellen können. Vielmehr gibt es einen impliziten Kontext: die eigene Kontogrenze des Kunden. Daher stellt das Konto selbst den Container für die Ressourcenerstellung dar, und dies kann in einer Richtlinie, die dem folgenden Beispiel ähnelt, explizit modelliert werden.

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

Was aber, wenn es auch keine Konten gibt? Sie können sich dafür entscheiden, den Workflow für die Kundenregistrierung so zu gestalten, dass neue Konten im System erstellt werden. In diesem Fall benötigen Sie einen Container für die äußerste Grenze, innerhalb derer der Prozess die Konten erstellen kann. Dieser Container auf Stammebene stellt das System als Ganzes dar und könnte so etwas wie „Systemstamm“ heißen. Die Entscheidung, ob dies erforderlich ist und wie er benannt wird, liegt jedoch bei Ihnen, dem Eigentümer der Anwendung.

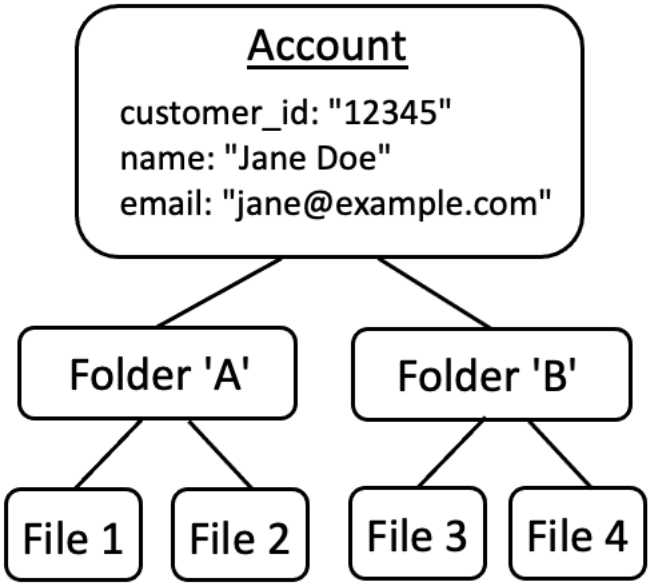
Für diese Beispielanwendung würde die resultierende Containerhierarchie daher wie folgt aussehen:



Dies ist ein Beispiel für eine Hierarchie. Andere sind ebenfalls gültig. Denken Sie daran, dass die Erstellung von Ressourcen immer im Kontext eines Ressourcencontainers erfolgt. Diese Container können implizit sein, z. B. eine Kontogrenze, und es kann leicht sein, sie zu übersehen. Achten Sie beim Entwurf Ihres Autorisierungsmodells darauf, diese impliziten Annahmen zu berücksichtigen, damit sie formell dokumentiert und im Autorisierungsmodell dargestellt werden können.

Trennen Sie die Prinzipale von den Ressourcencontainern

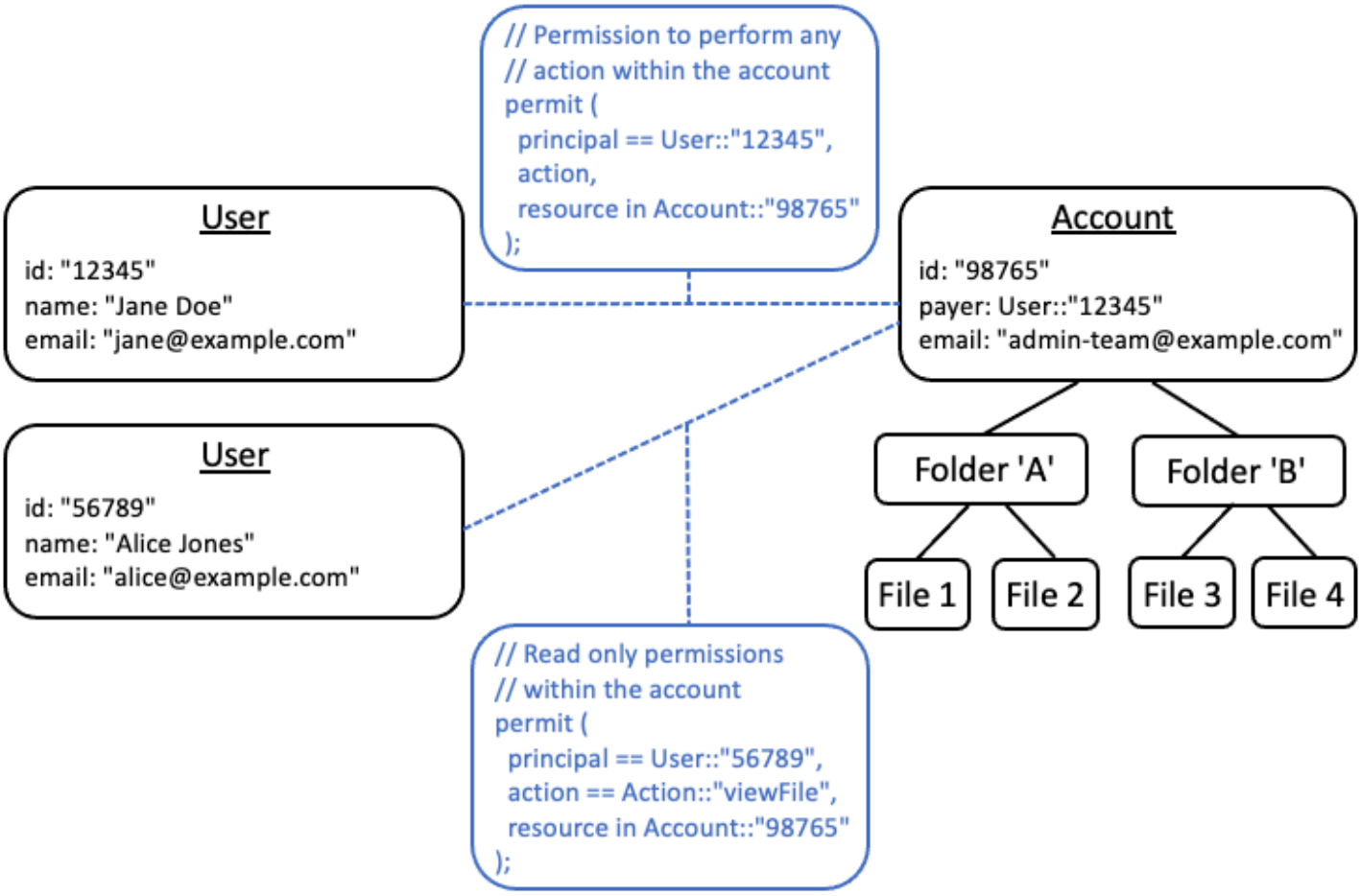
Beim Entwerfen einer Ressourcenhierarchie besteht eine der häufigsten Tendenzen, insbesondere bei kundenorientierten Anwendungen, darin, die Benutzeridentität des Kunden als Container für Ressourcen innerhalb eines Kundenkontos zu verwenden.



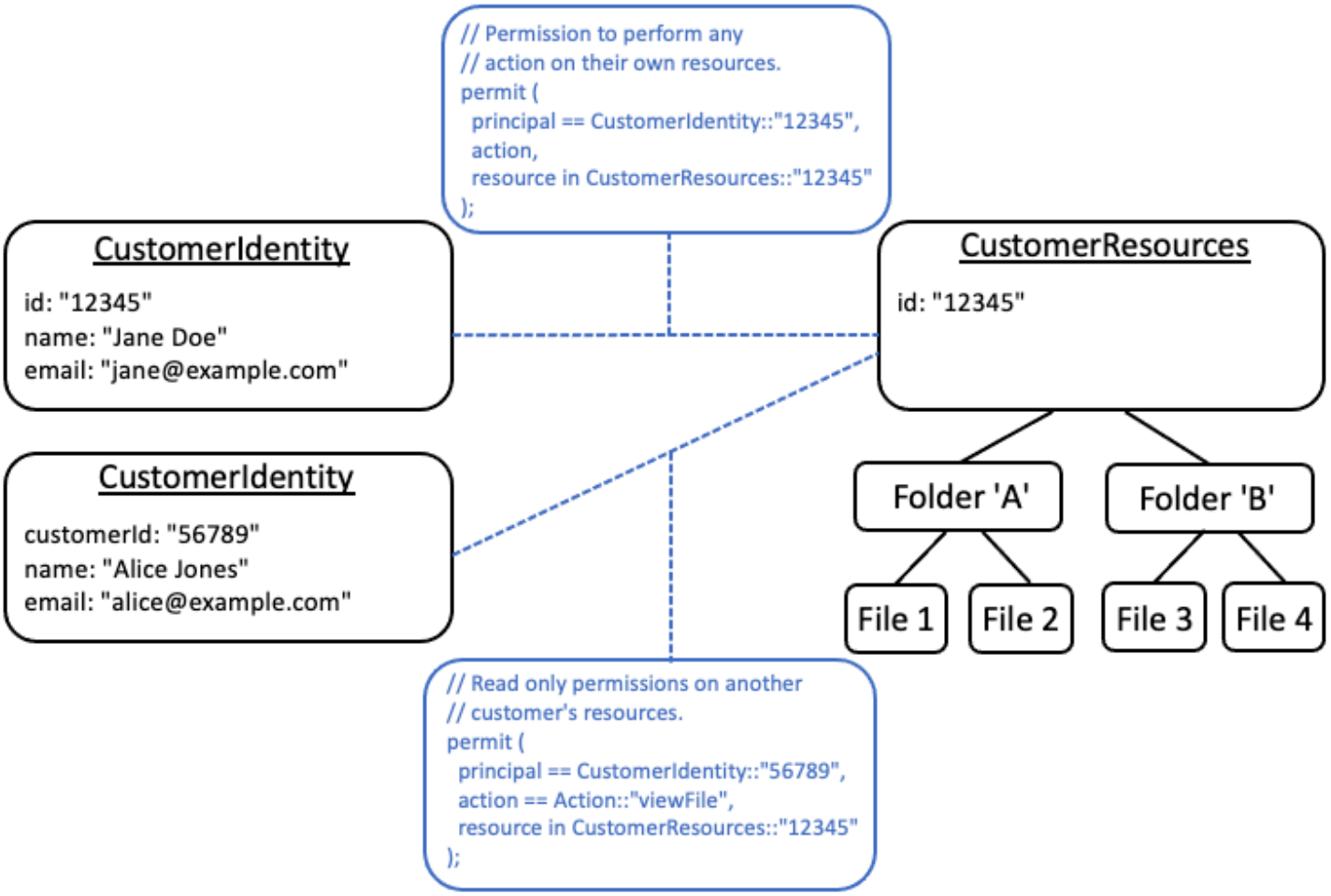
Wir empfehlen, diese Strategie als Anti-Pattern-Strategie zu behandeln. Dies liegt daran, dass umfangreichere Anwendungen von Natur aus dazu neigen, den Zugriff an zusätzliche Benutzer zu delegieren. Sie könnten sich beispielsweise dafür entscheiden, „Familienkonten“ einzuführen, bei denen andere Benutzer Kontoressourcen gemeinsam nutzen können. In ähnlicher Weise möchten Unternehmenskunden manchmal mehrere Mitarbeiter als Operatoren für Teile des Accounts benennen. Möglicherweise müssen Sie auch die Inhaberschaft eines Kontos auf einen anderen Benutzer übertragen oder die Ressourcen mehrerer Konten zusammenführen.

Wenn eine Benutzeridentität als Ressourcencontainer für ein Konto verwendet wird, wird es schwieriger, die vorherigen Szenarien zu erreichen. Noch alarmierender ist, dass, wenn anderen bei diesem Ansatz Zugriff auf den Kontocontainer gewährt wird, ihnen versehentlich Zugriff gewährt werden kann, um die Benutzeridentität selbst zu ändern, z. B. um Janes E-Mail-Adresse oder Anmeldeinformationen zu ändern.

Ein robusterer Ansatz besteht daher, wenn möglich, darin, die Principals von den Ressourcencontainern zu trennen und die Verbindung zwischen ihnen mithilfe von Konzepten wie „Administratorberechtigungen“ oder „Eigentum“ zu modellieren.



Wenn Sie über eine bestehende Anwendung verfügen, die dieses entkoppelte Modell nicht nutzen kann, empfehlen wir Ihnen, bei der Entwicklung eines Autorisierungsmodells zu erwägen, es so weit wie möglich nachzuahmen. Zum Beispiel eine Anwendung, die nur ein einziges Konzept mit dem Namen besitzt`Customer`das die Benutzeridentität, die Anmeldeinformationen und die Ressourcen, die sie besitzen, kapselt, könnte dies einem Autorisierungsmodell zuordnen, das eine logische Entität für enthält`Customer Identity`(enthält Name, E-Mail usw.) und eine separate logische Entität für`Customer Resources`oder`Customer Account`, fungiert als übergeordneter Knoten für alle Ressourcen, die sie besitzen. Beide Entitäten können sich dasselbe teilen`Id`, aber mit einem anderen`Type`.



Betten Sie keine Berechtigungen in Attribute ein

Attribute eignen sich am besten alsEingangzur Autorisierungsentscheidung. Verwenden Sie keine Attribute, um die Berechtigungen selbst darzustellen, z. B. indem Sie ein Attribut mit dem Namen „permittedFolders“ für einen Benutzer deklarieren:

```
// ANTI-PATTERN: comingling permissions into user attributes
{
    "id": "df82e4ad-949e-44cb-8acf-2d1acda71798",
    "name": "alice",
    "email": "alice@example.com",
    "permittedFolders": [
        "Folder::\"c943927f-d803-4f40-9a53-7740272cb969\"",
        "Folder::\"661817a9-d478-4096-943d-4ef1e082d19a\"",
        "Folder::\"b8ee140c-fa09-46c3-992e-099438930894\""
    ]
```

```
}
```

Und verwenden Sie anschließend das Attribut innerhalb einer Richtlinie:

```
// ANTI-PATTERN
permit (
    principal,
    action == Action::"readFile",
    resource
)
when {
    resource in principal.permittedFolders
};
```

Dieser Ansatz verwandelt ein ansonsten einfaches Autorisierungsmodell, bei dem ein bestimmter Principal Zugriff auf einen bestimmten Ordner hat, in ein ABAC-Modell (attribute-Based Access Control) mit den damit verbundenen Kompromissen. Ein solcher Kompromiss besteht darin, dass es schwieriger wird, schnell zu ermitteln, wer über Berechtigungen für eine Ressource verfügt. Um im vorherigen Beispiel zu ermitteln, wer Zugriff auf einen bestimmten Ordner hat, ist es notwendig, für jeden Benutzer eine Iteration durchzuführen, um zu überprüfen, ob dieser Ordner in seinen Attributen aufgeführt ist. Dabei ist zu beachten, dass es eine Richtlinie gibt, die Zugriff gewährt, wenn sie Zugriff haben.

Ein weiteres Risiko bei diesem Ansatz sind die Skalierungsfaktoren, wenn Berechtigungen in einem einzigen Paket zusammengefasst werden`User`eingetragene Wenn der Benutzer Zugriff auf viele Dinge hat, ist die Gesamtgröße seiner`User`Der Datensatz wird wachsen und sich möglicherweise der Höchstgrenze des Systems nähern, das die Daten speichert.

Stattdessen empfehlen wir, dieses Szenario mithilfe mehrerer einzelner Richtlinien darzustellen, möglicherweise mithilfe von Richtlinienvorlagen, um Wiederholungen zu minimieren.

```
//BETTER PATTERN
permit (
    principal == User::"df82e4ad-949e-44cb-8acf-2d1acda71798",
    action == Action::"readFile",
    resource in Folder::"c943927f-d803-4f40-9a53-7740272cb969"
);

permit (
    principal == User::"df82e4ad-949e-44cb-8acf-2d1acda71798",
    action == Action::"readFile",
```

```
    resource in Folder::"661817a9-d478-4096-943d-4ef1e082d19a"
);

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

Verified Permissions kann viele individuelle, fein abgestufte Richtlinien bei der Autorisierungsprüfung effizient handhaben. Die Modellierung von Dingen auf diese Weise ist im Laufe der Zeit einfacher zu handhaben und überprüfbar.

Bevorzugen Sie detaillierte Berechtigungen im Modell und aggregierte Berechtigungen in der Benutzeroberfläche

Eine Strategie, die Designer später oft bereuen, ist die Entwicklung eines Autorisierungsmodells mit sehr umfassenden Maßnahmen, wie z. B.`Read`und`Write`und später wird klar, dass genauere Maßnahmen erforderlich sind. Der Bedarf an feinerer Granularität kann auf Kundenfeedback für detailliertere Zugriffskontrollen oder auf Compliance- und Sicherheitsprüfer zurückzuführen sein, die Berechtigungen mit den geringsten Rechten befürworten.

Wenn keine detaillierten Berechtigungen im Voraus definiert werden, kann eine komplizierte Konvertierung erforderlich sein, um den Anwendungscode und die Richtlinienanweisungen in detailliertere Benutzerberechtigungen zu ändern. Beispielsweise muss der Anwendungscode, der zuvor für eine spezielle Aktion autorisiert wurde, geändert werden, um die detaillierten Aktionen verwenden zu können. Darüber hinaus müssen die Richtlinien aktualisiert werden, um der Migration Rechnung zu tragen:

```
permit (
    principal == User::"6688f676-1aa9-456a-acf4-228340b54e9d",
    // action  ==  Action::"read",              -- coarse-grained permission --
 commented out
    action in [                       //        -- finer grained permissions
        Action::"listFolderContents",
        Action::"viewFile"
    ],
    resource in Account::"c863f89b-461f-4fc2-b638-e5fa5f79a48b"
);
```

Um diese kostspielige Migration zu vermeiden, ist es besser, vorab detaillierte Berechtigungen zu definieren. Dies kann jedoch zu einem Kompromiss führen, wenn Ihre Endbenutzer später gezwungen sind, eine größere Anzahl detaillierter Berechtigungen zu verstehen, vor allem, wenn die meisten Kunden mit detaillierten Kontrollen zufrieden wären, wie`Read`und`Write`. Um das Beste aus beiden Welten herauszuholen, können Sie detaillierte Berechtigungen in vordefinierten Sammlungen gruppieren, z. B.`Read`und`Write`mithilfe von Mechanismen wie Richtlinienvorlagen oder Aktionsgruppen. Mithilfe dieses Ansatzes sehen Kunden nur die maßgeschneiderten Berechtigungen. Aber hinter den Kulissen haben Sie Ihre Anwendung zukunftssicher gemacht, indem Sie die kursspezifischen Berechtigungen als eine Sammlung detaillierter Aktionen modelliert haben. Wenn Kunden oder Prüfer danach fragen, können die detaillierten Berechtigungen offengelegt werden.

Erwägen Sie andere Gründe, um die Autorisierung abzufragen

Normalerweise verknüpfen wir Autorisierungsprüfungen mit Benutzeranfragen. Mit der Prüfung kann festgestellt werden, ob der Benutzer berechtigt ist, diese Anfrage auszuführen. Sie können jedoch auch Autorisierungsdaten verwenden, um das Design der Programmoberfläche zu beeinflussen. Möglicherweise möchten Sie beispielsweise einen Startbildschirm anzeigen, auf dem nur die Ressourcen aufgeführt sind, auf die der Endbenutzer zugreifen kann. Wenn Sie sich die Details einer Ressource ansehen, möchten Sie vielleicht, dass auf der Benutzeroberfläche nur die Operationen angezeigt werden, die der Benutzer mit dieser Ressource ausführen kann.

Diese Situationen können zu Kompromissen im Autorisierungsmodell führen. Beispielsweise kann eine starke Abhängigkeit von ABAC-Richtlinien (attributed-based access control) die schnelle Beantwortung der Frage „Wer hat Zugriff auf was?“ erschweren Das liegt daran, dass für die Beantwortung dieser Frage jede Regel anhand aller Prinzipien und Ressourcen geprüft werden muss, um festzustellen, ob eine Übereinstimmung vorliegt. Daher könnte sich ein Produkt, das dahingehend optimiert werden muss, dass es nur die Ressourcen auflistet, auf die der Benutzer Zugriff hat, für ein Modell der rollenbasierten Zugriffskontrolle (RBAC) entscheiden. Durch die Verwendung von RBAC kann es einfacher sein, alle einem Benutzer zugewiesenen Richtlinien zu überprüfen, um den Ressourcenzugriff zu bestimmen.

Prüfstand

Mit dem Prüfstand für verifizierte Berechtigungen können Sie Richtlinien für verifizierte Berechtigungen testen und Fehler beheben, indem Sie Autorisierungsanfragen anhand dieser Richtlinien ausführen. Der Prüfstand verwendet die von Ihnen angegebenen Parameter, um zu ermitteln, ob die Cedar-Richtlinien in Ihrem Richtlinienspeicher die Anfrage autorisieren würden. Sie können beim Testen von Autorisierungsanfragen zwischen dem visuellen Modus und dem JSON-Modus wechseln. Weitere Informationen darüber, wie die Richtlinien von Cedar strukturiert und bewertet werden, finden Sie unter Basic Policy Construction in Cedar im Cedar Policy Language Reference Guide.

> **Note**
>
> Wenn Sie mithilfe von Verified Permissions eine Autorisierungsanfrage stellen, können Sie die Liste der Principals und Ressourcen als Teil der Anfrage im Abschnitt Zusätzliche Entitäten angeben. Sie können jedoch keine Details zu den Aktionen angeben. Sie müssen im Schema angegeben oder aus der Anfrage abgeleitet werden. Sie können keine Aktion in den Abschnitt Zusätzliche Entitäten einfügen.

Einen visuellen Überblick und eine Demonstration des Prüfstands finden Sie in diesem Video.

Visual mode

> **Note**
>
> In Ihrem Richtlinienspeicher muss ein Schema definiert sein, um den visuellen Modus des Testbench verwenden zu können.

Um Richtlinien im visuellen Modus zu testen

1. Öffnen Sie die Konsole Verified Permissions unter https://console.aws.amazon.com/verifiedpermissions/. Wählen Sie Ihren Richtlinienspeicher aus.
2. Wählen Sie im Navigationsbereich auf der linken Seite die Option Testbench aus.
3. Wählen Sie den Visuellen Modus.

4. Wählen Sie im Abschnitt Principal aus den Principaltypen in Ihrem Schema den Principal aus, der die Aktion ausführt. Geben Sie einen Bezeichner für den Principal in das Textfeld ein.
5. (Optional) Wählen Sie Übergeordnetes Element hinzufügen aus, um übergeordnete Entitäten für den angegebenen Prinzipal hinzuzufügen. Um ein übergeordnetes Objekt zu entfernen, das dem Prinzipal hinzugefügt wurde, wählen Sie neben dem Namen des übergeordneten Elements die Option Entfernen aus.
6. Geben Sie den Attributwert für jedes Attribut des angegebenen Prinzipals an. Der Prüfstand verwendet die angegebenen Attributwerte in der simulierten Autorisierungsanfrage.
7. Wählen Sie im Abschnitt Ressource die Ressource aus, auf die der Principal reagiert. Geben Sie einen Bezeichner für die Ressource in das Textfeld ein.
8. (Optional) Wählen Sie Übergeordnetes Element hinzufügen aus, um übergeordnete Entitäten für die angegebene Ressource hinzuzufügen. Um ein übergeordnetes Element zu entfernen, das der Ressource hinzugefügt wurde, wählen Sie neben dem Namen der übergeordneten Ressource die Option Entfernen aus.
9. Geben Sie den Attributwert für jedes Attribut der angegebenen Ressource an. Der Prüfstand verwendet die angegebenen Attributwerte in der simulierten Autorisierungsanfrage.
10. Wählen Sie im Abschnitt Aktion aus der Liste der gültigen Aktionen für den angegebenen Prinzipal und die angegebene Ressource die Aktion aus, die der Prinzipal ausführt.
11. Geben Sie den Attributwert für jedes Attribut der angegebenen Aktion an. Der Prüfstand verwendet die angegebenen Attributwerte in der simulierten Autorisierungsanfrage.
12. (Optional) Wählen Sie im Abschnitt Zusätzliche Entitäten die Option Entität hinzufügen aus, um Entitäten hinzuzufügen, die für die Autorisierungsentscheidung bewertet werden sollen.
13. Wählen Sie den Entitätsbezeichner aus der Dropdownliste aus und geben Sie den Entitätsbezeichner ein.
14. (Optional) Wählen Sie Übergeordnetes Element hinzufügen aus, um übergeordnete Entitäten für die angegebene Entität hinzuzufügen. Um ein übergeordnetes Element zu entfernen, das der Entität hinzugefügt wurde, wählen Sie neben dem Namen der übergeordneten Entität die Option Entfernen aus.
15. Geben Sie den Attributwert für jedes Attribut der angegebenen Entität an. Der Prüfstand verwendet die angegebenen Attributwerte in der simulierten Autorisierungsanfrage.
16. Wählen Sie Bestätigen, um die Entität dem Prüfstand hinzuzufügen.
17. Wählen Sie Autorisierungsanfrage ausführen, um die Autorisierungsanfrage für die Cedar-Richtlinien in Ihrem Richtlinienspeicher zu simulieren. Auf dem Prüfstand werden die Entscheidung, ob die Anfrage zugelassen oder abgelehnt wurde, zusammen mit

Informationen zu den erfüllten Richtlinien oder den bei der Evaluierung aufgetretenen Fehlern angezeigt.

JSON mode

Um Richtlinien im JSON-Modus zu testen

1. Öffnen Sie die Konsole „Verified Permissions“ unter https://console.aws.amazon.com/verifiedpermissions/. Wählen Sie Ihren Richtlinienspeicher aus.
2. Wählen Sie im Navigationsbereich auf der linken Seite die Option Testbench aus.
3. Wählen Sie den JSON-Modus.
4. Wenn Sie im Abschnitt Anforderungsdetails ein Schema definiert haben, wählen Sie aus den Prinzipaltypen in Ihrem Schema den Principal aus, der die Aktion ausführt. Geben Sie einen Bezeichner für den Principal in das Textfeld ein.

   Wenn Sie kein Schema definiert haben, geben Sie den Principal in das Textfeld Principal ergreift Aktion ein.
5. Wenn Sie ein Schema definiert haben, wählen Sie die Ressource aus den Ressourcentypen in Ihrem Schema aus. Geben Sie einen Bezeichner für die Ressource in das Textfeld ein.

   Wenn Sie kein Schema definiert haben, geben Sie die Ressource in das Textfeld Ressource ein.
6. Wenn Sie ein Schema definiert haben, wählen Sie die Aktion aus der Liste der gültigen Aktionen für den angegebenen Prinzipal und die angegebene Ressource aus.

   Wenn Sie kein Schema definiert haben, geben Sie die Aktion in das Textfeld Aktion ein.
7. Geben Sie den Kontext der zu simulierenden Anforderung in das Feld Kontext ein. Der Anforderungskontext besteht aus zusätzlichen Informationen, die für Autorisierungsentscheidungen verwendet werden können.
8. Geben Sie im Feld Entitäten die Hierarchie der Entitäten und ihrer Attribute ein, die für die Autorisierungsentscheidung ausgewertet werden sollen.
9. Wählen Sie Autorisierungsanfrage ausführen, um die Autorisierungsanfrage für die Cedar-Richtlinien in Ihrem Richtlinienspeicher zu simulieren. Auf dem Prüfstand werden die Entscheidung, ob die Anfrage zugelassen oder abgelehnt wurde, zusammen mit Informationen zu den erfüllten Richtlinien oder den bei der Evaluierung aufgetretenen Fehlern angezeigt.

Implementierung der Autorisierung in Amazon Verified Permissions

Nachdem Sie Ihren Richtlinienspeicher, Ihre Richtlinien, Vorlagen, Ihr Schema und Ihr Autorisierungsmodell erstellt haben, können Sie mit der Autorisierung von Anfragen mithilfe von Amazon Verified Permissions beginnen. Um die Autorisierung mit verifizierten Berechtigungen zu implementieren, müssen Sie die Konfiguration von Richtlinien AWS mit der Integration in eine Anwendung kombinieren. Um Verified Permissions in Ihre Anwendung zu integrieren, fügen Sie ein AWS SDK hinzu und implementieren Sie die Methoden, die die Verified Permissions API aufrufen und Autorisierungsentscheidungen anhand Ihres Richtlinienspeichers generieren.

Die Autorisierung mit verifizierten Berechtigungen ist nützlich für UX-Berechtigungen und API-Berechtigungen in Ihren Anwendungen.

UX-Berechtigungen

: Steuern Sie den Benutzerzugriff auf Ihre Anwendungs-UX. Sie können einem Benutzer erlauben, nur genau die Formulare, Schaltflächen, Grafiken und anderen Ressourcen anzuzeigen, auf die er zugreifen muss. Wenn sich ein Benutzer beispielsweise anmeldet, möchten Sie vielleicht festlegen, ob die Schaltfläche „Geld überweisen“ in seinem Konto sichtbar ist. Sie können auch die Aktionen steuern, die ein Benutzer ausführen kann. Beispielsweise möchten Sie in derselben Banking-App möglicherweise festlegen, ob Ihr Benutzer die Kategorie einer Transaktion ändern darf.

API-Berechtigungen

: Steuern Sie den Benutzerzugriff auf Daten. Anwendungen sind häufig Teil eines verteilten Systems und beziehen Informationen von externen APIs. Im Beispiel der Banking-App, bei der Verified Permissions die Anzeige einer Schaltfläche „Geld überweisen“ zugelassen hat, muss eine komplexere Autorisierungsentscheidung getroffen werden, wenn Ihr Benutzer eine Überweisung veranlasst. Verified Permissions kann die API-Anfrage autorisieren, in der die Zielkonten aufgeführt sind, die als Übertragungsziele in Frage kommen, und dann die Anfrage, die Übertragung auf das andere Konto zu übertragen.

Die Beispiele, die diesen Inhalt veranschaulichen, stammen aus einem Beispielrichtlinienspeicher. Um dem nachzugehen, erstellen Sie den DigitalPetStore sample Policy Store in Ihrer Testumgebung.

Eine durchgängige Beispielanwendung, die UX-Berechtigungen mithilfe der Batch-Autorisierung implementiert, finden Sie im AWS Sicherheits-Blog unter Verwenden Sie von Amazon verifizierte Berechtigungen für eine detaillierte Autorisierung im großen Maßstab.

API-Operationen für die Autorisierung

Die Verified Permissions API verfügt über die folgenden Autorisierungsvorgänge.

IsAuthorized

Der `IsAuthorized` API-Vorgang ist der Einstiegspunkt für Autorisierungsanfragen mit verifizierten Berechtigungen. Sie müssen die Elemente „Principal“, „Action“, „Resource“, „Context“ und „Entities“ einreichen. Verified Permissions validiert die Entitäten in Ihrer Anfrage anhand Ihres Policy-Store-Schemas. Verified Permissions bewertet dann Ihre Anfrage anhand aller Richtlinien im angeforderten Richtlinienspeicher, die für die Entitäten in der Anfrage gelten.

IsAuthorizedWithToken

Der `IsAuthorizedWithToken` Vorgang generiert eine Autorisierungsanfrage anhand von Benutzerdaten in Amazon Cognito JSON Web Tokens (JWTs). Verified Permissions funktioniert direkt mit Amazon Cognito als Identitätsquelle in Ihrem Richtlinienspeicher. Verified Permissions füllt alle Attribute für den Principal in Ihrer Anfrage anhand der Ansprüche in der Benutzer-ID oder in den Zugriffstoken auf. Sie können Aktionen und Ressourcen anhand von Benutzerattributen oder Gruppenmitgliedschaften in einem Amazon Cognito Cognito-Benutzerpool autorisieren.

Sie können keine Informationen über Gruppen- oder Benutzerprinzipaltypen in eine `IsAuthorizedWithToken` Anfrage aufnehmen. Sie müssen alle Hauptdaten in das von Ihnen angegebene JWT eingeben.

BatchIsAutorisiert

Der `BatchIsAuthorized` Vorgang verarbeitet mehrere Autorisierungsentscheidungen für einen einzelnen Prinzipal oder eine einzelne Ressource in einer einzigen API-Anfrage. Dieser Vorgang gruppiert Anfragen in einem einzigen Batch-Vorgang, der die Quotennutzung minimiert und Autorisierungsentscheidungen für jede der bis zu 30 komplexen verschachtelten Aktionen zurückgibt. Mit der Batch-Autorisierung für eine einzelne Ressource können Sie die Aktionen filtern, die ein Benutzer für eine Ressource ausführen kann. Mit der Batch-Autorisierung für einen einzelnen Prinzipal können Sie nach den Ressourcen filtern, für die ein Benutzer Aktionen ausführen kann.

BatchIsAuthorizedWithToken

Der `BatchIsAuthorizedWithToken` Vorgang verarbeitet mehrere Autorisierungsentscheidungen für einen einzelnen Prinzipal in einer API-Anfrage. Der Prinzipal wird von der Identitätsquelle Ihres Richtlinienspeichers in einer ID oder einem Zugriffstoken bereitgestellt. Dieser Vorgang gruppiert Anfragen in einem einzigen Batch-Vorgang, der die Kontingentnutzung minimiert und Autorisierungsentscheidungen für jede von bis zu 30 Anfragen nach Aktionen und Ressourcen zurückgibt. In Ihren Richtlinien können Sie ihren Zugriff über ihre Attribute oder ihre Gruppenmitgliedschaft in einem Amazon Cognito Cognito-Benutzerpool autorisieren.

Wie bei `IsAuthorizedWithToken` können Sie keine Informationen über Gruppen- oder Benutzerprinzipaltypen in eine `BatchIsAuthorizedWithToken` Anfrage aufnehmen. Sie müssen alle Prinzipaldaten in das von Ihnen bereitgestellte JWT eingeben.

Testen Sie Ihr Autorisierungsmodell

Um zu verstehen, wie sich die Autorisierungsentscheidung für verifizierte Berechtigungen bei der Bereitstellung Ihrer Anwendung auswirkt, können Sie Ihre Richtlinien bei der Entwicklung mit Prüfstand und mit HTTPS-REST-API-Anfragen an verifizierte Berechtigungen evaluieren. Der Prüfstand ist ein Tool AWS Management Console zur Auswertung von Autorisierungsanfragen und Antworten in Ihrem Richtlinienspeicher.

Die REST-API für verifizierte Berechtigungen ist der nächste Schritt in Ihrer Entwicklung, wenn Sie von einem konzeptionellen Verständnis zum Anwendungsdesign übergehen. Die Verified Permissions API akzeptiert Autorisierungsanfragen mit IsAuthorizedIsAuthorizedWithToken, und BatchIsAutorisiert als signierte AWS API-Anfragen an regionale Service-Endpunkte. Um Ihr Autorisierungsmodell zu testen, können Sie Anfragen mit einem beliebigen API-Client generieren und überprüfen, ob Ihre Richtlinien die Autorisierungsentscheidungen erwartungsgemäß zurückgeben.

Mit dem folgenden Verfahren können Sie beispielsweise `IsAuthorized` in einem Beispiel-Richtlinienspeicher testen.

Test bench

1. Öffnen Sie die Konsole Verified Permissions unter https://console.aws.amazon.com/verifiedpermissions/. Erstellen Sie aus dem Beispielrichtlinienspeicher einen Richtlinienspeicher mit dem Namen DigitalPetStore.

2. Wählen Sie in Ihrem neuen Richtlinienspeicher die Option Testbench aus.
3. Füllen Sie Ihre Testbench-Anfrage IsAuthorizedin der API-Referenz für verifizierte Berechtigungen aus. Die folgenden Details entsprechen den Bedingungen in Beispiel 4, das auf das DigitalPetStore-Beispiel verweist.

    a. Stellen Sie Alice als Principal ein. Wählen Sie für Principal taking action aus `DigitalPetStore::User` und geben Sie ein`Alice`.

    b. Lege Alices Rolle als Kunde fest. Wählen Sie Elternteil hinzufügen`DigitalPetStore::Role`, wählen Sie und geben Sie Kunde ein.

    c. Geben Sie als Ressource die Bestellung „1234" ein. Wählen Sie unter Ressource, auf die der Principal reagiert, `DigitalPetStore::Order` und geben Sie ein`1234`.

    d. Die `DigitalPetStore::Order` Ressource benötigt ein `owner` Attribut. Lege Alice als Eigentümerin der Bestellung fest. Wähle `DigitalPetStore::User` und gib ein `Alice`

    e. Alice wollte die Bestellung einsehen. Wählen Sie für Aktion, die der Schulleiter gerade ergreift, die Option`DigitalPetStore::Action::"GetOrder"`.
4. Wählen Sie Autorisierungsanfrage ausführen aus. In einem unveränderten Richtlinienspeicher führt diese Anfrage zu einer `ALLOW` Entscheidung. Notieren Sie sich die Richtlinie „Zufrieden“, die die Entscheidung zurückgegeben hat.
5. Wählen Sie in der linken Navigationsleiste Richtlinien aus. Überprüfen Sie die statische Richtlinie mit der Beschreibung Kundenrolle — Bestellung abrufen.
6. Beachten Sie, dass Verified Permissions die Anfrage zugelassen hat, weil der Principal eine Kundenrolle innehatte und der Eigentümer der Ressource war.

REST API

1. Öffnen Sie die Konsole Verified Permissions unter https://console.aws.amazon.com/verifiedpermissions/. Erstellen Sie aus dem Beispielrichtlinienspeicher einen Richtlinienspeicher mit dem Namen DigitalPetStore.
2. Notieren Sie sich die Policy-Store-ID Ihres neuen Policy-Speichers.
3. Kopieren Sie aus IsAuthorizedder API-Referenz für verifizierte Berechtigungen den Anfragetext von Beispiel 4, der auf das DigitalPetStore-Beispiel verweist.
4. Öffnen Sie Ihren API-Client und erstellen Sie eine Anfrage an den regionalen Service-Endpunkt für Ihren Policy Store. Füllen Sie die Header wie im Beispiel gezeigt aus.

5. Fügen Sie den Text der Beispielanforderung ein und ändern Sie den Wert von in `policyStoreId` die zuvor notierte Policy Store-ID.
6. Reichen Sie die Anfrage ein und überprüfen Sie die Ergebnisse. In einem DigitalPetStandardspeicher für Store-Richtlinien gibt diese Anfrage eine `ALLOW` Entscheidung zurück.

Sie können in Ihrer Testumgebung Änderungen an Richtlinien, Schemas und Anforderungen vornehmen, um die Ergebnisse zu ändern und komplexere Entscheidungen zu treffen.

1. Ändern Sie die Anfrage so, dass die Entscheidung unter Verifizierte Berechtigungen geändert wird. Ändern Sie beispielsweise Alices Rolle auf `Employee` oder ändern Sie das `owner` Attribut der Bestellung 1234 auf`Bob`.
2. Ändern Sie Richtlinien so, dass sie sich auf Autorisierungsentscheidungen auswirken. Ändern Sie beispielsweise die Richtlinie mit der Beschreibung Kundenrolle — Bestellung abrufen, um die Bedingung zu entfernen, dass der Eigentümer der Anfrage sein `User` muss, `Resource` und ändern Sie die Anfrage so, dass der Kunde die Bestellung einsehen `Bob` möchte.
3. Ändern Sie das Schema, damit Richtlinien komplexere Entscheidungen treffen können. Aktualisieren Sie die Anforderungsentitäten, damit Alice die neuen Anforderungen erfüllen kann. Bearbeiten Sie das Schema beispielsweise so, dass `User` Sie Mitglied von `ActiveUsers` oder sein können`InactiveUsers`. Aktualisieren Sie die Richtlinie, sodass nur aktive Benutzer ihre eigenen Bestellungen einsehen können. Aktualisieren Sie die Anforderungsentitäten, sodass Alice ein aktiver oder inaktiver Benutzer ist.

Integration mit Apps und AWS SDKs

Um Amazon Verified Permissions in Ihrer Anwendung zu implementieren, müssen Sie die Richtlinien und das Schema definieren, die Ihre App durchsetzen soll. Nachdem Ihr Autorisierungsmodell eingerichtet und getestet wurde, besteht Ihr nächster Schritt darin, API-Anfragen vom Punkt der Durchsetzung aus zu generieren. Dazu müssen Sie eine Anwendungslogik einrichten, um Benutzerdaten zu sammeln und diese für Autorisierungsanfragen zu verwenden.

Wie eine App Anfragen mit verifizierten Berechtigungen autorisiert

1. Sammeln Sie Informationen über den aktuellen Benutzer. In der Regel werden die Details eines Benutzers in den Details einer authentifizierten Sitzung bereitgestellt, z. B. in einem JWT- oder Websitzungscookie. Diese Benutzerdaten können aus einer Amazon Cognito Cognito-

Identitätsquelle stammen, die mit Ihrem Policy Store verknüpft ist, oder von einem anderen OpenID Connect (OIDC) -Anbieter.

2. Sammeln Sie Informationen über die Ressource, auf die ein Benutzer zugreifen möchte. In der Regel erhält Ihre Anwendung Informationen über die Ressource, wenn ein Benutzer eine Auswahl trifft, sodass Ihre App ein neues Asset laden muss.
3. Bestimmen Sie die Aktion, die Ihr Benutzer ergreifen möchte.
4. Generieren Sie eine Autorisierungsanfrage an Verified Permissions mit dem Principal, der Aktion, der Ressource und den Entitäten für den versuchten Vorgang Ihres Benutzers. Verified Permissions bewertet die Anfrage anhand der Richtlinien in Ihrem Richtlinienspeicher und gibt eine Autorisierungsentscheidung zurück.
5. Ihre Anwendung liest die Antwort „Zulassen“ oder „Verweigern“ von Verified Permissions und erzwingt die Entscheidung in Bezug auf die Anfrage des Benutzers.

API-Operationen für verifizierte Berechtigungen sind in AWS SDKs integriert. Um verifizierte Berechtigungen in eine App aufzunehmen, integrieren Sie das AWS SDK für die von Ihnen gewählte Sprache in das App-Paket.

Weitere Informationen und das Herunterladen von AWS SDKs finden Sie unter Tools für Amazon Web Services.

Im Folgenden finden Sie Links zur Dokumentation für Ressourcen mit verifizierten Berechtigungen in verschiedenen 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

Das folgende AWS SDK for JavaScript Beispiel für `IsAuthorized` stammt aus der detaillierten Autorisierung von Simplify mit Amazon Verified Permissions und Amazon Cognito.

