ゼロトラスト認証認可連携における ユーザ同意付きコンテキスト共有

畠山 昂大[1] 小谷 大祐[1] 岡部 寿男[1]

概要：ネットワーク上のリソースのアクセス制御モデルとして境界型モデルが知られている。しかし、このモデルはアクセス元のネットワークによりアクセス制御を行うため、一旦侵入されてしまうと攻撃者にリソースへのアクセス権を与えてしまうなど問題がある。そこで、境界型モデルに代わる新しいセキュリティモデルとしてゼロトラストネットワーク (ZTN) が提案されている。このモデルでは、コンテキストと呼ばれるユーザやデバイスに関する様々な情報を用いてアクセス要求者を認証しアクセス要求を認可する (ゼロトラスト認証認可) ことでアクセス制御を行う。従って、このモデルでは正しいアクセス制御のために十分な量のコンテキストが必要となるが、Identity Federation により様々な組織が提供する多くのサービスを利用している場合には、コンテキストは組織のシステムに分散され管理されてしまうことが多い。そこで本研究では、Identity Federation 下で ZTN の概念を適用するゼロトラスト認証認可連携 (Zero Trust Federation; ZTF) という考え方を提唱し、組織を超えてコンテキストを共有する手法について提案する。加えて、コンテキストはユーザのプライバシー情報を多く含むため、この共有をユーザの制御下におけるような仕組み、ユーザの同意を伴う仕組みを提案する。さらに、本研究では ZTF のプロトタイプを実装し、その動作確認を行なった。

Sharing Context with User Consent in Zero Trust Federation

Koudai Hatakeyama[1] Daisuke Kotani[1] Yasuo Okabe[1]

1. 緒論

組織が情報システムへのアクセスやデータ (リソース) をセキュアに管理するために、誰がどのリソースへアクセスできるか判断するためのアクセス制御が必要である。今日、アクセス制御のモデルとして境界型モデル [11][5] が普及している。このモデルではネットワークを信頼のレベルに応じて分割し、アクセス元のネットワークに基づきアクセスの可否を判断することで制御を行う。

しかし、近年このモデルでは対応しきれない問題が発生している。例えば、モバイル端末の普及やリモートワークの促進により、信頼のレベルに応じて分割された組織内ネットワークからのアクセスを前提とすることが難しくなったことがある。また、クラウドの利用により組織が利用するリソースはインターネットのような組織外のネットワークに存在するようになり、アクセスのために組織内ネットワークを介する必然性が薄れてしまうことや、境界内ではネットワークのアクセス制御が行われないため、一度あるリソースが乗っ取られてしまえば攻撃者のアクセスをそのネットワークからのアクセスとして偽装できてしまうことも挙げられる。

こうしたモデル自体の問題に対応するため、ゼロトラストネットワーク (ZTN) という新しいアクセス制御モデルが提案されている [17]。このモデルでは、アクセス元のネットワークのような常に信頼できる属性などを仮定せず、アクセス要求の度に要求者を検証し信頼に値するか評価する。そして、その評価結果に基づきアクセスの可否を判断することで制御を行う。

アクセス要求者の検証・評価をするときは、「コンテキスト」を用いる。コンテキストとはアクセス要求に付随する情報のことで、ユーザ ID といった従来の identity 情報に関することに加え、使用デバイスや最新のセキュリティアップデートが適用されているかといったデバイスに関す

[1] 京都大学

ることも含まれる。さらに、最近アクセスしたデバイスは何か、どこからアクセスしていたかという過去の挙動に基づく情報等も含まれる [3]。

ZTN の実装として、単一の管理者が運用するシステムのリソースへのアクセス全てを監視するコントロールプレーンと呼ばれるものを用意し、そこで一元的にアクセス制御するよう設計することがある [4]。その際問題となるのは、このシステムがどのようにして ZTN のコントロールプレーンを管理・運営するかである。その中でも本研究ではコントロールプレーンがどのようにアクセス制御の材料となるコンテキストを収集するかについて着目する。コントロールプレーンが収集できるコンテキストはあくまでも監視の届くシステム内に存在するものだけであり、システム外に存在するアクセス要求者のコンテキストは収集することができない。特に Identity Federation 下で identity 情報を交換しつつ異なる管理者が運営するシステムのサービスを利用する場合、アクセス要求者の identity と紐づくコンテキストは複数のシステムのコントロールプレーンで分散して管理されてしまう。コンテキストも identity 情報と同じく共有すれば良いのだが、コンテキストはアクセス要求者のプライバシーを含む情報のため、単純に共有することはできないといった問題がある。