```
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"
                }
            }
         }
       ]
    }
});
```

Weitere Ressourcen für Entwickler

- Workshop „Von Amazon verifizierte Berechtigungen“
- Von Amazon verifizierte Berechtigungen — Ressourcen
- Implementieren Sie einen benutzerdefinierten Autorisierungsrichtlinienanbieter für ASP.NET Core-Apps mithilfe von Amazon Verified Permissions
- Erstellen Sie mit Amazon Verified Permissions einen Berechtigungsservice für Geschäftsanwendungen
- Vereinfachen Sie die detaillierte Autorisierung mit Amazon Verified Permissions und Amazon Cognito

Hinzufügen von Kontext

Kontext sind die Informationen, die für Richtlinienentscheidungen relevant sind, aber nicht Teil der Identität Ihres Prinzipals, Ihrer Aktion oder Ihrer Ressource sind. Möglicherweise möchten Sie eine Aktion nur von einer Reihe von Quell-IP-Adressen zulassen oder nur, wenn sich Ihr Benutzer mit MFA angemeldet hat. Ihre Anwendung hat Zugriff auf diese kontextbezogenen Sitzungsdaten und muss sie für Autorisierungsanforderungen ausfüllen. Die Kontextdaten in einer Autorisierungsanforderung für verifizierte Berechtigungen müssen in einem `-contextMap`Element JSON-formatiert sein.

Die Beispiele, die diesen Inhalt veranschaulichen, stammen aus einem Beispielrichtlinienspeicher . Erstellen Sie als Nächstes den DigitalPetStore Beispielrichtlinienspeicher in Ihrer Testumgebung.

Das folgende Kontextobjekt deklariert einen der einzelnen Cedar-Datentypen für eine Anwendung basierend auf dem DigitalPetStore Beispielrichtlinienspeicher.

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

```
         "record": {
            "IPAddress": {
               "string": "192.0.2.178"
            },
            "Country": {
               "string": "United States of America"
            },
            "SSL": {
               "boolean": true
            }
         }
      },
      "approvedBy": {
         "entityIdentifier": {
            "entityId": "Bob",
            "entityType": "DigitalPetStore::User"
         }
      }
   }
}
```

Datentypen im Autorisierungskontext

Boolesch

Eine Binärdatei `true` oder ein `false` Wert. Im Beispiel `MfaAuthenticated` gibt der boolesche Wert `true` für an, dass der Kunde eine Multi-Faktor-Authentifizierung durchgeführt hat, bevor er auffordert, seine Bestellung anzuzeigen.

Einstellen

Eine Sammlung von Kontextelementen. Satzmitglieder können denselben Typ haben, wie in diesem Beispiel, oder von verschiedenen Typen, einschließlich eines verschachtelten Satzes. Im Beispiel ist der Kunde mit 3 verschiedenen Konten verknüpft.

String

Eine Folge von Buchstaben, Zahlen oder Symbolen, eingeschlossen in '' Zeichen. Im Beispiel stellt die `UserAgent` Zeichenfolge den Browser dar, den der Kunde zum Anzeigen seiner Bestellung verwendet hat.

Long

Als ganze Zahl. In dem Beispiel `RequestedOrderCount` zeigt die an, dass diese Anfrage Teil eines Batches ist, der dazu geführt hat, dass der Kunde vier seiner früheren Bestellungen anfragt.

Datensatz

Eine Sammlung von Attributen. Sie müssen diese Attribute im Anforderungskontext deklarieren. Ein Richtlinienspeicher mit einem Schema muss diese Entität und die Attribute der Entität im Schema enthalten. Im Beispiel enthält der `NetworkInfo` Datensatz Informationen über die ursprüngliche IP des Benutzers, die vom Client festgelegte Geolokalisierung dieser IP und die Verschlüsselung während der Übertragung.

EntityIdentifier

Ein Verweis auf eine Entität und Attribute, die im -`entities`Element der Anforderung deklariert sind. Im Beispiel wurde die Bestellung des Benutzers vom Mitarbeiter genehmigt`Bob`.

Um diesen Beispielkontext in der Beispiel-DigitalPetStoreApp zu testen, müssen Sie Ihre Anfrage `entities`, Ihr Richtlinienspeicherschema und die statische Richtlinie mit der Beschreibung Customer Role - Get Order aktualisieren.

Ändern, DigitalPetStore um Autorisierungskontext zu akzeptieren

Anfänglich DigitalPetStore ist kein sehr komplexer Richtlinienspeicher. Es enthält keine vorkonfigurierten Richtlinien oder Kontextattribute zur Unterstützung des von uns vorgestellten Kontextes. Um eine Beispielautorisierungsanforderung mit diesen Kontextinformationen auszuwerten, nehmen Sie die folgenden Änderungen an Ihrem Richtlinienspeicher und Ihrer Autorisierungsanforderung vor.

Schema

Wenden Sie die folgenden Aktualisierungen auf Ihr Richtlinienspeicherschema an, um die neuen Kontextattribute zu unterstützen. Aktualisieren Sie `GetOrder` in `actions` wie folgt.

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

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

Um auf den `record` Datentyp namens `NetworkInfo` in Ihrem Anforderungskontext zu verweisen, erstellen Sie wie folgt ein commonType-Konstrukt in Ihrem Schema. Ein `commonType`Konstrukt ist ein gemeinsam genutzter Satz von Attributen, die Sie auf verschiedene Entitäten anwenden können.

> **Note**
>
> Der visuelle Schemaeditor für Verified Permissions unterstützt derzeit keine `commonType`Konstrukte. Wenn Sie sie Ihrem Schema hinzufügen, können Sie Ihr Schema nicht mehr im visuellen Modus anzeigen.

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

Policy

Die folgende Richtlinie richtet Bedingungen ein, die von jedem der bereitgestellten Kontextelemente erfüllt werden müssen. Es baut auf der vorhandenen statischen Richtlinie mit der Beschreibung Customer Role – Get Order auf. Diese Richtlinie erfordert zunächst nur, dass der Prinzipal, der eine Anforderung stellt, der Eigentümer der Ressource ist.

```
permit (
    principal in DigitalPetStore::Role::"Customer",
    action in [DigitalPetStore::Action::"GetOrder"],
    resource
) when {
    principal == resource.owner &&
    context.MfaAuthorized == true &&
```

```
    context.UserAgent like "*My UserAgent*" &&
    context.RequestedOrderCount <= 4 &&
    context.AccountCodes.contains(111122223333) &&
    context.NetworkInfo.Country like "*United States*" &&
    context.NetworkInfo.SSL == true &&
    context.NetworkInfo.IPAddress like "192.0.2.*" &&
    context.approvedBy in DigitalPetStore::Role::"Employee"
};
```

Wir haben nun verlangt, dass die Anforderung zum Abrufen einer Bestellung die zusätzlichen Kontextbedingungen erfüllt, die wir der Anforderung hinzugefügt haben.

1. Der Benutzer muss sich mit MFA angemeldet haben.
2. Der Webbrowser des Benutzers `User-Agent` muss die Zeichenfolge enthaltenMy `UserAgent`.
3. Der Benutzer muss aufgefordert haben, 4 oder weniger Bestellungen anzuzeigen.
4. Einer der Kontocodes des Benutzers muss sein`111122223333`.
5. Die IP-Adresse des Benutzers muss aus den USA stammen, er muss sich in einer verschlüsselten Sitzung befinden und seine IP-Adresse muss mit beginnen`192.0.2.`.
6. Ein Mitarbeiter muss seine Bestellung genehmigt haben. Im -`entities`Element der Autorisierungsanforderung deklarieren wir einen Benutzer `Bob` mit der Rolle `Employee`.

Request body

Nachdem Sie Ihren Richtlinienspeicher mit dem entsprechenden Schema und der entsprechenden Richtlinie konfiguriert haben, können Sie diese Autorisierungsanforderung der API-Operation Verified Permissions vorlegenIsAuthorized. Beachten Sie, dass das `entities` Segment eine Definition von `Bob`, einem Benutzer mit der Rolle , enthält`Employee`.

```
{
   "principal": {
      "entityType": "DigitalPetStore::User",
      "entityId": "Alice"
   },
   "action": {
      "actionType": "DigitalPetStore::Action",
      "actionId": "GetOrder"
   },
   "resource": {
```

```
      "entityType": "DigitalPetStore::Order",
      "entityId": "1234"
    },
    "context": {
      "contextMap": {
       "MfaAuthorized": {
        "boolean": true
       },
       "UserAgent": {
        "string": "My UserAgent 1.12"
       },
       "RequestedOrderCount":{
        "long": 4
       },
       "AccountCodes": {
        "set": [
         {"long": 111122223333},
         {"long": 444455556666},
         {"long": 123456789012}
         ]
       },
       "NetworkInfo": {
         "record": {
           "IPAddress": {"string": "192.0.2.178"},
           "Country": {"string": "United States of America"},
           "SSL": {"boolean": true}
         }
       },
       "approvedBy": {
         "entityIdentifier": {
          "entityId": "Bob",
          "entityType": "DigitalPetStore::User"
         }
       }
      }
    },
    "entities": {
      "entityList": [
        {
          "identifier": {
             "entityType": "DigitalPetStore::User",
             "entityId": "Alice"
          },
          "attributes": {
```

```
            "memberId": {
              "string": "801b87f2-1a5c-40b3-b580-eacad506d4e6"
              }
        },
        "parents": [
           {
           "entityType": "DigitalPetStore::Role",
           "entityId": "Customer"
           }
        ]
      },
      {
        "identifier": {
           "entityType": "DigitalPetStore::User",
           "entityId": "Bob"
        },
        "attributes": {
           "memberId": {
             "string": "49d9b81e-735d-429c-989d-93bec0bcfd8b"
             }
        },
        "parents": [
           {
           "entityType": "DigitalPetStore::Role",
           "entityId": "Employee"
           }
        ]
      },
      {
        "identifier": {
           "entityType": "DigitalPetStore::Order",
           "entityId": "1234"
        },
        "attributes": {
           "owner": {
             "entityIdentifier": {
               "entityType": "DigitalPetStore::User",
               "entityId": "Alice"
             }
          }
        },
        "parents": []
      }
    ]
```