本来、コンテキストは単一の管理者が運用するシステムに限らずアクセス要求者が利用する様々なシステムから収集しなければならないが、現状の ZTN の実装 [21] ではコントロールプレーンは単一のシステムが収集できる限りのコンテキストしか扱うことができていない。そこで、本研究では単一の管理者が運用するシステムを超えてコンテキストを共有する手法について提案する。加えて、ユーザのプライバシーに配慮するために本研究では単一の管理者が運用するシステムを超えたコンテキストの共有をユーザの制御下におけるような仕組み、つまりユーザの同意を伴う仕組みを提案する。コンテキストの共有は Continuous Access Evaluation Protocol[20] と呼ばれる現在標準化が進められているイベント共有プロトコルを使うことを前提とし、ユーザの同意を伴う仕組みとして OAuth2.0[6] と呼ばれる権限委譲プロトコルを使用して設計・実装を行なった。

本研究の主な貢献は、Identity Federation 下で ZTN の概念を適用するゼロトラスト認証認可連携 (Zero Trust Federation; ZTF) という考え方を提唱したこと、ZTF でコンテキストを共有する際にユーザの同意を得てから共有するための方法を示したことである。

本論文の構成は次のとおりである。第 2 章では本研究の背景となる技術に関して説明し、第 3 章ではゼロトラスト認証認可連携 (ZTF) という新しい概念について説明する。第 4 章では ZTF においてユーザ同意を伴いつつコンテキスト共有を実現するシステムについて提案する。第 5 章でその評価を実装したプロトタイプを使ったユースケースとともに説明する。最後に第 6 章で本論文をまとめる。

2. 背景

2.1 ゼロトラストネットワーク (ZTN) モデル

ZTN は “Never Trust, Always Verify” を重要視するネットワークにおけるアクセス制御モデルである。ZTN は従来の境界型モデルにおけるアクセス要求元のネットワークのような特定の属性を信頼してアクセス制御を行うことはしない (“Never Trust”)。代わりに、アクセス要求がある度に要求者を検証 (“Always Verify”) することでアクセス制御を行う。検証にはコンテキストが使われる。コンテキストとはリクエストに付随する情報の集合であり、アクセス要求者に関する情報・アクセス要求者が使用しているデバイスに関する情報など様々なものが考えられる。

2.1.1 ZTN のアーキテクチャ

Gilman ら [4] によると ZTN はコントロールプレーンとデータプレーンの二つから構成される。コントロールプレーンは保護対象リソースへのアクセスを許可するかの判断を行う。判断の主体を Policy Decision Point (PDP) という [17]。データプレーンはアクセス要求者がリソースと通信を行う場所で、アクセス制御の実施点がある場所でもある。アクセス制御実施点を Policy Enforcement Point (PEP) という [17]。

2.1.2 ZTN の実装例: BeyondCorp

BeyondCorp とは Google が ZTN を自社ネットワークに実装したもので、その構想はいくつかのホワイトペーパーにまとめられている [21][15][18]。この実装で特徴的なのは、PEP と PDP を一つにまとめた大きなアクセスプロキシ [18] を用意して ZTN を実現していることである。Identity-Aware Proxy (IAP)[*1] と呼ばれるこの (リバース) プロキシは、アクセスの検証を一手に引き受ける。検証の結果認められれば IAP はアクセス要求者と保護対象リソースの間にセキュアな通信路を確立する。拒否されると IAP はアクセスを遮断する。

PEP と PDP を一つにまとめることで、ポリシーの変更を素早くそして一貫して変更することができるようになった [18]。その一方で、異なる管理者が運用するシステム間で IAP を共に利用してアクセス判断に足る十分な量のコンテキストを確保しようとした場合、それらシステムのサービス全てが一つの IAP を介して通信するため、複数の管理者にとっての単一障害点となり得ることや、共通の性能上のボトルネックとなってしまうことなど課題が残されている。

2.2 コンテキストを使った認証認可

2.1 節で述べたように、ZTN では、アクセス制御のため

[*1] https://cloud.google.com/iap/

にコンテキストを用いてアクセス要求者を認証しアクセス要求を認可する。ここではコンテキストを使った認証認可について先行研究を取り上げながら説明する。

コンテキストを使った認証はリスクベース認証と呼ばれることもある。リスクベース認証に使われるコンテキストの例として、IP アドレス [2]、マウス操作やキーストローク [19] がある。これら認証ではコンテキストを使うことに加えて、継続的にユーザであるか検証していることも特徴的である [19]。これは Continuous Authentication とも呼ばれ、研究が進められている [3]。例えばタッチスクリーンの操作履歴を収集し、現在の操作しているユーザが以前の履歴と同じ特性を持つかで継続的な認証を行うものがある [3]。さらに、ユーザと認証者の一対一の認証を考えるだけでなく、[16] や [20] など認証連携下で行う Continuous Authentication の研究も進められている。

他に、コンテキストを使った認可として、現在本社にいるスタッフのみにデータベースアクセスを許可する [10]、デバイスが置かれている実際の環境に応じて認可を決定する [1] などがある。

2.3 Idenitity Federation (IdF)

Identity Federation (IdF) とは組織間でユーザの identity を共有する仕組みで、identity とはユーザ ID やユーザ属性といったユーザに関する情報のことを指す。IdF を大学間で構築した例として学認[*2]がある。学認によって、ユーザは異なる大学のサービスに新規ユーザ登録などせずシームレスにアクセスできるようになる、例えば異なる大学の無線 LAN を新たにユーザ登録せず使用できるなど、identity を共有することで一つの大学だけでは提供しきれない様々なサービスを利用できるようになる。

IdF を構築するために SAML[12] を用いた Shibboleth[*3] や OpenID Connect[13] など様々な技術が発展している。

2.3.1 Continuous Access Evaluation Protocol (CAEP)

CAEP とは、IdF で Continuous Authentication を行うためのイベント共有プロトコルとして、Google が発表した新しいプロトコルである [20]。このプロトコルでは、IdF に参加している組織を Relying Party (RP) と呼び、Identity Provider を IdP と呼ぶ。前提として、RP が IdP から提供された認証情報を使用してユーザを識別しているとする。そして CAEP を使うことで、RP は内部で発生したユーザに関するイベントを、認証を行なった IdP と共有することができる。イベントとはユーザに関する出来事で、例えば、使用しているネットワークが変わった、使用デバイスで脆弱性が発見された、などがある。つまり、IdP はこのプロトコルなしでは知ることのできなかった RP 内部でのユーザのコンテキストの更新を知ることができ、それを元に Continuous Authentication を行うことができる。さらに、IdP は CAEP を使うことで、認証の結果とそれに付随する情報、例えば、他の RP のためにより強い認証が行われ AAL(Authenticator Assurance Level[*4]) が変化したこと、他の RP で不審な挙動があったこと、などを RP へ通知することができる。RP はそれらの情報をもとにそのユーザに対する認可判断をすることができる。

現在、このプロトコルは OpenID Foundation の Shared Signals and Events ワーキンググループ[*5]にて標準化が進められている。

2.3.2 連携と同意

組織同士が連携し共有する内容がユーザのプライバシーに関する情報であれば、ユーザの同意なく共有することはできない。例えば、学認ではユーザの仮名 ID だけでなくユーザの属性を共有することがある。ユーザの属性とは、ユーザのメールアドレスや、大学生かどうかといった情報で、これはユーザのプライバシーに関する情報である。uApproveJP[14] を用いることで、IdF 環境においてユーザの同意を伴うユーザ属性の共有が行えるようになる。