```
    },
    "policyStoreId": "PSEXAMPLEabcdefg111111"
}
```

Sicherheit bei Amazon Verified Permissions

Die Sicherheit in der Cloud hat bei AWS höchste Priorität. Als AWS-Kunde profitieren Sie von Rechenzentren und Netzwerkarchitekturen, die eingerichtet wurden, um die Anforderungen der anspruchsvollsten Organisationen in puncto Sicherheit zu erfüllen.

Sicherheit ist eine übergreifende Verantwortlichkeit zwischen AWS und Ihnen. Das Modell der geteilten Verantwortung beschreibt dies als Sicherheit der Cloud selbst und als Sicherheit in der Cloud:

- Sicherheit der Cloud – AWS ist dafür verantwortlich, die Infrastruktur zu schützen, mit der AWS-Services in der AWS Cloud ausgeführt werden. AWS stellt Ihnen außerdem Services bereit, die Sie sicher nutzen können. Auditoren von Drittanbietern testen und überprüfen die Effektivität unserer Sicherheitsmaßnahmen im Rahmen der AWS-Compliance-Programme regelmäßig. Weitere Informationen zu den Compliance-Programmen, die für Amazon Verified Permissions gelten, finden Sie unterAWSIm Umfang des Compliance-Programms enthaltene Dienstleistungen.
- Sicherheit in der Cloud – Ihr Verantwortungsumfang wird durch den AWS-Service bestimmt, den Sie verwenden. Sie sind auch für andere Faktoren verantwortlich, etwa für die Vertraulichkeit Ihrer Daten, für die Anforderungen Ihres Unternehmens und für die geltenden Gesetze und Vorschriften.

In dieser Dokumentation erfahren Sie, wie Sie das Modell der geteilten Verantwortung anwenden, wenn Sie verifizierte Berechtigungen verwenden. In den folgenden Themen erfahren Sie, wie Sie verifizierte Berechtigungen konfigurieren, um Ihre Sicherheits- und Compliance-Ziele zu erreichen. Du lernst auch, wie man andere benutztAWSDienste, die Ihnen helfen, Ihre Ressourcen für verifizierte Berechtigungen zu überwachen und zu sichern.

Themen

- Datenschutz bei Amazon Verified Permissions
- Identitäts- und Zugriffsmanagement für Amazon Verified Permissions
- Konformitätsprüfung für von Amazon Verified Permissions
- Stabilität der von Amazon verifizierten Berechtigungen

Datenschutz bei Amazon Verified Permissions

DerAWS Modell der geteilten Verantwortunggilt für den Datenschutz in Amazon Verified Permissions. Wie in diesem Modell beschrieben, ist AWS verantwortlich für den Schutz der globalen Infrastruktur, in der die gesamte AWS Cloud ausgeführt wird. Sie sind dafür verantwortlich, die Kontrolle über Ihre in dieser Infrastruktur gehosteten Inhalte zu behalten. Dieser Inhalt enthält die Sicherheitskonfigurations- und Verwaltungsaufgaben für die von Ihnen verwendeten AWS-Services. Weitere Informationen zum Datenschutz finden Sie unter Häufig gestellte Fragen zum Datenschutz. Informationen zum Datenschutz in Europa finden Sie im Blog-Beitrag AWS-Modell der geteilten Verantwortung und in der DSGVO im AWS-Sicherheitsblog.

- Aus Datenschutzgründen empfehlen wir Ihnen,AWS-KontoAnmeldeinformationen und richte einzelne Benutzer ein mitAWS IAM Identity CenteroderAWS Identity and Access Management(IAM). So erhält jeder Benutzer nur die Berechtigungen, die zum Durchführen seiner Aufgaben erforderlich sind.
- Wir empfehlen Ihnen, Ihre Daten auf folgende Weise zu sichern:
  - Verwenden Sie für jedes Konto die Multi-Faktor Authentifizierung (MFA).
  - Verwenden Sie SSL/TLS für die Kommunikation mit AWS-Ressourcen. Wir benötigen TLS 1.2.
  - Richten Sie die API und die Protokollierung von Benutzeraktivitäten mit AWS CloudTrail ein.
  - Verwenden Sie AWS-Verschlüsselungslösungen zusammen mit allen Standardsicherheitskontrollen in AWS-Services.
  - Verwenden Sie erweiterte verwaltete Sicherheitsservices wie Amazon Macie, die dabei helfen, in Amazon S3 gespeicherte persönliche Daten zu erkennen und zu schützen.
  - Wenn Sie für den Zugriff auf AWS über eine Befehlszeilenschnittstelle oder über eine API FIPS 140-2-validierte kryptografische Module benötigen, verwenden Sie einen FIPS-Endpunkt. Weitere Informationen über verfügbare FIPS-Endpunkte finden Sie unter Federal Information Processing Standard (FIPS) 140-2.
- Wir empfehlen dringend, in Freitextfeldern, z. B. im Feld Name, keine vertraulichen oder sensiblen Informationen wie die E-Mail-Adressen Ihrer Kunden einzugeben. Dies gilt auch, wenn Sie mit verifizierten Berechtigungen oder anderen Tools arbeiten.AWS-Servicesmit der Konsole, API,AWS CLI, oderAWSSDKs. Alle Daten, die Sie in Tags oder Freitextfelder eingeben, die für Namen verwendet werden, können für Abrechnungs- oder Diagnoseprotokolle verwendet werden. Wenn Sie eine URL für einen externen Server bereitstellen, empfehlen wir dringend, keine Anmeldeinformationen zur Validierung Ihrer Anforderung an den betreffenden Server in die URL einzuschließen.

- Ihre Aktionsnamen sollten keine vertraulichen Informationen enthalten.
- Wir empfehlen außerdem dringend, immer eindeutige, nicht veränderbare und nicht wiederverwendbare Identifikatoren für Ihre Entitäten (Ressourcen und Prinzipale) zu verwenden. In einer Testumgebung können Sie sich für die Verwendung einfacher Entitätskennungen entscheiden, wie`jane`oder`bob`für den Namen einer Entität vom Typ`User`. In einem Produktionssystem ist es jedoch aus Sicherheitsgründen wichtig, dass Sie eindeutige Werte verwenden, die nicht wiederverwendet werden können. Wir empfehlen, Werte wie Universally Unique Identifiers (UUIDs) zu verwenden. Denken Sie zum Beispiel an den Benutzer`jane`wer verlässt das Unternehmen. Später lässt du jemand anderen den Namen benutzen`jane`. Dieser neue Benutzer erhält automatisch Zugriff auf alles, was durch Richtlinien gewährt wird, die immer noch verweisen`User::"jane"`. Verified Permissions und Cedar können nicht zwischen dem neuen Benutzer und dem vorherigen Benutzer unterscheiden.

  Diese Anleitung gilt sowohl für Hauptkennungen als auch für Ressourcenkennungen. Verwenden Sie immer Identifikatoren, die garantiert eindeutig sind und niemals wiederverwendet werden, um sicherzustellen, dass Sie nicht unbeabsichtigt Zugriff gewähren, weil in einer Richtlinie eine alte Kennung enthalten ist.

- Stellen Sie sicher, dass die Zeichenketten, die Sie zur Definition angeben`Long`und`Decimal`Die Werte liegen innerhalb des gültigen Bereichs jedes Typs. Stellen Sie außerdem sicher, dass Ihre Verwendung von arithmetischen Operatoren nicht zu einem Wert außerhalb des gültigen Bereichs führt. Wenn der Bereich überschritten wird, führt der Vorgang zu einer Überlauf-Ausnahme. Eine Richtlinie, die zu einem Fehler führt, wird ignoriert, was bedeutet, dass eine Genehmigungsrichtlinie den Zugriff möglicherweise unerwartet nicht zulässt oder dass eine Verboten-Richtlinie den Zugriff unerwartet nicht blockiert.

Datenverschlüsselung

Amazon Verified Permissions verschlüsselt automatisch alle Kundendaten wie Richtlinien mit einem Von AWS verwalteter Schlüssel, sodass die Verwendung eines vom Kunden verwalteten Schlüssels weder erforderlich noch unterstützt wird.

Identitäts- und Zugriffsmanagement für Amazon Verified Permissions

AWS Identity and Access Management (IAM) hilft einem Administrator AWS-Service , den Zugriff auf AWS Ressourcen sicher zu kontrollieren. IAM Administratoren kontrollieren, wer authentifiziert (angemeldet) und autorisiert werden kann (über Berechtigungen verfügt), um Ressourcen mit verifizierten Berechtigungen zu verwenden. IAM ist eine AWS-Service , die Sie ohne zusätzliche Kosten verwenden können.

Themen

Zielgruppe

Wie Sie AWS Identity and Access Management (IAM) verwenden, hängt davon ab, welche Arbeit Sie im Bereich Verifizierte Berechtigungen ausführen.

Dienstbenutzer — Wenn Sie den Dienst Verified Permissions für Ihre Arbeit verwenden, stellt Ihnen Ihr Administrator die erforderlichen Anmeldeinformationen und Berechtigungen zur Verfügung. Wenn Sie für Ihre Arbeit mehr Funktionen von Verified Permissions verwenden, benötigen Sie möglicherweise zusätzliche Berechtigungen. Wenn Sie die Fuktionsweise der Zugriffskontrolle nachvollziehen, wissen Sie bereits, welche Berechtigungen Sie von Ihrem Administrator anzufordern müssen. Wenn Sie unter Verifizierte Berechtigungen nicht auf eine Funktion zugreifen können, finden Sie weitere Informationen unterFehlerbehebung bei Identität und Zugriff auf Amazon Verified Permissions.

Dienstadministrator — Wenn Sie in Ihrem Unternehmen für die Ressourcen mit verifizierten Berechtigungen verantwortlich sind, haben Sie wahrscheinlich vollen Zugriff auf verifizierte Berechtigungen. Es ist Ihre Aufgabe, zu bestimmen, auf welche Funktionen und Ressourcen von Verified Permissions Ihre Servicebenutzer zugreifen sollen. Anschließend müssen Sie Anfragen an

Ihren IAM Administrator senden, um die Berechtigungen Ihrer Servicebenutzer zu ändern. Lesen Sie die Informationen auf dieser Seite, um die grundlegenden Konzepte von zu verstehen IAM. Weitere Informationen darüber, wie Ihr Unternehmen verifizierte Berechtigungen nutzen IAM kann, finden Sie unterSo funktioniert Amazon Verified Permissions mit IAM.

IAM Administrator — Wenn Sie ein IAM Administrator sind, möchten Sie vielleicht mehr darüber erfahren, wie Sie Richtlinien zur Verwaltung des Zugriffs auf verifizierte Berechtigungen verfassen können. Beispiele für identitätsbasierte Richtlinien für verifizierte Berechtigungen, die Sie in verwenden können IAM, finden Sie unter. Beispiele für identitätsbasierte Richtlinien für Amazon Verified Permissions

Authentifizierung mit Identitäten

Authentifizierung ist die Art und Weise, wie Sie sich AWS mit Ihren Identitätsdaten anmelden. Sie müssen als IAM-Benutzer authentifiziert (angemeldet AWS) sein oder eine Rolle übernehmen. Root-Benutzer des AWS-Kontos IAM

Sie können sich AWS als föderierte Identität anmelden, indem Sie Anmeldeinformationen verwenden, die über eine Identitätsquelle bereitgestellt wurden. AWS IAM Identity Center (IAM Identity Center) -Benutzer, die Single Sign-On-Authentifizierung Ihres Unternehmens und Ihre Google- oder Facebook-Anmeldeinformationen sind Beispiele für föderierte Identitäten. Wenn Sie sich als föderierte Identität anmelden, hat Ihr Administrator zuvor einen Identitätsverbund mithilfe von Rollen eingerichtet. IAM Wenn Sie AWS mithilfe eines Verbunds darauf zugreifen, übernehmen Sie indirekt eine Rolle.

Je nachdem, welcher Benutzertyp Sie sind, können Sie sich beim AWS Management Console oder beim AWS Zugangsportal anmelden. Weitere Informationen zur Anmeldung finden Sie AWS unter So melden Sie sich bei Ihrem an AWS-Konto im AWS-Anmeldung Benutzerhandbuch.

Wenn Sie AWS programmgesteuert zugreifen, AWS stellt es ein Software Development Kit (SDK) und eine Befehlszeilenschnittstelle (CLI) bereit, um Ihre Anfragen mithilfe Ihrer Anmeldeinformationen kryptografisch zu signieren. Wenn Sie keine AWS Tools verwenden, müssen Sie Anfragen selbst signieren. Weitere Informationen zur Verwendung der empfohlenen Methode, um Anfragen selbst zu signieren, finden Sie im IAM Benutzerhandbuch unter AWS API-Anfragen signieren.

Unabhängig von der verwendeten Authentifizierungsmethode müssen Sie möglicherweise zusätzliche Sicherheitsinformationen angeben. AWS Empfiehlt beispielsweise, die Multi-Faktor-Authentifizierung (MFA) zu verwenden, um die Sicherheit Ihres Kontos zu erhöhen. Weitere Informationen finden Sie unter Multi-Faktor-Authentifizierung im AWS IAM Identity Center

Benutzerhandbuch und Verwenden der Multi-Faktor-Authentifizierung (MFA) AWS imIAM Benutzerhandbuch.

AWS-Konto Root-Benutzer

Wenn Sie einen erstellen AWS-Konto, beginnen Sie mit einer Anmeldeidentität, die vollständigen Zugriff auf alle AWS-Services Ressourcen im Konto hat. Diese Identität wird als AWS-Konto Root-Benutzer bezeichnet. Sie können darauf zugreifen, indem Sie sich mit der E-Mail-Adresse und dem Passwort anmelden, mit denen Sie das Konto erstellt haben. Wir raten ausdrücklich davon ab, den Root-Benutzer für Alltagsaufgaben zu verwenden. Schützen Sie Ihre Root-Benutzer-Anmeldeinformationen und verwenden Sie diese, um die Aufgaben auszuführen, die nur der Root-Benutzer ausführen kann. Eine vollständige Liste der Aufgaben, für die Sie sich als Root-Benutzer anmelden müssen, finden Sie im Benutzerhandbuch unter Aufgaben, für die Root-Benutzeranmeldedaten erforderlich sind.IAM

Verbundidentität

Als bewährte Methode sollten menschliche Benutzer, einschließlich Benutzer, die Administratorzugriff benötigen, für den Zugriff AWS-Services mithilfe temporärer Anmeldeinformationen den Verbund mit einem Identitätsanbieter verwenden.

Eine föderierte Identität ist ein Benutzer aus Ihrem Unternehmensbenutzerverzeichnis, einem Web-Identitätsanbieter AWS Directory Service, dem Identity Center-Verzeichnis oder einem beliebigen Benutzer, der mithilfe AWS-Services von Anmeldeinformationen zugreift, die über eine Identitätsquelle bereitgestellt wurden. Wenn föderierte Identitäten darauf zugreifen AWS-Konten, übernehmen sie Rollen, und die Rollen stellen temporäre Anmeldeinformationen bereit.

Für die zentrale Zugriffsverwaltung empfehlen wir Ihnen, AWS IAM Identity Center zu verwenden. Sie können Benutzer und Gruppen in IAM Identity Center erstellen, oder Sie können eine Verbindung zu einer Gruppe von Benutzern und Gruppen in Ihrer eigenen Identitätsquelle herstellen und diese synchronisieren, um sie in all Ihren AWS-Konten Anwendungen zu verwenden. Informationen zu IAM Identity Center finden Sie unter Was ist IAM Identity Center? im AWS IAM Identity Center - Benutzerhandbuch.

IAM-Benutzer und -Gruppen

Ein IAM-Benutzer ist eine Identität innerhalb Ihres Unternehmens AWS-Konto , die über spezifische Berechtigungen für eine einzelne Person oder Anwendung verfügt. Wenn möglich, empfehlen wir, temporäre Anmeldeinformationen zu verwenden, anstatt IAM-Benutzer zu erstellen, die langfristige Anmeldeinformationen wie Passwörter und Zugriffsschlüssel haben. Bei speziellen

Anwendungsfällen, die langfristige Anmeldeinformationen mit IAM-Benutzern erfordern, empfehlen wir jedoch, die Zugriffsschlüssel zu rotieren. Weitere Informationen finden Sie im Benutzerhandbuch unter Regelmäßiges Wechseln der Zugriffsschlüssel für Anwendungsfälle, für die IAM langfristige Anmeldeinformationen erforderlich sind.

Eine IAM Gruppe ist eine Identität, die eine Sammlung von IAM-Benutzern spezifiziert. Sie können sich nicht als Gruppe anmelden. Mithilfe von Gruppen können Sie Berechtigungen für mehrere Benutzer gleichzeitig angeben. Gruppen vereinfachen die Verwaltung von Berechtigungen, wenn es zahlreiche Benutzer gibt. Sie könnten beispielsweise einer Gruppe mit dem Namen IAM Admins Berechtigungen zum Verwalten von IAM -Ressourcen erteilen.

Benutzer unterscheiden sich von Rollen. Ein Benutzer ist einer einzigen Person oder Anwendung eindeutig zugeordnet. Eine Rolle kann von allen Personen angenommen werden, die sie benötigen. Benutzer besitzen dauerhafte Anmeldeinformationen. Rollen stellen temporäre Anmeldeinformationen bereit. Weitere Informationen finden Sie unter Wann sollte ein IAM-Benutzer (statt einer Rolle) erstellt werden? im IAM Benutzerhandbuch.

IAM Rollen

Eine IAM Rolle ist eine Identität innerhalb von Ihnen AWS-Konto , für die bestimmte Berechtigungen gelten. Sie ist einem IAM-Benutzer vergleichbar, ist aber nicht mit einer bestimmten Person verknüpft. Sie können vorübergehend eine IAM Rolle in der übernehmen, AWS Management Console indem Sie die Rollen wechseln. Sie können eine Rolle übernehmen, indem Sie eine AWS CLI oder AWS API-Operation aufrufen oder eine benutzerdefinierte URL verwenden. Weitere Informationen zu Methoden zur Verwendung von Rollen finden Sie unter Verwenden von IAM Rollen im IAM Benutzerhandbuch.

IAM Rollen mit temporären Anmeldeinformationen sind in den folgenden Situationen nützlich:

- Verbundbenutzerzugriff – Um einer Verbundidentität Berechtigungen zuzuweisen, erstellen Sie eine Rolle und definieren Berechtigungen für die Rolle. Wird eine Verbundidentität authentifiziert, so wird die Identität der Rolle zugeordnet und erhält die von der Rolle definierten Berechtigungen. Informationen zu Rollen für den Verbund finden Sie im IAM Benutzerhandbuch unter Erstellen einer Rolle für einen externen Identitätsanbieter. Wenn Sie IAM Identity Center verwenden, konfigurieren Sie einen Berechtigungssatz. Um zu steuern, worauf Ihre Identitäten nach der Authentifizierung zugreifen können, korreliert IAM Identity Center den Berechtigungssatz mit einer Rolle in IAM. Informationen zu Berechtigungssätzen finden Sie unter Berechtigungssätze im AWS IAM Identity Center -Benutzerhandbuch.

- Temporäre IAM-Benutzerberechtigungen — Ein IAM-Benutzer oder eine IAM-Rolle kann eine IAM Rolle übernehmen, um vorübergehend verschiedene Berechtigungen für eine bestimmte Aufgabe zu übernehmen.
- Kontoübergreifender Zugriff: Sie können eine IAM -Rolle verwenden, um jemandem (einem vertrauenswürdigen Prinzipal) in einem anderen Konto den Zugriff auf Ressourcen in Ihrem Konto zu ermöglichen. Rollen stellen die primäre Möglichkeit dar, um kontoübergreifendem Zugriff zu gewähren. Bei einigen können Sie AWS-Services jedoch eine Richtlinie direkt an eine Ressource anhängen (anstatt eine Rolle als Proxy zu verwenden). Informationen zum Unterschied zwischen Rollen und ressourcenbasierten Richtlinien für den kontoübergreifenden Zugriff finden Sie im Benutzerhandbuch unter Unterschiede zwischen IAM Rollen und ressourcenbasierten Richtlinien.IAM
- Anwendungen, die auf einer EC2-Instance ausgeführt werden Amazon EC2— Sie können eine IAM Rolle verwenden, um temporäre Anmeldeinformationen für Anwendungen zu verwalten, die auf einer EC2-Instance ausgeführt werden und API-Anfragen stellen. AWS CLI AWS Das ist eher zu empfehlen, als Zugriffsschlüssel innerhalb der EC2-Instance zu speichern. Um einer EC2-Instance eine AWS Rolle zuzuweisen und sie all ihren Anwendungen zur Verfügung zu stellen, erstellen Sie ein Instance-Profil, das an die Instance angehängt ist. Ein Instance-Profil enthält die Rolle und ermöglicht, dass Programme, die in der EC2-Instance ausgeführt werden, temporäre Anmeldeinformationen erhalten. Weitere Informationen finden Sie im IAM Benutzerhandbuch unter Verwenden einer IAM Rolle zur Erteilung von Berechtigungen für Anwendungen, die auf Amazon EC2 Instances ausgeführt werden.

Informationen darüber, ob Sie IAM Rollen oder IAM-Benutzer verwenden sollten, finden Sie im Benutzerhandbuch unter Wann sollte eine IAM Rolle (anstelle eines Benutzers) erstellt werden.IAM

Verwalten des Zugriffs mit Richtlinien

Sie steuern den Zugriff, AWS indem Sie Richtlinien erstellen und diese an AWS Identitäten oder Ressourcen anhängen. Eine Richtlinie ist ein Objekt, AWS das, wenn es einer Identität oder Ressource zugeordnet ist, deren Berechtigungen definiert. AWS wertet diese Richtlinien aus, wenn ein Prinzipal (Benutzer, Root-Benutzer oder Rollensitzung) eine Anfrage stellt. Berechtigungen in den Richtlinien bestimmen, ob die Anforderung zugelassen oder abgelehnt wird. Die meisten Richtlinien werden AWS als JSON-Dokumente gespeichert. Weitere Informationen zur Struktur und zum Inhalt von JSON-Richtliniendokumenten finden Sie im IAM Benutzerhandbuch unter Überblick über JSON-Richtlinien.

Administratoren können mithilfe von AWS JSON-Richtlinien angeben, wer auf was Zugriff hat. Das bedeutet, welcher Prinzipal kann Aktionen für welche Ressourcen und unter welchen Bedingungen ausführen.

Standardmäßig haben Benutzer, Gruppen und Rollen keine Berechtigungen. Um Benutzern die Erlaubnis zu erteilen, Aktionen mit den Ressourcen durchzuführen, die sie benötigen, kann ein IAM Administrator IAM Richtlinien erstellen. Der Administrator kann dann die IAM Richtlinien zu Rollen hinzufügen, und Benutzer können die Rollen übernehmen.

IAM Richtlinien definieren Berechtigungen für eine Aktion, unabhängig von der Methode, mit der Sie den Vorgang ausführen. Angenommen, es gibt eine Richtlinie, die Berechtigungen für die `iam:GetRole`-Aktion erteilt. Ein Benutzer mit dieser Richtlinie kann Rolleninformationen von der AWS Management Console AWS CLI, der oder der AWS API abrufen.

Identitätsbasierte Richtlinien

Identitätsbasierte Richtlinien sind JSON-Berechtigungsrichtliniendokumente, die Sie einer Identität anfügen können, wie z. B. IAM-Benutzern, -Benutzergruppen oder -Rollen. Diese Richtlinien steuern, welche Aktionen die Benutzer und Rollen für welche Ressourcen und unter welchen Bedingungen ausführen können. Informationen zum Erstellen einer identitätsbasierten Richtlinie finden Sie unter IAM Richtlinien erstellen im IAM Benutzerhandbuch.

Identitätsbasierte Richtlinien können weiter als Inline-Richtlinien oder verwaltete Richtlinien kategorisiert werden. Inline-Richtlinien sind direkt in einen einzelnen Benutzer, eine einzelne Gruppe oder eine einzelne Rolle eingebettet. Verwaltete Richtlinien sind eigenständige Richtlinien, die Sie mehreren Benutzern, Gruppen und Rollen in Ihrem System zuordnen können. AWS-Konto Zu den verwalteten Richtlinien gehören AWS verwaltete Richtlinien und vom Kunden verwaltete Richtlinien. Informationen dazu, wie Sie zwischen einer verwalteten Richtlinie oder einer Inline-Richtlinie wählen können, finden Sie im IAM Benutzerhandbuch unter Auswahl zwischen verwalteten Richtlinien und Inline-Richtlinien.

Ressourcenbasierte Richtlinien

Ressourcenbasierte Richtlinien sind JSON-Richtliniendokumente, die Sie an eine Ressource anfügen. Beispiele für ressourcenbasierte Richtlinien sind IAM Rollenvertrauensrichtlinien und Amazon S3 S3-Bucket-Richtlinien. In Services, die ressourcenbasierte Richtlinien unterstützen, können Service-Administratoren sie verwenden, um den Zugriff auf eine bestimmte Ressource zu steuern. Für die Ressource, an welche die Richtlinie angehängt ist, legt die Richtlinie fest, welche Aktionen ein bestimmter Prinzipal unter welchen Bedingungen für diese Ressource ausführen kann.

Sie müssen in einer ressourcenbasierten Richtlinie einen Prinzipal angeben. Zu den Prinzipalen können Konten, Benutzer, Rollen, Verbundbenutzer oder gehören. AWS-Services

Ressourcenbasierte Richtlinien sind Richtlinien innerhalb dieses Diensts. Sie können AWS verwaltete Richtlinien nicht IAM in einer ressourcenbasierten Richtlinie verwenden.

Zugriffssteuerungslisten (ACLs)

Zugriffssteuerungslisten (ACLs) steuern, welche Prinzipale (Kontomitglieder, Benutzer oder Rollen) auf eine Ressource zugreifen können. ACLs sind ähnlich wie ressourcenbasierte Richtlinien, verwenden jedoch nicht das JSON-Richtliniendokumentformat.

Amazon S3 und Amazon VPC sind Beispiele für Services, die ACLs unterstützen. AWS WAF Weitere Informationen“ zu ACLs finden Sie unter Zugriffskontrollliste (ACL) – Übersicht (Access Control List) im Amazon-Simple-Storage-Service-Entwicklerhandbuch.

Weitere Richtlinientypen

AWS unterstützt zusätzliche, weniger verbreitete Richtlinientypen. Diese Richtlinientypen können die maximalen Berechtigungen festlegen, die Ihnen von den häufiger verwendeten Richtlinientypen erteilt werden können.

- Berechtigungsgrenzen — Eine Berechtigungsgrenze ist eine erweiterte Funktion, mit der Sie die maximalen Berechtigungen festlegen, die eine identitätsbasierte Richtlinie einer IAM Entität (IAM-Benutzer oder Rolle) gewähren kann. Sie können eine Berechtigungsgrenze für eine Entität festlegen. Die daraus resultierenden Berechtigungen sind der Schnittpunkt der identitätsbasierten Richtlinien einer Entität und ihrer Berechtigungsgrenzen. Ressourcenbasierte Richtlinien, die den Benutzer oder die Rolle im Feld `Principal` angeben, werden nicht durch Berechtigungsgrenzen eingeschränkt. Eine explizite Zugriffsverweigerung in einer dieser Richtlinien setzt eine Zugriffserlaubnis außer Kraft. Weitere Informationen zu Berechtigungsgrenzen finden Sie unter Berechtigungsgrenzen für IAM Entitäten im IAM Benutzerhandbuch.
- Service Control Policies (SCPs) — SCPs sind JSON-Richtlinien, die die maximalen Berechtigungen für eine Organisation oder Organisationseinheit (OU) in festlegen. AWS Organizations AWS Organizations ist ein Dienst zur Gruppierung und zentralen Verwaltung mehrerer Objekte AWS-Konten , die Ihrem Unternehmen gehören. Wenn Sie innerhalb einer Organisation alle Features aktivieren, können Sie Service-Kontrollrichtlinien (SCPs) auf alle oder einzelne Ihrer Konten anwenden. Das SCP schränkt die Berechtigungen für Entitäten in Mitgliedskonten ein, einschließlich der einzelnen Entitäten. Root-Benutzer des AWS-Kontos

Weitere Informationen zu Organizations und SCPs finden Sie unter Funktionsweise von SCPs im AWS Organizations -Benutzerhandbuch.

- Sitzungsrichtlinien – Sitzungsrichtlinien sind erweiterte Richtlinien, die Sie als Parameter übergeben, wenn Sie eine temporäre Sitzung für eine Rolle oder einen verbundenen Benutzer programmgesteuert erstellen. Die resultierenden Sitzungsberechtigungen sind eine Schnittmenge der auf der Identität des Benutzers oder der Rolle basierenden Richtlinien und der Sitzungsrichtlinien. Berechtigungen können auch aus einer ressourcenbasierten Richtlinie stammen. Eine explizite Zugriffsverweigerung in einer dieser Richtlinien setzt eine Zugriffserlaubnis außer Kraft. Weitere Informationen finden Sie unter Sitzungsrichtlinien im Benutzerhandbuch zu IAM .

Mehrere Richtlinientypen

Wenn mehrere auf eine Anforderung mehrere Richtlinientypen angewendet werden können, sind die entsprechenden Berechtigungen komplizierter. Informationen darüber, wie AWS festgelegt wird, ob eine Anfrage zulässig ist, wenn mehrere Richtlinientypen betroffen sind, finden Sie unter Bewertungslogik für Richtlinien im IAM Benutzerhandbuch.

So funktioniert Amazon Verified Permissions mit IAM

Bevor Sie IAM den Zugriff auf verifizierte Berechtigungen verwalten, sollten Sie sich darüber informieren, welche IAM Funktionen mit verifizierten Berechtigungen verwendet werden können.

IAM Funktionen, die Sie mit Amazon Verified Permissions verwenden können

IAM Funktion	Unterstützung für verifizierte Berechtigungen
Identitätsbasierte Richtlinien	Ja
Ressourcenbasierte Richtlinien	Nein
Richtlinienaktionen	Ja
Richtlinienressourcen	Ja
Bedingungsschlüssel für die Richtlinie	Nein
ACLs	Nein

IAM Funktion	Unterstützung für verifizierte Berechtigungen
ABAC (Tags in Richtlinien)	Nein
Temporäre Anmeldeinformationen	Ja
Hauptberechtigungen	Ja
Servicerollen	Nein
Serviceverknüpfte Rollen	Nein

Einen allgemeinen Überblick darüber, wie verifizierte Berechtigungen und andere AWS Dienste mit den meisten IAM Funktionen funktionieren, finden Sie IAM im IAM Benutzerhandbuch unter AWS Dienste, die mit funktionieren.

Identitätsbasierte Richtlinien für verifizierte Berechtigungen

Unterstützt Richtlinien auf Identitätsbasis.	Ja

Identitätsbasierte Richtlinien sind JSON-Berechtigungsrichtliniendokumente, die Sie einer Identität anfügen können, wie z. B. IAM-Benutzern, -Benutzergruppen oder -Rollen. Diese Richtlinien steuern, welche Aktionen die Benutzer und Rollen für welche Ressourcen und unter welchen Bedingungen ausführen können. Informationen zum Erstellen einer identitätsbasierten Richtlinie finden Sie unter IAM Richtlinien erstellen im Benutzerhandbuch.IAM

Mit IAM identitätsbasierten Richtlinien können Sie zulässige oder verweigerte Aktionen und Ressourcen sowie die Bedingungen angeben, unter denen Aktionen zulässig oder verweigert werden. Sie können den Prinzipal nicht in einer identitätsbasierten Richtlinie angeben, da er für den Benutzer oder die Rolle gilt, dem er zugeordnet ist. Weitere Informationen zu allen Elementen, die Sie in einer JSON-Richtlinie verwenden können, finden Sie im IAM Benutzerhandbuch unter Referenz zu IAM JSON-Richtlinienelementen.

Beispiele für identitätsbasierte Richtlinien für verifizierte Berechtigungen

Beispiele für identitätsbasierte Richtlinien für verifizierte Berechtigungen finden Sie unter. Beispiele für identitätsbasierte Richtlinien für Amazon Verified Permissions

Ressourcenbasierte Richtlinien innerhalb von Verified Permissions

Unterstützt ressourcenbasierte Richtlinien	Nein

Ressourcenbasierte Richtlinien sind JSON-Richtliniendokumente, die Sie an eine Ressource anfügen. Beispiele für ressourcenbasierte Richtlinien sind IAM Rollenvertrauensrichtlinien und Amazon S3 S3-Bucket-Richtlinien. In Services, die ressourcenbasierte Richtlinien unterstützen, können Service-Administratoren sie verwenden, um den Zugriff auf eine bestimmte Ressource zu steuern. Für die Ressource, an welche die Richtlinie angehängt ist, legt die Richtlinie fest, welche Aktionen ein bestimmter Prinzipal unter welchen Bedingungen für diese Ressource ausführen kann. Sie müssen in einer ressourcenbasierten Richtlinie einen Prinzipal angeben. Zu den Prinzipalen können Konten, Benutzer, Rollen, Verbundbenutzer oder gehören. AWS-Services

Um den kontoübergreifenden Zugriff zu ermöglichen, können Sie in einer ressourcenbasierten Richtlinie ein ganzes Konto oder IAM Entitäten in einem anderen Konto als Prinzipal angeben. Durch das Hinzufügen eines kontoübergreifenden Auftraggebers zu einer ressourcenbasierten Richtlinie ist nur die halbe Vertrauensbeziehung eingerichtet. Wenn sich der Prinzipal und die Ressource unterscheiden AWS-Konten, muss ein IAM Administrator des vertrauenswürdigen Kontos auch der Prinzipalentität (Benutzer oder Rolle) die Berechtigung zum Zugriff auf die Ressource gewähren. Sie erteilen Berechtigungen, indem Sie der juristischen Stelle eine identitätsbasierte Richtlinie anfügen. Wenn jedoch eine ressourcenbasierte Richtlinie Zugriff auf einen Prinzipal in demselben Konto gewährt, ist keine zusätzliche identitätsbasierte Richtlinie erforderlich. Weitere Informationen finden Sie IAM im IAM Benutzerhandbuch unter Kontenübergreifender Ressourcenzugriff.

Richtlinienaktionen für verifizierte Berechtigungen

Unterstützt Richtlinienaktionen	Ja

Administratoren können mithilfe von AWS JSON-Richtlinien angeben, wer auf was Zugriff hat. Das heißt, welcher Prinzipal kann Aktionen für welche Ressourcen und unter welchen Bedingungen ausführen.

Das Element `Action` einer JSON-Richtlinie beschreibt die Aktionen, mit denen Sie den Zugriff in einer Richtlinie zulassen oder verweigern können. Richtlinienaktionen haben normalerweise denselben Namen wie der zugehörige AWS API-Vorgang. Es gibt einige Ausnahmen, z. B. Aktionen, die nur mit Genehmigung durchgeführt werden können und für die es keinen passenden API-Vorgang

gibt. Es gibt auch einige Operationen, die mehrere Aktionen in einer Richtlinie erfordern. Diese zusätzlichen Aktionen werden als abhängige Aktionen bezeichnet.

Schließen Sie Aktionen in eine Richtlinie ein, um Berechtigungen zur Durchführung der zugeordneten Operation zu erteilen.

Eine Liste der Aktionen mit verifizierten Berechtigungen finden Sie unter Von Amazon Verified Permissions definierte Aktionen in der Service Authorization Reference.

Bei Richtlinienaktionen unter Verifizierte Berechtigungen wird vor der Aktion das folgende Präfix verwendet:

```
verifiedpermissions
```

Um mehrere Aktionen in einer einzigen Anweisung anzugeben, trennen Sie sie mit Kommata:

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

Sie können auch Platzhalter verwenden, um mehrere Aktionen anzugeben. Beispielsweise können Sie alle Aktionen festlegen, die mit dem Wort `Get` beginnen, einschließlich der folgenden Aktion:

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

Beispiele für identitätsbasierte Richtlinien für verifizierte Berechtigungen finden Sie unter. Beispiele für identitätsbasierte Richtlinien für Amazon Verified Permissions

Richtlinienressourcen für verifizierte Berechtigungen

Unterstützt Richtlinienressourcen	Ja

Administratoren können mithilfe von AWS JSON-Richtlinien angeben, wer auf was Zugriff hat. Das bedeutet die Festlegung, welcher Prinzipal Aktionen für welche Ressourcen unter welchen Bedingungen ausführen kann.

Das JSON-Richtlinienelement `Resource` gibt die Objekte an, auf welche die Aktion angewendet wird. Anweisungen müssen entweder ein – `Resource`oder ein `NotResource`-Element enthalten.

Als bewährte Methode geben Sie eine Ressource mit dem zugehörigen Amazon-Ressourcennamen (ARN) an. Sie können dies für Aktionen tun, die einen bestimmten Ressourcentyp unterstützen, der als Berechtigungen auf Ressourcenebene bezeichnet wird.

Verwenden Sie für Aktionen, die keine Berechtigungen auf Ressourcenebene unterstützen, z. B. Auflistungsoperationen, einen Platzhalter (*), um anzugeben, dass die Anweisung für alle Ressourcen gilt.

```
"Resource": "*"
```

Eine Liste der Ressourcentypen mit verifizierten Berechtigungen und ihren ARNs finden Sie unter Von Amazon Verified Permissions definierte Ressourcentypen in der Service Authorization Reference. Informationen darüber, mit welchen Aktionen Sie den ARN jeder Ressource angeben können, finden Sie unter Von Amazon Verified Permissions definierte Aktionen.

Bedingungsschlüssel für Richtlinien für verifizierte Berechtigungen

Unterstützt servicespezifische Richtlini enbedingungsschlüssel	Nein

Administratoren können mithilfe von AWS JSON-Richtlinien angeben, wer auf was Zugriff hat. Das heißt, welcher Prinzipal kann Aktionen für welche Ressourcen und unter welchen Bedingungen ausführen.

Das Element `Condition` (oder `Condition` block) ermöglicht Ihnen die Angabe der Bedingungen, unter denen eine Anweisung wirksam ist. Das Element `Condition` ist optional. Sie können bedingte Ausdrücke erstellen, die Bedingungsoperatoren verwenden, z. B. ist gleich oder kleiner als, damit die Bedingung in der Richtlinie mit Werten in der Anforderung übereinstimmt.

Wenn Sie mehrere `Condition`-Elemente in einer Anweisung oder mehrere Schlüssel in einem einzelnen `Condition`-Element angeben, wertet AWS diese mittels einer logischen `AND`-Operation aus. Wenn Sie mehrere Werte für einen einzelnen Bedingungsschlüssel angeben, AWS wertet die Bedingung mithilfe einer logischen `OR` Operation aus. Alle Bedingungen müssen erfüllt werden, bevor die Berechtigungen der Anweisung gewährt werden.

Sie können auch Platzhaltervariablen verwenden, wenn Sie Bedingungen angeben. Beispielsweise können Sie einem IAM-Benutzer die Berechtigung für den Zugriff auf eine Ressource nur dann

gewähren, wenn sie mit dessen IAM-Benutzernamen gekennzeichnet ist. Weitere Informationen finden Sie im IAM Benutzerhandbuch unter IAM Richtlinienelemente: Variablen und Tags.

AWS unterstützt globale Bedingungsschlüssel und dienstspezifische Bedingungsschlüssel. Eine Übersicht aller AWS globalen Bedingungsschlüssel finden Sie unter Kontext-Schlüssel für AWS globale Bedingungen im IAM Benutzerhandbuch.

ACLs in verifizierten Berechtigungen

Unterstützt ACLs	Nein

Zugriffssteuerungslisten (ACLs) steuern, welche Prinzipale (Kontomitglieder, Benutzer oder Rollen) auf eine Ressource zugreifen können. ACLs sind ähnlich wie ressourcenbasierte Richtlinien, verwenden jedoch nicht das JSON-Richtliniendokumentformat.

ABAC mit verifizierten Berechtigungen

Unterstützt ABAC (Tags in Richtlinien)	Nein

Die attributbasierte Zugriffskontrolle (ABAC) ist eine Autorisierungsstrategie, bei der Berechtigungen basierend auf Attributen definiert werden. In AWS werden diese Attribute als Tags bezeichnet. Sie können Tags an IAM Entitäten (Benutzer oder Rollen) und an viele AWS Ressourcen anhängen. Das Markieren von Entitäten und Ressourcen ist der erste Schritt von ABAC. Anschließend entwerfen Sie ABAC-Richtlinien, um Operationen zuzulassen, wenn das Tag des Prinzipals mit dem Tag der Ressource übereinstimmt, auf die sie zugreifen möchten.

ABAC ist in Umgebungen hilfreich, die schnell wachsen, und unterstützt Sie in Situationen, in denen die Richtlinienverwaltung mühsam wird.

Um den Zugriff auf der Grundlage von Tags zu steuern, geben Sie im Bedingungselement einer Richtlinie Tag-Informationen an, indem Sie die Schlüssel `aws:ResourceTag/`*`key-name`*, `aws:RequestTag/`*`key-name`*, oder Bedingung `aws:TagKeys` verwenden.

Wenn ein Service alle drei Bedingungsschlüssel für jeden Ressourcentyp unterstützt, lautet der Wert für den Service Ja. Wenn ein Service alle drei Bedingungsschlüssel für nur einige Ressourcentypen unterstützt, lautet der Wert Teilweise.

Weitere Informationen zu ABAC finden Sie unter Was ist ABAC? im IAM Benutzerhandbuch. Ein Tutorial mit Schritten zur Einrichtung von ABAC finden Sie im Benutzerhandbuch unter Verwenden der attributebasierten Zugriffskontrolle (ABAC).IAM

Verwendung temporärer Anmeldeinformationen mit verifizierten Berechtigungen

Unterstützt temporäre Anmeldeinformationen	Ja

Einige funktionieren AWS-Services nicht, wenn Sie sich mit temporären Anmeldeinformationen anmelden. Weitere Informationen, einschließlich Informationen darüber, AWS-Services wie Sie mit temporären Anmeldeinformationen arbeiten können AWS-Services , finden Sie IAM im IAM Benutzerhandbuch.

Sie verwenden temporäre Anmeldeinformationen, wenn Sie sich mit einer anderen AWS Management Console Methode als einem Benutzernamen und einem Kennwort anmelden. Wenn Sie beispielsweise AWS über den Single Sign-On-Link (SSO) Ihres Unternehmens darauf zugreifen, werden bei diesem Vorgang automatisch temporäre Anmeldeinformationen erstellt. Sie erstellen auch automatisch temporäre Anmeldeinformationen, wenn Sie sich als Benutzer bei der Konsole anmelden und dann die Rollen wechseln. Weitere Informationen zum Rollenwechsel finden Sie unter Wechseln zu einer Rolle (Konsole) im IAM Benutzerhandbuch.

Mithilfe der AWS API AWS CLI oder können Sie temporäre Anmeldeinformationen manuell erstellen. Sie können diese temporären Anmeldeinformationen dann für den Zugriff verwenden AWS. AWS empfiehlt, temporäre Anmeldeinformationen dynamisch zu generieren, anstatt langfristige Zugriffsschlüssel zu verwenden. Weitere Informationen finden Sie unter Temporäre Sicherheitsanmeldeinformationen in IAM.

Serviceübergreifende Prinzipalberechtigungen für verifizierte Berechtigungen

Unterstützt Prinzipal-Berechtigungen	Ja

Wenn Sie einen IAM-Benutzer oder eine IAM-Rolle verwenden, um Aktionen auszuführen AWS, gelten Sie als Principal. Bei einigen Services könnte es Aktionen geben, die dann eine andere Aktion in einem anderen Service initiieren. FAS verwendet die Berechtigungen des Prinzipals, der einen aufruft AWS-Service, kombiniert mit der Anforderung, Anfragen an nachgelagerte Dienste AWS-Service zu stellen. FAS-Anfragen werden nur gestellt, wenn ein Dienst eine Anfrage erhält, für deren

Abschluss Interaktionen mit anderen AWS-Services oder Ressourcen erforderlich sind. In diesem Fall müssen Sie über Berechtigungen zum Ausführen beider Aktionen verfügen. Einzelheiten zu den Richtlinien für FAS-Anfragen finden Sie unter Zugriffssitzungen weiterleiten.

Servicerollen für verifizierte Berechtigungen

Unterstützt Servicerollen	Nein

Eine Servicerolle ist eine IAM Rolle, die ein Dienst übernimmt, um Aktionen in Ihrem Namen auszuführen. Ein IAM Administrator kann eine Servicerolle von innen heraus erstellen, ändern und löschen IAM. Weitere Informationen finden Sie unter Erstellen einer Rolle zum Delegieren von Berechtigungen an einen AWS-Service im IAM -Benutzerhandbuch.

Mit Diensten verknüpfte Rollen für verifizierte Berechtigungen

Unterstützt serviceverknüpfte Rollen	Nein

Eine dienstbezogene Rolle ist eine Art von Servicerolle, die mit einer verknüpft ist. AWS-Service Der Service kann die Rolle übernehmen, um eine Aktion in Ihrem Namen auszuführen. Dienstbezogene Rollen werden in Ihrem Dienst angezeigt AWS-Konto und gehören dem Dienst. Ein IAM Administrator kann die Berechtigungen für dienstbezogene Rollen anzeigen, aber nicht bearbeiten.

Einzelheiten zum Erstellen oder Verwalten von dienstbezogenen Rollen finden Sie unter AWS Dienste, die mit funktionieren. IAM Suchen Sie in der Tabelle nach einem Service mit einem `Yes` in der Spalte Service-linked role (Serviceverknüpfte Rolle). Wählen Sie den Link Yes (Ja) aus, um die Dokumentation für die serviceverknüpfte Rolle für diesen Service anzuzeigen.

Beispiele für identitätsbasierte Richtlinien für Amazon Verified Permissions

Standardmäßig sind Benutzer und Rollen nicht berechtigt, Ressourcen mit verifizierten Berechtigungen zu erstellen oder zu ändern. Sie können auch keine Aufgaben mithilfe der AWS Management Console, AWS Command Line Interface (AWS CLI) oder AWS API ausführen. Ein IAM Administrator muss IAM Richtlinien erstellen, die Benutzern und Rollen die Berechtigung gewähren, Aktionen mit den Ressourcen durchzuführen, die sie benötigen. Der Administrator muss diese Richtlinien anschließend den Benutzern anfügen, die sie benötigen.

Informationen zum Erstellen einer IAM identitätsbasierten Richtlinie mithilfe dieser Beispieldokumente zu JSON-Richtlinien finden Sie unter IAM Richtlinien erstellen im IAM Benutzerhandbuch.

Einzelheiten zu den von Verified Permissions definierten Aktionen und Ressourcentypen, einschließlich des Formats der ARNs für jeden Ressourcentyp, finden Sie unter Aktionen, Ressourcen und Bedingungsschlüssel für Amazon Verified Permissions in der Service Authorization Reference.

Themen

- Bewährte Methoden für Richtlinien
- Verwenden der Konsole „Verified Permissions“
- Gewähren der Berechtigung zur Anzeige der eigenen Berechtigungen für Benutzer

Bewährte Methoden für Richtlinien

Identitätsbasierte Richtlinien legen fest, ob jemand Ressourcen mit verifizierten Berechtigungen in Ihrem Konto erstellen, darauf zugreifen oder sie löschen kann. Dies kann zusätzliche Kosten für Ihr verursachen AWS-Konto. Befolgen Sie beim Erstellen oder Bearbeiten identitätsbasierter Richtlinien die folgenden Anleitungen und Empfehlungen:

- Beginnen Sie mit AWS verwalteten Richtlinien und wechseln Sie zu Berechtigungen mit den geringsten Rechten — Verwenden Sie die AWS verwalteten Richtlinien, die Berechtigungen für viele gängige Anwendungsfälle gewähren, um Ihren Benutzern und Workloads zunächst Berechtigungen zu gewähren. Sie sind in Ihrem verfügbar. AWS-Konto Wir empfehlen Ihnen, die Berechtigungen weiter zu reduzieren, indem Sie vom AWS Kunden verwaltete Richtlinien definieren, die speziell auf Ihre Anwendungsfälle zugeschnitten sind. Weitere Informationen finden Sie AWS im IAM Benutzerhandbuch unter AWS Verwaltete Richtlinien oder Verwaltete Richtlinien für Jobfunktionen.
- Berechtigungen mit den geringsten Rechten anwenden — Wenn Sie Berechtigungen mit IAM Richtlinien festlegen, gewähren Sie nur die Berechtigungen, die für die Ausführung einer Aufgabe erforderlich sind. Sie tun dies, indem Sie die Aktionen definieren, die für bestimmte Ressourcen unter bestimmten Bedingungen durchgeführt werden können, auch bekannt als die geringsten Berechtigungen. Weitere Informationen zur Verwendung IAM zum Anwenden von Berechtigungen finden Sie IAM im Benutzerhandbuch unter Richtlinien und Berechtigungen.IAM
- Verwenden Sie Bedingungen in IAM Richtlinien, um den Zugriff weiter einzuschränken — Sie können Ihren Richtlinien eine Bedingung hinzufügen, um den Zugriff auf Aktionen und

Ressourcen einzuschränken. Sie können beispielsweise eine Richtlinienbedingung schreiben, um festzulegen, dass alle Anforderungen mithilfe von SSL gesendet werden müssen. Sie können auch Bedingungen verwenden, um Zugriff auf Serviceaktionen zu gewähren, wenn diese im Rahmen einer bestimmten Aktion verwendet werden AWS-Service, wie AWS CloudFormation z. Weitere Informationen finden Sie unter IAM -JSON-Richtlinienelemente: Bedingung im IAM - Benutzerhandbuch.

- Verwenden Sie IAM Access Analyzer, um Ihre IAM Richtlinien zu validieren, um sichere und funktionale Berechtigungen zu gewährleisten. IAM Access Analyzer validiert neue und bestehende Richtlinien, sodass die Richtlinien der IAM Richtliniensprache (JSON) und den IAM-Best Practices entsprechen. IAM Access Analyzer stellt mehr als 100 Richtlinienprüfungen und umsetzbare Empfehlungen zur Verfügung, damit Sie sichere und funktionale Richtlinien erstellen können. Weitere Informationen finden Sie unter IAM Access Analyzer-Richtlinienvalidierung im Benutzerhandbuch.IAM
- Multi-Faktor-Authentifizierung (MFA) erforderlich — Wenn Sie ein Szenario haben, das IAM-Benutzer oder einen Root-Benutzer in Ihrem System erfordert AWS-Konto, aktivieren Sie MFA für zusätzliche Sicherheit. Um MFA beim Aufrufen von API-Vorgängen anzufordern, fügen Sie Ihren Richtlinien MFA-Bedingungen hinzu. Weitere Informationen finden Sie unter Konfiguration des MFA-geschützten API-Zugriffs im IAM Benutzerhandbuch.

Weitere Informationen zu bewährten Methoden finden Sie IAM unter Bewährte Sicherheitsmethoden IAM im IAM Benutzerhandbuch.

Verwenden der Konsole „Verified Permissions“

Um auf die Amazon Verified Permissions-Konsole zugreifen zu können, benötigen Sie ein Mindestmaß an Berechtigungen. Diese Berechtigungen müssen es Ihnen ermöglichen, die Ressourcen mit verifizierten Berechtigungen in Ihrem aufzulisten und Details zu diesen Ressourcen anzuzeigen AWS-Konto. Wenn Sie eine identitätsbasierte Richtlinie erstellen, die strenger ist als die mindestens erforderlichen Berechtigungen, funktioniert die Konsole nicht wie vorgesehen für Entitäten (Benutzer oder Rollen) mit dieser Richtlinie.

Sie müssen Benutzern, die nur die API AWS CLI oder die AWS API aufrufen, keine Mindestberechtigungen für die Konsole gewähren. Stattdessen sollten Sie nur Zugriff auf die Aktionen zulassen, die der API-Operation entsprechen, die die Benutzer ausführen möchten.

Um sicherzustellen, dass Benutzer und Rollen die Konsole für verifizierte Berechtigungen weiterhin verwenden können, fügen Sie den Entitäten auch die verifizierten Berechtigungen `ConsoleAccess`

oder die *ReadOnly* AWS verwaltete Richtlinie hinzu. Weitere Informationen finden Sie im Benutzerhandbuch unter Hinzufügen von Berechtigungen für einen IAM Benutzer.

Gewähren der Berechtigung zur Anzeige der eigenen Berechtigungen für Benutzer

In diesem Beispiel wird gezeigt, wie Sie eine Richtlinie erstellen, die IAM-Benutzern die Berechtigung zum Anzeigen der eingebundenen Richtlinien und verwalteten Richtlinien gewährt, die ihrer Benutzeridentität angefügt sind. Diese Richtlinie umfasst Berechtigungen zum Ausführen dieser Aktion auf der Konsole oder programmgesteuert mithilfe der API AWS CLI oder AWS .