ユーザの同意とは、第三者が自分の代わりに何かすることに対して同意することである。これは、第三者に自分の代わりに何かをして良いという権限を付与していることとみなせる。これは権限委譲といい、OAuth2.0 と呼ばれるプロトコルで実装されている [6]。OAuth2.0 は [22] のような変遷を経て IETF によって策定され、権限委譲するためのデファクトスタンダードとなっている。

3. ゼロトラスト認証認可連携

この章ではゼロトラスト認証認可連携 (Zero Trust Federation; ZTF) を定義し、ZTF の設計と実装について提案する。

3.1 問題設定

ゼロトラストネットワーク (ZTN) を実装する時、アクセス要求の可否を判断するために用いるコンテキストをどう収集し管理するかが重要となる。2.1.1 節のような ZTN アーキテクチャに従うとすると、コントロールプレーンがコンテキストを収集する。しかしコントロールプレーンは、ZTN を構築する管理者の運用が届く範囲でしか監視をすることができず、十分な量のコンテキストを収集できない場合がある。特に、SaaS (Service as a Service) など異なる管理者により運用されているサービスを多く利用している場合、SaaS 内で発生したコンテキストを一元的に収集することができない。しかし、ZTN で十分な量のコンテキストが収集できなければ、正しくアクセス制御するこ

[*2] https://www.gakunin.jp/
[*3] https://www.shibboleth.net/
[*4] https://pages.nist.gov/800-63-3/sp800-63b.html
[*5] https://openid.net/wg/sse/

とが難しくなってしまう。

そこで、本論文では管理者の異なるシステム同士がそれぞれのシステムで ZTN を実現するための連携について考える。特にコンテキストは identity と深く関連する情報であるため、Identity Federation 下にいるシステム同士の連携について考える。そのために新しい概念としてゼロトラスト認証認可連携を定義し、それの実装方法について検討する。

3.2 ゼロトラスト認証認可連携の定義

本研究でゼロトラスト認証認可連携 (Zero Trust Federation; ZTF) とは、Identity Federation (IdF) 下で複数の Identity Provider(IdP) と Relying Party(RP) がゼロトラストネットワーク (ZTN) の考えに従って認証認可を行う連携のことを指す。ここで RP とは単一の管理者が運用するシステムのことを指す。また、この連携のもと行われる認証認可のことをゼロトラスト認証認可と呼ぶ。

RP がゼロトラスト認証認可によるアクセス制御を行うためにはコンテキストが必要である。ここでいうコンテキストとは identity を包含する情報のことで、identity から取得できるユーザ属性だけでなくユーザが使用しているデバイスの情報や、ユーザやデバイス周辺の環境情報など様々な情報からなる。

コンテキストは IdP や RP とも異なる別の entity が管理することもある。例えばデバイス情報であればモバイルデバイスマネジメント (MDM) サービスが、デバイスの安全性は Endpoint Detection and Response (EDR) サービス等がそれぞれコンテキストを収集・管理している。ここではそのようなコンテキストを管理する entity を Context Attribute Provider (CAP) と呼ぶことにする。さらに、これら CAP で保持しているコンテキストを RP が利用できるようにするための連携を Context Federation と呼ぶこととする。

以上のことをまとめて、ZTF を、Identity Federation (IdF) と Context Federation (CtxF) から構成されるフェデレーションのことと定義する。ZTF の全体像を図 1 に表す。IdP は RP や CAP へ identity を提供し、CAP は RP へコンテキストを提供する。

さらに ZTF において、RP がアクセス制御のためにゼロトラスト認証認可を行う場合は図 2 のような構成になる。ユーザはあるサービスにアクセス要求を行うと RP 内の PDP がユーザの identity を IdP から取得し、コンテキストを CAP から取得する。IdP は identity を提供する前にユーザの認証を行い、CAP はコンテキストをユーザから収集しておく。そして PDP はコンテキストを元にアクセス可否を判断し PEP がその判断に基づきアクセス制御を行う。