```
{
    "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": "*"
        }
    ]
}
```

Fehlerbehebung bei Identität und Zugriff auf Amazon Verified Permissions

Verwenden Sie die folgenden Informationen, um häufig auftretende Probleme zu diagnostizieren und zu beheben, die bei der Arbeit mit Verified Permissions und auftreten können IAM.

Themen

Ich bin nicht berechtigt, eine Aktion unter Verifizierte Berechtigungen auszuführen

Wenn Sie eine Fehlermeldung erhalten, dass Sie nicht zur Durchführung einer Aktion berechtigt sind, müssen Ihre Richtlinien aktualisiert werden, damit Sie die Aktion durchführen können.

Der folgende Beispielfehler tritt auf, wenn der IAM-Benutzer `mateojackson` versucht, über die Konsole Details zu einer fiktiven *`my-example-widget`*-Ressource anzuzeigen, jedoch nicht über `verifiedpermissions:`*`GetWidget`*-Berechtigungen verfügt.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 verifiedpermissions:GetWidget on resource: my-example-widget
```

In diesem Fall muss die Richtlinie für den Benutzer `mateojackson` aktualisiert werden, damit er mit der `verifiedpermissions:`*`GetWidget`*-Aktion auf die *`my-example-widget`*-Ressource zugreifen kann.

Wenn Sie Hilfe benötigen, wenden Sie sich an Ihren AWS Administrator. Ihr Administrator hat Ihnen Ihre Anmeldeinformationen zur Verfügung gestellt.

Ich bin nicht berechtigt, iam auszuführen: PassRole

Wenn Sie die Fehlermeldung erhalten, dass Sie nicht berechtigt sind, die `iam:PassRole` Aktion auszuführen, müssen Ihre Richtlinien aktualisiert werden, damit Sie eine Rolle an Verified Permissions übergeben können.

Einige AWS-Services ermöglichen es Ihnen, eine bestehende Rolle an diesen Dienst zu übergeben, anstatt eine neue Servicerolle oder eine dienstverknüpfte Rolle zu erstellen. Hierzu benötigen Sie Berechtigungen für die Übergabe der Rolle an den Dienst.

Der folgende Beispielfehler tritt auf, wenn ein IAM-Benutzer mit dem Namen Verified Permissions `marymajor` versucht, über die Konsole eine Aktion auszuführen. Die Aktion erfordert jedoch, dass der Service über Berechtigungen verfügt, die durch eine Servicerolle gewährt werden. Mary besitzt keine Berechtigungen für die Übergabe der Rolle an den Dienst.

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

In diesem Fall müssen die Richtlinien von Mary aktualisiert werden, um die Aktion `iam:PassRole` ausführen zu können.

Wenn Sie Hilfe benötigen, wenden Sie sich an Ihren AWS Administrator. Ihr Administrator hat Ihnen Ihre Anmeldeinformationen zur Verfügung gestellt.

Ich möchte Personen außerhalb von mir den Zugriff AWS-Konto auf meine Ressourcen mit verifizierten Berechtigungen ermöglichen

Sie können eine Rolle erstellen, die Benutzer in anderen Konten oder Personen außerhalb Ihrer Organisation für den Zugriff auf Ihre Ressourcen verwenden können. Sie können festlegen, wem die Übernahme der Rolle anvertraut wird. Im Fall von Diensten, die ressourcenbasierte Richtlinien oder Zugriffskontrolllisten (Access Control Lists, ACLs) verwenden, können Sie diese Richtlinien verwenden, um Personen Zugriff auf Ihre Ressourcen zu gewähren.

Weitere Informationen dazu finden Sie hier:

- Informationen darüber, ob Verified Permissions diese Funktionen unterstützt, finden Sie unterSo funktioniert Amazon Verified Permissions mit IAM.
- Informationen dazu, wie Sie Zugriff auf Ihre Ressourcen gewähren können, AWS-Konten die Ihnen gehören, finden Sie im Benutzerhandbuch unter Bereitstellen von Zugriff für einen IAM-Benutzer in einem anderen AWS-Konto , den IAM Sie besitzen.
- Informationen dazu, wie Sie Dritten Zugriff auf Ihre Ressourcen gewähren können AWS-Konten, finden Sie AWS-Konten im IAM Benutzerhandbuch unter Gewähren des Zugriffs für Dritte.
- Informationen dazu, wie Sie Zugriff über einen Identitätsverbund gewähren, finden Sie im Benutzerhandbuch unter Zugriff für extern authentifizierte Benutzer (Identitätsverbund).IAM

- Informationen zum Unterschied zwischen der Verwendung von Rollen und ressourcenbasierten Richtlinien für den kontenübergreifenden Zugriff finden Sie IAM im Benutzerhandbuch unter Kontoübergreifender Ressourcenzugriff.IAM

Konformitätsprüfung für von Amazon Verified Permissions

Informationen darüber, ob AWS-Service ein AWS-Services in den Geltungsbereich bestimmter Compliance-Programme fällt, finden Sie unter Umfang nach Compliance-Programm AWS-Services unter . Wählen Sie dort das Compliance-Programm aus, an dem Sie interessiert sind. Allgemeine Informationen finden Sie unter AWS Compliance-Programme AWS .

Sie können Prüfberichte von Drittanbietern unter herunterladen AWS Artifact. Weitere Informationen finden Sie unter Berichte herunterladen unter .

Ihre Verantwortung für die Einhaltung der Vorschriften bei der Nutzung AWS-Services hängt von der Vertraulichkeit Ihrer Daten, den Compliance-Zielen Ihres Unternehmens und den geltenden Gesetzen und Vorschriften ab. AWS stellt die folgenden Ressourcen zur Verfügung, die Sie bei der Einhaltung der Vorschriften unterstützen:

- Schnellstartanleitungen zu Sicherheit und Compliance — In diesen Bereitstellungsleitfäden werden architektonische Überlegungen erörtert und Schritte für die Implementierung von Basisumgebungen beschrieben AWS , bei denen Sicherheit und Compliance im Mittelpunkt stehen.
- Architecting for HIPAA Security and Compliance on Amazon Web Services — In diesem Whitepaper wird beschrieben, wie Unternehmen HIPAA-fähige Anwendungen entwickeln AWS können.

> **Note**
>
> Nicht AWS-Services alle sind HIPAA-fähig. Weitere Informationen finden Sie in der Referenz für HIPAA-berechtigte Services.

- AWS Compliance-Ressourcen — Diese Sammlung von Arbeitsmappen und Leitfäden gilt möglicherweise für Ihre Branche und Ihren Standort.
- AWS Leitfäden zur Einhaltung von Vorschriften für Kunden — Verstehen Sie das Modell der gemeinsamen Verantwortung aus dem Blickwinkel der Einhaltung von Vorschriften. In den Leitfäden werden die bewährten Verfahren zur Sicherung zusammengefasst AWS-Services und die Leitlinien den Sicherheitskontrollen in verschiedenen Frameworks (einschließlich des National

Institute of Standards and Technology (NIST), des Payment Card Industry Security Standards Council (PCI) und der International Organization for Standardization (ISO)) zugeordnet.

- Evaluierung von Ressourcen anhand von Regeln im AWS Config Entwicklerhandbuch — Der AWS Config Service bewertet, wie gut Ihre Ressourcenkonfigurationen den internen Praktiken, Branchenrichtlinien und Vorschriften entsprechen.
- AWS Security Hub— Auf diese AWS-Service Weise erhalten Sie einen umfassenden Überblick über Ihren internen Sicherheitsstatus. AWS Security Hub verwendet Sicherheitskontrollen, um Ihre AWS -Ressourcen zu bewerten und Ihre Einhaltung von Sicherheitsstandards und bewährten Methoden zu überprüfen. Eine Liste der unterstützten Services und Kontrollen finden Sie in der Security-Hub-Steuerungsreferenz.
- Amazon GuardDuty — Dies AWS-Service erkennt potenzielle Bedrohungen für Ihre Workloads AWS-Konten, Container und Daten, indem es Ihre Umgebung auf verdächtige und böswillige Aktivitäten überwacht. GuardDuty kann Ihnen helfen, verschiedene Compliance-Anforderungen wie PCI DSS zu erfüllen, indem es die in bestimmten Compliance-Frameworks vorgeschriebenen Anforderungen zur Erkennung von Eindringlingen erfüllt.
- AWS Audit Manager— Auf diese AWS-Service Weise können Sie Ihre AWS Nutzung kontinuierlich überprüfen, um das Risikomanagement und die Einhaltung von Vorschriften und Industriestandards zu vereinfachen.

Stabilität der von Amazon verifizierten Berechtigungen

Die globale AWS-Infrastruktur ist um AWS-Regionen und Availability Zones herum aufgebaut. AWS-Regionen bieten mehrere physisch getrennte und isolierte Availability Zones, die mit einem Netzwerk mit geringer Latenz, hohem Durchsatz und hoher Redundanz verbunden sind. Mithilfe von Availability Zones können Sie Anwendungen und Datenbanken erstellen und ausführen, die automatisch Failover zwischen Zonen ausführen, ohne dass es zu Unterbrechungen kommt. Availability Zones sind besser verfügbar, fehlertoleranter und skalierbarer als herkömmliche Infrastrukturen mit einem oder mehreren Rechenzentren.

Wenn Sie einen Richtlinienspeicher für verifizierte Berechtigungen erstellen, wird er innerhalb einer Person erstelltAWS-Regionund wird automatisch in allen Rechenzentren repliziert, aus denen die Availability Zones dieser Region bestehen. Derzeit unterstützt Verified Permissions keine regionsübergreifende Replikation.

Weitere Informationen über AWS-Regionen und Availability Zones finden Sie unter Globale AWS-Infrastruktur.

Überwachung der von Amazon verifizierten Berechtigungen

Überwachung ist wichtig, Verfügbarkeit und Leistung zu AWS bewahren. AWSstellt die folgenden Überwachungstools bereit, um die verifizierten Berechtigungen zu überwachen, zu melden, wenn etwas nicht stimmt, und um gegebenenfalls automatische Aktionen durchzuführen:

- AWS CloudTrail erfasst API-Aufrufe und zugehörige Ereignisse, die von oder im Namen Ihres AWS-Kontos erfolgten, und übermittelt die Protokolldateien an einen von Ihnen angegebenen Amazon-S3-Bucket. Sie können die Benutzer und Konten, die AWS aufgerufen haben, identifizieren, sowie die Quell-IP-Adresse, von der diese Aufrufe stammen, und den Zeitpunkt der Aufrufe ermitteln. Weitere Informationen finden Sie im AWS CloudTrail-Benutzerhandbuch.

Protokollieren von API-Aufrufen von Amazon Verified Permissions mit AWS CloudTrail

Amazon Verified Permissions ist in integriert, einem ServiceAWS CloudTrail, der die Aktionen eines Benutzers, einer Rolle oder eines -AWSServices in Verified Permissions. CloudTrail captures aller API-Aufrufe für Verified Permissions als Ereignisse aufzeichnet. Zu den erfassten Aufrufen gehören Aufrufe von der Verified Permissions-Konsole und Codeaufrufe der Verified Permissions API-Operationen. Wenn Sie einen Trail erstellen, können Sie die kontinuierliche Bereitstellung von CloudTrail Ereignissen an einen Amazon S3-Bucket aktivieren, einschließlich Ereignissen für Verified Permissions. Wenn Sie keinen Trail konfigurieren, können Sie trotzdem die neuesten Ereignisse in der CloudTrail Konsole unter Ereignisverlauf anzeigen. Anhand der von CloudTrailgesammelten Informationen können Sie die an Verified Permissions gestellte Anfrage, die IP-Adresse, von der die Anfrage gestellt wurde, den Initiator der Anfrage, den Zeitpunkt der Anfrage und zusätzliche Details bestimmen.

Weitere Informationen zu CloudTrailfinden Sie im AWS CloudTrail -Benutzerhandbuch.

Informationen zu verifizierten Berechtigungen in CloudTrail

CloudTrail wird beim Erstellen des Kontos AWS-Konto auf Ihrem aktiviert. Wenn eine Aktivität in Verified Permissions auftritt, wird diese Aktivität in einem - CloudTrail Ereignis zusammen mit anderen -AWSServiceereignissen im Ereignisverlauf aufgezeichnet. Sie können in Ihrem AWS-Konto

die neusten Ereignisse anzeigen, suchen und herunterladen. Weitere Informationen finden Sie unter Anzeigen von Ereignissen mit dem CloudTrail Ereignisverlauf .

Erstellen Sie für eine fortlaufende Aufzeichnung der Ereignisse in Ihrem AWS-Konto, einschließlich Ereignissen für Verified Permissions, einen Trail. Ein Trail ermöglicht CloudTrail die Bereitstellung von Protokolldateien an einen Amazon S3-Bucket. Wenn Sie einen Trail in der Konsole anlegen, gilt dieser für alle AWS-Regionen-Regionen. Der Trail protokolliert Ereignisse aus allen Regionen in der AWS-Partition und stellt die Protokolldateien in dem von Ihnen angegebenen Amazon-S3-Bucket bereit. Darüber hinaus können Sie andere -AWSServices konfigurieren, um die in den CloudTrail Protokollen erfassten Ereignisdaten weiter zu analysieren und entsprechend zu agieren. Weitere Informationen finden Sie hier:

- Übersicht zum Erstellen eines Trails
- CloudTrail Von unterstützte Services und Integrationen
- Konfigurieren von Amazon SNS-Benachrichtigungen für CloudTrail
- Empfangen von CloudTrail Protokolldateien aus mehreren Regionen und Empfangen von CloudTrail Protokolldateien aus mehreren Konten

Alle Aktionen mit verifizierten Berechtigungen werden von protokolliert CloudTrail und sind im API-Referenzhandbuch für Amazon Verified Permissions dokumentiert. Aufrufe der `ListPolicyStores` Aktionen `CreateIdentitySource`, `DeletePolicy`und erzeugen beispielsweise Einträge in den CloudTrail Protokolldateien.

Jeder Ereignis- oder Protokolleintrag enthält Informationen zu dem Benutzer, der die Anforderung generiert hat. Die Identitätsinformationen unterstützen Sie bei der Ermittlung der folgenden Punkte:

- Ob die Anforderung mit Stamm- oder AWS Identity and Access Management (IAM)-Benutzeranmeldeinformationen gesendet wurde
- Gibt an, ob die Anforderung mit temporären Sicherheitsanmeldeinformationen für eine Rolle oder einen Verbundbenutzer gesendet wurde.
- Ob die Anforderung aus einem anderen AWS-Service gesendet wurde

Weitere Informationen finden Sie unter CloudTrail -Element userIdentity.

Datenereignisse wie IsAuthorized und IsAuthorizedWithToken werden nicht standardmäßig protokolliert, wenn Sie einen Trail oder Ereignisdatenspeicher erstellen. Um CloudTrail Datenereignisse aufzuzeichnen, müssen Sie die unterstützten Ressourcen oder Ressourcentypen,

für die Sie Aktivitäten erfassen möchten, explizit hinzufügen. Weitere Informationen finden Sie unter Datenereignisse im Benutzerhandbuch für AWS CloudTrail.

Grundlegendes zu Protokolldateieinträgen für verifizierte Berechtigungen

Ein Trail ist eine Konfiguration, die die Bereitstellung von Ereignissen als Protokolldateien an einen von Ihnen angegebenen Amazon S3-Bucket ermöglicht. CloudTrail Protokolldateien enthalten einen oder mehrere Protokolleinträge. Ein Ereignis stellt eine einzelne Anforderung aus einer beliebigen Quelle dar und enthält Informationen über die angeforderte Aktion, das Datum und die Uhrzeit der Aktion, Anforderungsparameter usw. CloudTrail Protokolldateien sind kein geordnetes Stacktrace der öffentlichen API-Aufrufe und erscheinen daher nicht in einer bestimmten Reihenfolge.

Themen

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

> **Note**
>
> Aus Datenschutzgründen wurden einige Felder aus den Beispielen geschwärzt.

IsAuthorized


```
{
    "eventVersion": "1.08",
    "userIdentity": {
  "type": "AssumedRole",
  "principalId": "EXAMPLE_PRINCIPAL_ID",
  "arn": "arn:aws:iam::123456789012:role/ExampleRole",
  "accountId": "123456789012",
  "accessKeyId": "AKIAIOSFODNN7EXAMPLE"
    },
    "eventTime": "2023-11-20T22:55:03Z",
    "eventSource": "verifiedpermissions.amazonaws.com",
    "eventName": "IsAuthorized",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "203.0.113.0",
    "userAgent": "aws-cli/2.11.18 Python/3.11.3 Linux/5.4.241-160.348.amzn2int.x86_64
 exe/x86_64.amzn.2 prompt/off command/verifiedpermissions.is-authorized",
    "requestParameters": {
        "principal": {
            "entityType": "PhotoFlash::User",
            "entityId": "alice"
        },
        "action": {
            "actionType": "PhotoFlash::Action",
            "actionId": "ViewPhoto"
        },
        "resource": {
            "entityType": "PhotoFlash::Photo",
            "entityId": "VacationPhoto94.jpg"
        },
        "policyStoreId": "PSEXAMPLEabcdefg111111"
    },
    "responseElements": null,
    "additionalEventData": {
        "decision": "ALLOW"
    },
    "requestID": "346c4b6a-d12f-46b6-bc06-6c857bd3b28e",
    "eventID": "8a4fed32-9605-45dd-a09a-5ebbf0715bbc",
    "readOnly": true,
    "resources": [
        {
    "accountId": "123456789012",
```

```
    "type": "AWS::VerifiedPermissions::PolicyStore",
    "ARN": "arn:aws:verifiedpermissions::123456789012:policy-store/
PSEXAMPLEabcdefg111111"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": false,
    "recipientAccountId": "123456789012",
    "eventCategory": "Data"
}
```

BatchIsAuthorized

```
{
    "eventVersion": "1.08",
    "userIdentity": {
  "type": "AssumedRole",
  "principalId": "EXAMPLE_PRINCIPAL_ID",
  "arn": "arn:aws:iam::123456789012:role/ExampleRole",
  "accountId": "123456789012",
  "accessKeyId": "AKIAIOSFODNN7EXAMPLE"
    },
    "eventTime": "2023-11-20T23:02:33Z",
    "eventSource": "verifiedpermissions.amazonaws.com",
    "eventName": "BatchIsAuthorized",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "203.0.113.0",
    "userAgent": "aws-cli/2.11.18 Python/3.11.3 Linux/5.4.241-160.348.amzn2int.x86_64
 exe/x86_64.amzn.2 prompt/off command/verifiedpermissions.is-authorized",
    "requestParameters": {
        "requests": [
            {
                "principal": {
                    "entityType": "PhotoFlash::User",
                    "entityId": "alice"
                },
                "action": {
                    "actionType": "PhotoFlash::Action",
                    "actionId": "ViewPhoto"
                },
                "resource": {
                    "entityType": "PhotoFlash::Photo",
                    "entityId": "VacationPhoto94.jpg"
```

```
                }
            },
            {
                "principal": {
                    "entityType": "PhotoFlash::User",
                    "entityId": "annalisa"
                },
                "action": {
                    "actionType": "PhotoFlash::Action",
                    "actionId": "DeletePhoto"
                },
                "resource": {
                    "entityType": "PhotoFlash::Photo",
                    "entityId": "VacationPhoto94.jpg"
                }
            }
        ],
        "policyStoreId": "PSEXAMPLEabcdefg111111"
    },
    "responseElements": null,
    "additionalEventData": {
        "results": [
            {
                "request": {
                    "principal": {
                        "entityType": "PhotoFlash::User",
                        "entityId": "alice"
                    },
                    "action": {
                        "actionType": "PhotoFlash::Action",
                        "actionId": "ViewPhoto"
                    },
                    "resource": {
                        "entityType": "PhotoFlash::Photo",
                        "entityId": "VacationPhoto94.jpg"
                    }
                },
                "decision": "ALLOW"
            },
            {
                "request": {
                    "principal": {
                        "entityType": "PhotoFlash::User",
                        "entityId": "annalisa"
```

```
                    },
                    "action": {
                        "actionType": "PhotoFlash::Action",
                        "actionId": "DeletePhoto"
                    },
                    "resource": {
                        "entityType": "PhotoFlash::Photo",
                        "entityId": "VacationPhoto94.jpg"
                    }
                },
                "decision": "DENY"
            }
        ]
    },
    "requestID": "a8a5caf3-78bd-4139-924c-7101a8339c3b",
    "eventID": "7d81232f-f3d1-4102-b9c9-15157c70487b",
    "readOnly": true,
    "resources": [
        {
    "accountId": "123456789012",
    "type": "AWS::VerifiedPermissions::PolicyStore",
    "ARN": "arn:aws:verifiedpermissions::123456789012:policy-store/
PSEXAMPLEabcdefg111111"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": false,
    "recipientAccountId": "123456789012",
    "eventCategory": "Data"
}
```

CreatePolicyStore

```
{
  "eventVersion": "1.08",
  "userIdentity": {
    "type": "AssumedRole",
    "principalId": "EXAMPLE_PRINCIPAL_ID",
    "arn": "arn:aws:iam::123456789012:role/ExampleRole",
    "accountId": "123456789012",
    "accessKeyId": "AKIAIOSFODNN7EXAMPLE"
  },
  "eventTime": "2023-05-22T07:43:33Z",
```

```
  "eventSource": "verifiedpermissions.amazonaws.com",
  "eventName": "CreatePolicyStore",
  "awsRegion": "us-west-2",
  "sourceIPAddress": "203.0.113.0",
  "userAgent": "aws-sdk-rust/0.55.2 os/linux lang/rust/1.69.0",
  "requestParameters": {
    "clientToken": "a1b2c3d4-e5f6-a1b2-c3d4-TOKEN1111111",
    "validationSettings": {
      "mode": "OFF"
    }
  },
  "responseElements": {
    "policyStoreId": "PSEXAMPLEabcdefg111111",
    "arn": "arn:aws:verifiedpermissions::123456789012:policy-store/
PSEXAMPLEabcdefg111111",
    "createdDate": "2023-05-22T07:43:33.962794Z",
    "lastUpdatedDate": "2023-05-22T07:43:33.962794Z"
  },
  "requestID": "1dd9360e-e2dc-4554-ab65-b46d2cf45c29",
  "eventID": "b6edaeee-3584-4b4e-a48e-311de46d7532",
  "readOnly": false,
  "eventType": "AwsApiCall",
  "managementEvent": true,
  "recipientAccountId": "123456789012",
  "eventCategory": "Management"
}
```

ListPolicyStores

```
{
  "eventVersion": "1.08",
  "userIdentity": {
    "type": "AssumedRole",
    "principalId": "EXAMPLE_PRINCIPAL_ID",
    "arn": "arn:aws:iam::123456789012:role/ExampleRole",
    "accountId": "123456789012",
    "accessKeyId": "AKIAIOSFODNN7EXAMPLE"
  },
  "eventTime": "2023-05-22T07:43:33Z",
  "eventSource": "verifiedpermissions.amazonaws.com",
  "eventName": "ListPolicyStores",
  "awsRegion": "us-west-2",
  "sourceIPAddress": "203.0.113.0",
```

```
  "userAgent": "aws-sdk-rust/0.55.2 os/linux lang/rust/1.69.0",
  "requestParameters": {
    "maxResults": 10
  },
  "responseElements": null,
  "requestID": "5ef238db-9f87-4f37-ab7b-6cf0ba5df891",
  "eventID": "b0430fb0-12c3-4cca-8d05-84c37f99c51f",
  "readOnly": true,
  "eventType": "AwsApiCall",
  "managementEvent": true,
  "recipientAccountId": "123456789012",
  "eventCategory": "Management"
}
```

DeletePolicyStore

```
{
  "eventVersion": "1.08",
  "userIdentity": {
    "type": "AssumedRole",
    "principalId": "EXAMPLE_PRINCIPAL_ID",
    "arn": "arn:aws:iam::123456789012:role/ExampleRole",
    "accountId": "123456789012",
    "accessKeyId": "AKIAIOSFODNN7EXAMPLE"
  },
  "eventTime": "2023-05-22T07:43:32Z",
  "eventSource": "verifiedpermissions.amazonaws.com",
  "eventName": "DeletePolicyStore",
  "awsRegion": "us-west-2",
  "sourceIPAddress": "203.0.113.0",
  "userAgent": "aws-sdk-rust/0.55.2 os/linux lang/rust/1.69.0",
  "requestParameters": {
    "policyStoreId": "PSEXAMPLEabcdefg111111"
  },
  "responseElements": null,
  "requestID": "1368e8f9-130d-45a5-b96d-99097ca3077f",
  "eventID": "ac482022-b2f6-4069-879a-dd509123d8d7",
  "readOnly": false,
  "resources": [
    {
      "accountId": "123456789012",
      "type": "AWS::VerifiedPermissions::PolicyStore",
```

```
      "arn": "arn:aws:verifiedpermissions::123456789012:policy-store/
PSEXAMPLEabcdefg111111"
    }
  ],
  "eventType": "AwsApiCall",
  "managementEvent": true,
  "recipientAccountId": "123456789012",
  "eventCategory": "Management"
}
```

PutSchema

```
{
  "eventVersion": "1.08",
  "userIdentity": {
    "type": "AssumedRole",
    "principalId": "EXAMPLE_PRINCIPAL_ID",
    "arn": "arn:aws:iam::123456789012:role/ExampleRole",
    "accountId": "123456789012",
    "accessKeyId": "AKIAIOSFODNN7EXAMPLE"
  },
  "eventTime": "2023-05-16T12:58:57Z",
  "eventSource": "verifiedpermissions.amazonaws.com",
  "eventName": "PutSchema",
  "awsRegion": "us-west-2",
  "sourceIPAddress": "203.0.113.0",
  "userAgent": "aws-sdk-rust/0.55.2 os/linux lang/rust/1.69.0",
  "requestParameters": {
    "policyStoreId": "PSEXAMPLEabcdefg111111"
  },
  "responseElements": {
    "lastUpdatedDate": "2023-05-16T12:58:57.513442Z",
    "namespaces": "[some_namespace]",
    "createdDate": "2023-05-16T12:58:57.513442Z",
    "policyStoreId": "PSEXAMPLEabcdefg111111",
  },
  "requestID": "631fbfa1-a959-4988-b9f8-f1a43ff5df0d",
  "eventID": "7cd0c677-733f-4602-bc03-248bae581fe5",
  "readOnly": false,
  "resources": [
    {
      "accountId": "123456789012",
      "type": "AWS::VerifiedPermissions::PolicyStore",
```

```
      "ARN": "arn:aws:verifiedpermissions::123456789012:policy-store/
PSEXAMPLEabcdefg111111"
    }
  ],
  "eventType": "AwsApiCall",
  "managementEvent": true,
  "recipientAccountId": "123456789012",
  "eventCategory": "Management"
}
```

GetSchema

```
{
  "eventVersion": "1.08",
  "userIdentity": {
    "type": "AssumedRole",
    "principalId": "EXAMPLE_PRINCIPAL_ID",
    "arn": "arn:aws:iam::222222222222:role/ExampleRole",
    "accountId": "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

```
{
  "eventVersion": "1.08",
  "userIdentity": {
    "type": "AssumedRole",
    "principalId": "EXAMPLE_PRINCIPAL_ID",
    "arn": "arn:aws:iam::123456789012:role/ExampleRole",
    "accountId": "123456789012",
    "accessKeyId": "AKIAIOSFODNN7EXAMPLE"
  },
  "eventTime": "2023-05-16T13:00:24Z",
  "eventSource": "verifiedpermissions.amazonaws.com",
  "eventName": "CreatePolicyTemplate",
  "awsRegion": "us-west-2",
  "sourceIPAddress": "203.0.113.0",
  "userAgent": "aws-sdk-rust/0.55.2 os/linux lang/rust/1.69.0",
  "requestParameters": {
    "policyStoreId": "PSEXAMPLEabcdefg111111"
  },
  "responseElements": {
    "lastUpdatedDate": "2023-05-16T13:00:23.444404Z",
    "createdDate": "2023-05-16T13:00:23.444404Z",
    "policyTemplateId": "PTEXAMPLEabcdefg111111",
    "policyStoreId": "PSEXAMPLEabcdefg111111",
  },
  "requestID": "73953bda-af5e-4854-afe2-7660b492a6d0",
  "eventID": "7425de77-ed84-4f91-a4b9-b669181cc57b",
  "readOnly": false,
  "resources": [
    {
      "accountId": "123456789012",
      "type": "AWS::VerifiedPermissions::PolicyStore",
      "arn": "arn:aws:verifiedpermissions::123456789012:policy-store/
PSEXAMPLEabcdefg111111"
    }
  ],
  "eventType": "AwsApiCall",
```

```
  "managementEvent": true,
  "recipientAccountId": "123456789012",
  "eventCategory": "Management"
}
```

DeletePolicyTemplate

```
{
  "eventVersion": "1.08",
  "userIdentity": {
    "type": "AssumedRole",
    "principalId": "EXAMPLE_PRINCIPAL_ID",
    "arn": "arn:aws:iam::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


```
{
  "eventVersion": "1.08",
  "userIdentity": {
    "type": "AssumedRole",
    "principalId": "EXAMPLE_PRINCIPAL_ID",
    "arn": "arn:aws:iam::123456789012:role/ExampleRole",
    "accountId": "123456789012",
    "accessKeyId": "AKIAIOSFODNN7EXAMPLE"
  },
  "eventTime": "2023-05-22T07:42:30Z",
  "eventSource": "verifiedpermissions.amazonaws.com",
  "eventName": "CreatePolicy",
  "awsRegion": "us-west-2",
  "sourceIPAddress": "203.0.113.0",
  "userAgent": "aws-sdk-rust/0.55.2 os/linux lang/rust/1.69.0",
  "requestParameters": {
    "clientToken": "a1b2c3d4-e5f6-a1b2-c3d4-TOKEN1111111",
    "policyStoreId": "PSEXAMPLEabcdefg111111"
  },
  "responseElements": {
    "policyStoreId": "PSEXAMPLEabcdefg111111",
    "policyId": "SPEXAMPLEabcdefg111111",
    "policyType": "STATIC",
    "principal": {
        "entityType": "PhotoApp::Role",
        "entityId": "PhotoJudge"
    },
    "resource": {
        "entityType": "PhotoApp::Application",
        "entityId": "PhotoApp"
    },
    "lastUpdatedDate": "2023-05-22T07:42:30.70852Z",
    "createdDate": "2023-05-22T07:42:30.70852Z"
  },
  "requestID": "93ffa151-3841-4960-9af6-30a7f817ef93",
  "eventID": "30ab405f-3dff-43ff-8af9-f513829e8bde",
  "readOnly": false,
  "resources": [
    {
      "accountId": "123456789012",
      "type": "AWS::VerifiedPermissions::PolicyStore",
```

```
      "arn": "arn:aws:verifiedpermissions::123456789012:policy-store/
PSEXAMPLEabcdefg111111"
    }
  ],
  "eventType": "AwsApiCall",
  "managementEvent": true,
  "recipientAccountId": "123456789012",
  "eventCategory": "Management"
}
```

GetPolicy

```
{
  "eventVersion": "1.08",
  "userIdentity": {
    "type": "AssumedRole",
    "principalId": "EXAMPLE_PRINCIPAL_ID",
    "arn": "arn:aws:iam::123456789012:role/ExampleRole",
    "accountId": "123456789012",
    "accessKeyId": "AKIAIOSFODNN7EXAMPLE"
  },
  "eventTime": "2023-05-22T07:43:29Z",
  "eventSource": "verifiedpermissions.amazonaws.com",
  "eventName": "GetPolicy",
  "awsRegion": "us-west-2",
  "sourceIPAddress": "203.0.113.0",
  "userAgent": "aws-sdk-rust/0.55.2 os/linux lang/rust/1.69.0",
  "requestParameters": {
    "policyStoreId": "PSEXAMPLEabcdefg111111",
    "policyId": "SPEXAMPLEabcdefg111111"
  },
  "responseElements": null,
  "requestID": "23022a9e-2f5c-4dac-b653-59e6987f2fac",
  "eventID": "9b4d5037-bafa-4d57-b197-f46af83fc684",
  "readOnly": true,
  "resources": [
    {
      "accountId": "123456789012",
      "type": "AWS::VerifiedPermissions::PolicyStore",
      "arn": "arn:aws:verifiedpermissions::123456789012:policy-store/
PSEXAMPLEabcdefg111111"
    }
  ],
```

```
  "eventType": "AwsApiCall",
  "managementEvent": true,
  "recipientAccountId": "123456789012",
  "eventCategory": "Management"
}
```