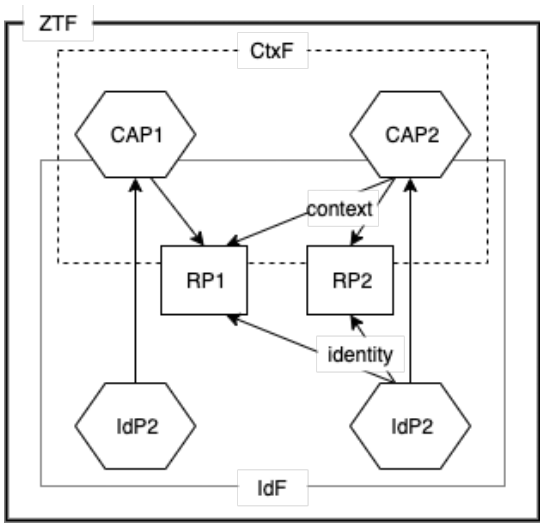

図 1　ZTF の全体像

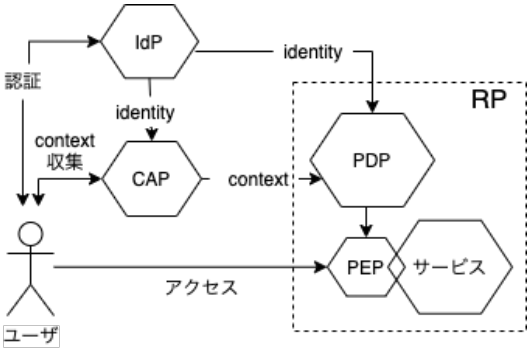

図 2　ZTF 下の RP の ZTN 構成

3.3 ZTF の設計方針

3.3.1 Context Attribute Provider (CAP)

Context Attribute Provider (CAP) はコンテキストを 1）RP へ提供し、2）RP から収集を行う、CtxF の entity である。2）について、CAP はユーザとのインタラクションが RP と比較して少なく、直接ユーザからコンテキストを収集できないため、2）のような設計としている。

1）のコンテキストの提供は次のように行われる。各 RP は CAP に対してアクセスしてきたユーザに関するコンテキストの提供を依頼する。そして CAP は依頼を受け、このコンテキストをその RP へ提供する。さらに CAP がこのコンテキストの更新を検知すると、その更新されたコンテキストを RP へ提供する。

2）のコンテキストの収集は次のように行われる。あるユーザに関するコンテキストの提供を依頼してきた RP に対して、逆にそのユーザに関するコンテキストの提供を依頼する。RP はその依頼を受け、このユーザに関するコンテキストを RP へ提供する。さらに RP がこのコンテキストの更新を検知すると、その更新されたコンテキストを CAP へ提供する。

3.3.2 ZTF でのアクセス制御までの流れ

ZTF においてユーザの動きに注目し、ユーザが RP1 に

アクセス要求をしてからアクセス可否の判断が下されるまでのフローを図 3 に示す。なお、過去にユーザは RP2 へアクセスしているものとする。

(1) ユーザは RP1 へのアクセスを試みる
(2) ユーザは IdP を通して identity (id) を RP1 へ提供する
(3) ユーザは CAP を通してコンテキスト (ctx) を RP1 へ提供する
(4) RP1 は取得したコンテキストを元にアクセス制御を行う
(5) RP1 はユーザの振る舞いを監視し、コンテキストに変化があればそれを CAP に通知する。CAP は通知されたコンテキストを RP2 と連携する。