CreateIdentitySource

```
{
  "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

```
{
  "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

```
{
  "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

```
{
  "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"
}
```

Ressourcen mit Amazon Verified Permissions erstellen mit AWS CloudFormation

Amazon Verified Permissions ist integriert AWS CloudFormation, ein Service, der Ihnen hilft, Ihre AWS Ressourcen zu modellieren und einzurichten, sodass Sie weniger Zeit mit der Erstellung und Verwaltung Ihrer Ressourcen und Infrastruktur verbringen müssen. Sie erstellen eine Vorlage, die alle gewünschten AWS Ressourcen beschreibt (z. B. Richtlinienspeicher) und diese Ressourcen für Sie AWS CloudFormation bereitstellt und konfiguriert.

Wenn Sie sie verwenden AWS CloudFormation, können Sie Ihre Vorlage wiederverwenden, um Ihre Ressourcen für verifizierte Berechtigungen konsistent und wiederholt einzurichten. Beschreiben Sie Ihre Ressourcen einmal und stellen Sie dann dieselben Ressourcen immer wieder in mehreren AWS-Konten Regionen bereit.

> **Important**
>
> Amazon Cognito Identity ist nicht in allen Versionen verfügbar AWS-Regionen wie Amazon Verified Permissions. Wenn Sie eine Fehlermeldung AWS CloudFormation bezüglich Amazon Cognito Identity erhalten, z. B. empfehlen wir Ihnen`Unrecognized resource types: AWS::Cognito::UserPool, AWS::Cognito::UserPoolClient`, den Amazon Cognito-Benutzerpool und den Client in der geografisch nächstgelegenen Region zu erstellen, AWS-Region an der Amazon Cognito Identity verfügbar ist. Verwenden Sie diesen neu erstellten Benutzerpool, wenn Sie die Identitätsquelle Verified Permissions erstellen.

Verifizierte Berechtigungen und AWS CloudFormation Vorlagen

Um Ressourcen für Verified Permissions und verwandte Dienste bereitzustellen und zu konfigurieren, müssen Sie sich mit AWS CloudFormation Vorlagen auskennen. Vorlagen sind formatierte Textdateien in JSON oder YAML. Diese Vorlagen beschreiben die Ressourcen, die Sie in Ihren AWS CloudFormation Stacks bereitstellen möchten. Wenn Sie mit JSON oder YAML nicht vertraut sind, können Sie AWS CloudFormation Designer verwenden, um Ihnen die ersten Schritte mit Vorlagen zu erleichtern. AWS CloudFormation Weitere Informationen finden Sie unter Was ist AWS CloudFormation Designer? im AWS CloudFormation Benutzerhandbuch.

Verified Permissions unterstützt das Erstellen von Identitätsquellen, Richtlinien, Richtlinienspeichern und Richtlinienvorlagen in AWS CloudFormation. Weitere Informationen, einschließlich Beispielen

für JSON- und YAML-Vorlagen für Ressourcen mit verifizierten Berechtigungen, finden Sie in der Referenz zum Ressourcentyp Amazon Verified Permissions im AWS CloudFormation Benutzerhandbuch.

AWS CDK-Konstrukte

Das AWS Cloud Development Kit (AWS CDK) ist ein Open-Source-Framework für die Softwareentwicklung, mit dem Cloud-Infrastruktur im Code definiert und bereitgestellt werden kann. AWS CloudFormation Konstrukte oder wiederverwendbare Cloud-Komponenten können zum Erstellen von Vorlagen verwendet werden. AWS CloudFormation Diese Vorlagen können dann zur Bereitstellung Ihrer Cloud-Infrastruktur verwendet werden.

Weitere Informationen und das Herunterladen von AWS CDKs finden Sie unter AWS Cloud Development Kit.

Im Folgenden finden Sie Links zur Dokumentation für AWS CDK Ressourcen mit verifizierten Berechtigungen, z. B. Konstrukte.

- Von Amazon verifizierte Berechtigungen L2 CDK Construct

Erfahren Sie mehr über AWS CloudFormation

Weitere Informationen AWS CloudFormation dazu finden Sie in den folgenden Ressourcen:

- AWS CloudFormation
- AWS CloudFormation Benutzerhandbuch
- AWS CloudFormation API Reference
- AWS CloudFormation Benutzerhandbuch für die Befehlszeilenschnittstelle

Zugriff auf Amazon Verified Permissions über einen AWS PrivateLink Schnittstellen-Endpunkt

Sie können verwendenAWS PrivateLink, um eine private Verbindung zwischen Ihrer VPC und Amazon Verified Permissions herzustellen. Sie können auf die verifizierten Berechtigungen wie in Ihrer VPC zugreifen, ohne dass dafür ein Internet-Gateway, ein NAT-Gerät, eine VPN-Verbindung oder AWS Direct Connect eine -Verbindung notwendig ist. Instances in Ihrer VPC benötigen keine öffentlichen IP-Adressen, um auf verifizierte Berechtigungen zuzugreifen.

Sie stellen diese private Verbindung her, indem Sie einen Schnittstellen-Endpunkt erstellen, der von AWS PrivateLink unterstützt wird. Wir erstellen eine Endpunkt-Netzwerkschnittstelle in jedem Subnetz, das Sie für den Schnittstellen-Endpunkt aktivieren. Diese sind vom Anforderer verwaltete Netzwerkschnittstellen, die als Eingangspunkt für den Datenverkehr dienen, der für verifizierte Berechtigungen bestimmt ist.

Weitere Informationen finden Sie unter Zugriff auf AWS-Services über AWS PrivateLink im AWS PrivateLink-Leitfaden.

Überlegungen zu verifizierten Berechtigungen

Bevor Sie einen Schnittstellen-Endpunkt für verifizierte Berechtigungen einrichten, lesen Sie die Überlegungen im AWS PrivateLink-Leitfaden.

Verified Permissions unterstützt Aufrufe all seiner API-Aktionen über den Schnittstellen-Endpunkt.

VPC-Endpunktrichtlinien werden für verifizierte Berechtigungen nicht unterstützt. Standardmäßig ist der vollständige Zugriff auf verifizierte Berechtigungen über den Schnittstellen-Endpunkt zulässig. Alternativ können Sie den Netzwerkschnittstellen des Endpunkts eine Sicherheitsgruppe zuordnen, um den Datenverkehr zu Verified Permissions über den Schnittstellen-Endpunkt zu steuern.

Erstellen Sie einen Schnittstellen-Endpunkt für verifizierte Berechtigungen

Sie können einen Schnittstellen-Endpunkt für verifizierte Berechtigungen entweder über die Amazon VPC-Konsole oder die AWS Command Line Interface (AWS CLI) () Weitere Informationen finden Sie unter Erstellen eines Schnittstellenendpunkts im AWS PrivateLink-Leitfaden.

Erstellen Sie einen Schnittstellen-Endpunkt für verifizierte Berechtigungen mit dem folgenden Service-Namen:

```
com.amazonaws.region.verifiedpermissions
```

Wenn Sie einen privaten DNS für den Schnittstellen-Endpunkt aktivieren, können Sie mittels seines standardmäßigen regionalen DNS-Namen, API-Anforderungen an Verified Permissions senden. Zum Beispiel , ., `verifiedpermissions.us-east-1.amazonaws.com`.

Kontingente für von Amazon verifizierte Berechtigungen

Ihr AWS-Konto hat Standardkontingente, früher als Limits bezeichnet, für jeden AWS Dienst. Wenn nicht anders angegeben, gilt jedes Kontingent spezifisch für eine Region. Sie können Erhöhungen für einige Kontingente beantragen und andere Kontingente können nicht erhöht werden.

Um die Kontingente für verifizierte Berechtigungen anzuzeigen, öffnen Sie die Konsole Service Quotas. Wählen Sie im Navigationsbereich AWS Dienste und dann Verifizierte Berechtigungen aus.

Informationen zur Erhöhung eines Kontingents finden Sie unter Anfordern einer Kontingenterhöhung im Benutzerhandbuch zu Service Quotas. Wenn das Kontingent unter Service Quotas noch nicht in verfügbar ist, verwenden Sie das Formular zur Erhöhung des Service-Limits.

Für Ihr AWS-Konto gelten die folgenden Kontingente in Bezug auf verifizierte Berechtigungen.

Themen

Kontingente für Ressourcen

Name	Standard	Anpas	Beschreibung
Policy-Shops pro Region und Konto	Jede unterstützte Region: 1 000	Ja	Die maximale Anzahl von Policy-Stores.
Richtlinienvorlagen pro Richtlinienspeicher	Jede unterstützte Region: 40	Ja	Die maximale Anzahl von Richtlinienvorlagen in einem Richtlinienspeicher.
Identitätsquellen pro Richtlinienspeicher	1	Nein	Die maximale Anzahl von Identitätsquellen, die Sie für einen Richtlinienspeicher definieren können.

Name	Standard	Anpas	Beschreibung
Größe der Autorisierungsanforderung[1]	1 MB	Nein	Die maximale Größe einer Autorisierungsanfr age.
Größe der Richtlinie	10,000 Bytes	Nein	Die maximale Größe einer einzelnen Police.
Größe des Schemas	100 000 Bytes	Nein	Die maximale Größe des Schemas eines Richtlini enspeichers.
Richtliniengröße pro Ressource	200.000 Byte[2]	Nein	Die maximale Größe aller Richtlinien, die auf eine bestimmte Ressource verweisen.

[1] Das Kontingent für eine Autorisierungsanfrage ist für IsAuthorizedund dasselbe IsAuthorizedWithToken.

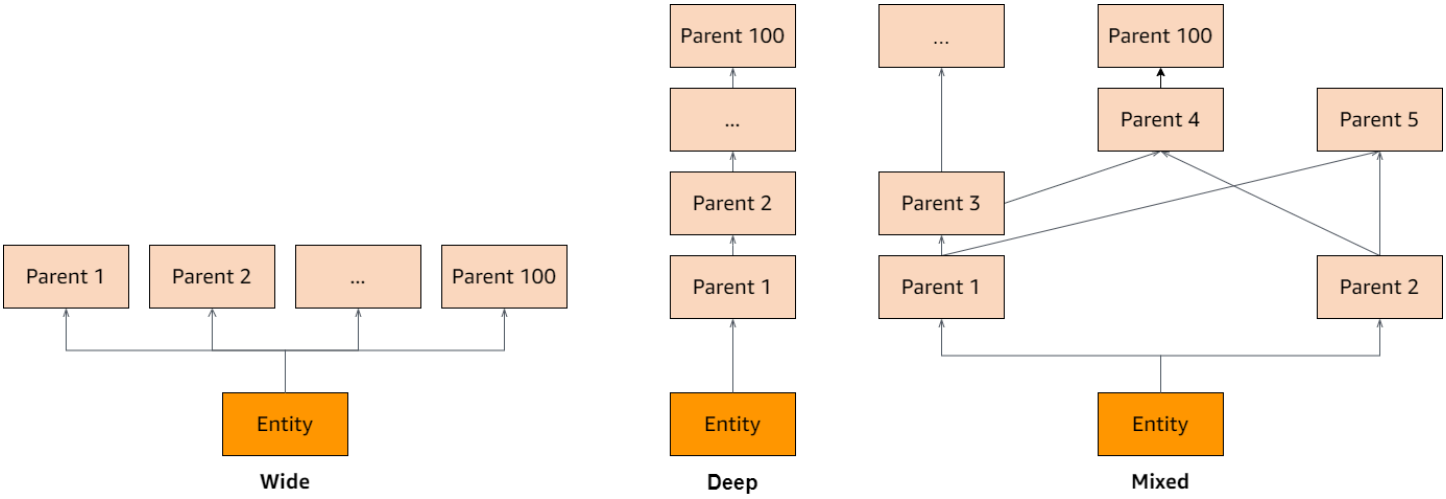
[2] Die Gesamtgröße aller Richtlinien, die sich auf eine einzelne Ressource beziehen, darf 200.000 Byte nicht überschreiten. Bei Richtlinien, die mit Vorlagen verknüpft sind, wird die Größe der Richtlinienvorlage nur einmal gezählt, zuzüglich der Größe jedes Parametersatzes, der zur Instanziierung jeder mit der Vorlage verknüpften Richtlinie verwendet wird.

Kontingente für Hierarchien

Name	Standard	Anpas	Beschreibung
Transitive Eltern pro Schulleiter	100	Nein	Die maximale Anzahl transitiver Eltern für jeden Schulleiter.

Name	Standard	Anpas	Beschreibung
Transitive Eltern pro Aktion	100	Nein	Die maximale Anzahl transitiver Eltern für jede Aktion.
Transitive Eltern pro Ressource	100	Nein	Die maximale Anzahl transitiver Eltern für jede Ressource.

Das folgende Diagramm zeigt, wie transitive Eltern für eine Entität (Prinzipal, Aktion oder Ressource) definiert werden können.



Kontingente für Operationen pro Sekunde

Verified Permissions drosselt Anfragen an Dienstendpunkte, AWS-Region wenn Anwendungsanfragen das Kontingent für einen API-Vorgang überschreiten. Verified Permissions gibt möglicherweise eine Ausnahme zurück, wenn Sie das Kontingent an Anfragen pro Sekunde überschreiten oder wenn Sie versuchen, gleichzeitig Schreibvorgänge auszuführen. Sie können Ihre aktuellen RPS-Kontingente unter Service Quotas einsehen. Um zu verhindern, dass Anwendungen das Kontingent für einen Vorgang überschreiten, müssen Sie sie für Wiederholungsversuche und exponentielles Backoff optimieren. Weitere Informationen finden Sie unter Wiederholen mit Backoff-Muster und Verwalten und Überwachen der API-Drosselung in Ihren Workloads.

Name	Standard	Anpas	Beschreibung
BatchIsAuthorized Anfragen pro Sekunde, Region, pro Konto	Jede unterstützte Region: 30	Ja	Die maximale Anzahl von BatchIsAuthorized Anfragen pro Sekunde.
BatchIsAuthorizedWithToken Anfragen pro Sekunde, pro Region, pro Konto	Jede unterstützte Region: 30	Yes (Ja)	Die maximale Anzahl von BatchIsAuthorizedWithToken Anfragen pro Sekunde.
CreatePolicy Anfragen pro Sekunde, pro Region, pro Konto	Jede unterstützte Region: 10	Ja	Die maximale Anzahl von CreatePolicy Anfragen pro Sekunde.
CreatePolicyStore Anfragen pro Sekunde, pro Region, pro Konto	Jede unterstützte Region: 1	Nein	Die maximale Anzahl von CreatePolicyStore Anfragen pro Sekunde.
CreatePolicyTemplate Anfragen pro Sekunde, pro Region, pro Konto	Jede unterstützte Region: 10	Ja	Die maximale Anzahl von CreatePolicyTemplate Anfragen pro Sekunde.
DeletePolicy Anfragen pro Sekunde, pro Region, pro Konto	Jede unterstützte Region: 10	Ja	Die maximale Anzahl von DeletePolicy Anfragen pro Sekunde.
DeletePolicyStore Anfragen pro Sekunde, pro Region, pro Konto	Jede unterstützte Region: 1	Nein	Die maximale Anzahl von DeletePolicyStore Anfragen pro Sekunde.
DeletePolicyTemplate Anfragen pro Sekunde, pro Region, pro Konto	Jede unterstützte Region: 10	Ja	Die maximale Anzahl von DeletePolicyTemplate Anfragen pro Sekunde.
GetPolicy Anfragen pro Sekunde, pro Region, pro Konto	Jede unterstützte Region: 10	Ja	Die maximale Anzahl von GetPolicy Anfragen pro Sekunde.

Name	Standard	Anpas	Beschreibung
GetPolicyTemplate Anfragen pro Sekunde, pro Region, pro Konto	Jede unterstützte Region: 10	Ja	Die maximale Anzahl von GetPolicyTemplate Anfragen pro Sekunde.
GetSchema Anfragen pro Sekunde, pro Region, pro Konto	Jede unterstützte Region: 10	Ja	Die maximale Anzahl von GetSchema Anfragen pro Sekunde.
IsAuthorized Anfragen pro Sekunde, pro Region, pro Konto	Jede unterstützte Region: 200	Ja	Die maximale Anzahl von IsAuthorized Anfragen pro Sekunde.
IsAuthorizedWithToken Anfragen pro Sekunde, pro Region, pro Konto	Jede unterstützte Region: 200	Ja	Die maximale Anzahl von IsAuthorizedWithToken Anfragen pro Sekunde.
ListPolicies Anfragen pro Sekunde, pro Region, pro Konto	Jede unterstützte Region: 10	Ja	Die maximale Anzahl von ListPolicies Anfragen pro Sekunde.
ListPolicyStores Anfragen pro Sekunde, pro Region, pro Konto	Jede unterstützte Region: 10	Ja	Die maximale Anzahl von ListPolicyStores Anfragen pro Sekunde.
ListPolicyTemplates Anfragen pro Sekunde, pro Region, pro Konto	Jede unterstützte Region: 10	Ja	Die maximale Anzahl von ListPolicyTemplates Anfragen pro Sekunde.
PutSchema Anfragen pro Sekunde, pro Region, pro Konto	Jede unterstützte Region: 10	Ja	Die maximale Anzahl von PutSchema Anfragen pro Sekunde.
UpdatePolicy Anfragen pro Sekunde, pro Region, pro Konto	Jede unterstützte Region: 10	Ja	Die maximale Anzahl von UpdatePolicy Anfragen pro Sekunde.

Name	Standard	Anpas	Beschreibung
UpdatePolicyTemplate Anfragen pro Sekunde, pro Region, pro Konto	Jede unterstützte Region: 10	Ja	Die maximale Anzahl von UpdatePolicyTemplate Anfragen pro Sekunde.

Dokumentenverlauf für das Amazon Verified Permissions User Guide

In der folgenden Tabelle werden die Dokumentationsversionen für Verified Permissions beschrieben.

Änderung	Beschreibung	Datum
OIDC-Identitätsquellen	Sie können jetzt Benutzer von OpenID Connect (OIDC) - Identitätsanbietern autorisieren.	8. Juni 2024
Batch-Autorisierung mit Identitätsquellen-Token	Sie können jetzt Benutzer aus einem Amazon Cognito Cognito-Benutzerpool in einer einzigen `BatchIsAuthorizedWithToken` API-Anfrage autorisieren.	5. April 2024
Einen Richtlinienspeicher mit API Gateway erstellen	Sie können jetzt einen Richtlinienspeicher aus einer vorhandenen API und einem Amazon Cognito Cognito-Benutzerpool erstellen.	1. April 2024
Kontextkonzepte und Beispiel	Es wurden Informationen zum Kontext in Autorisierungsanfragen mit verifizierten Berechtigungen hinzugefügt.	1. Februar 2024
Autorisierungskonzepte und Beispiel	Es wurden Informationen zu Autorisierungsanfragen mit verifizierten Berechtigungen hinzugefügt.	1. Februar 2024
AWS CloudFormation Integration	Verified Permissions unterstützt das Erstellen von Identität	30. Juni 2023

	squellen, Richtlinien, Richtlinienspeichern und Richtlinienvorlagen in AWS CloudFormation.	
Erstversion	Erste Version des Amazon Verified Permissions User Guide	13. Juni 2023

Die vorliegende Übersetzung wurde maschinell erstellt. Im Falle eines Konflikts oder eines Widerspruchs zwischen dieser übersetzten Fassung und der englischen Fassung (einschließlich infolge von Verzögerungen bei der Übersetzung) ist die englische Fassung maßgeblich.