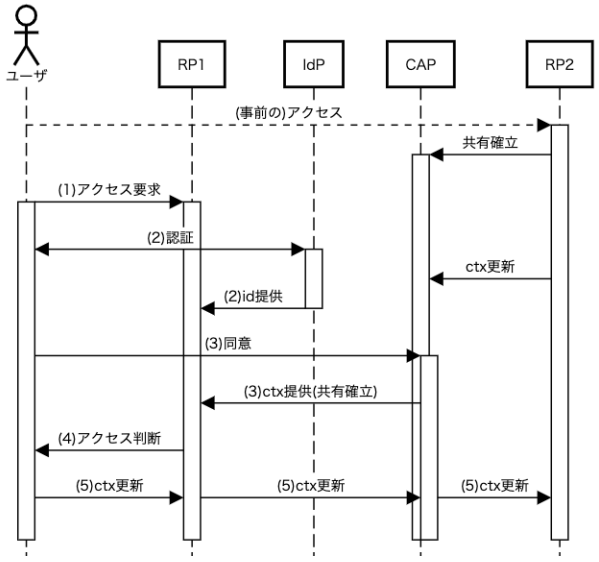

図 3 アクセスの可否が判断されるまでの流れ

3.4 ZTF の実装

以上より、ZTF を実装するためには identity の提供とコンテキストの共有の仕組みが必要である。本論文では、identity の提供には OpenID Connect と呼ばれるアイデンティティ連携プロトコル [13] を、コンテキスト共有には CAEP と呼ばれるイベント共有プロトコル [20] の使用を想定しているが、CAEP の標準化はまだなされておらず仕様が存在しないため、現段階で議論されている設計を元に仕様が策定されることを想定して実装した。

3.4.1 idenitity の提供

identity の提供はユーザの認証を行なった IdP から CAP、RP に対して行われる。実装の都合上、IdP から直接 identity を提供する先は CAP とし、RP はそれを使い回すこととした。よって、identity の提供を行うフローには、ユーザ、IdP、CAP の三者が登場する。ここで OpenID Connect (OIDC) において、IdP は OpenID Provider (OP)、ユーザは End-User、CAP は Relying Party (RP)、identity は ID-token に対応する。この節に限り、わかりやすさのため ZTF の各 entity を OIDC の用語で表現することとする。RP が OP から End-User の ID-token を取得するまでの流れは図 4 のようになる。このフローは [13] の 3.1;Authentication using Authorization Code Grant Flow に従っている。

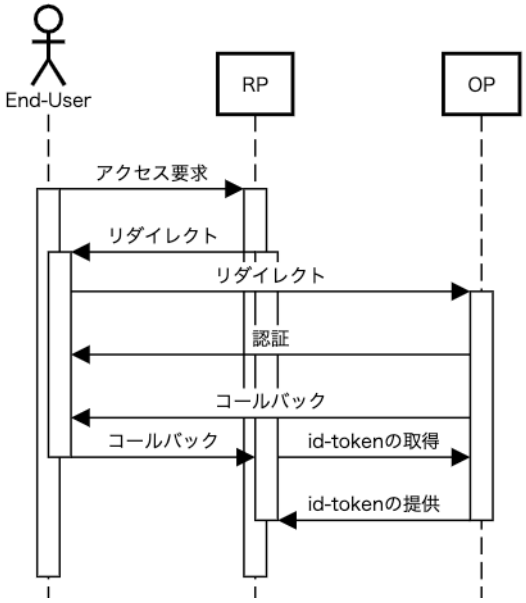

図 4 OIDC を用いた ID-token 取得フロー

3.4.2 コンテキスト共有

コンテキスト共有には 2 つの通信チャネルが用意される。1）CAP が RP へコンテキストを提供するチャネルと、2）RP が CAP へコンテキストを提供するチャネルである。このチャネルでは CAEP を利用して通信することを想定している。

1）については、RP がコンテキストを受け取るためのエンドポイントを CAP に登録することで実装する。エンドポイントを登録することで、RP はコンテキストの更新をリアクティブに受け取ることができる。

2）については、RP が CAP の提供しているエンドポイントに対して更新のあったコンテキストを送信することで実装する。

コンテキストの共有の際は、CAEP で提案されているように SET[7] と呼ばれる拡張 JWT を用いる。例えば、ユーザの位置情報 (`user:location`) が更新されたことを CAP が認識し、RP に送ることを考える。この時、RP は図 5 のような SET を受け取ることになる。ただし簡単のため SET の JWT クレームのみ記述している。

4. ユーザ同意を伴うコンテキスト共有

この章では、ZTF においてユーザ同意を伴うコンテキスト共有を実現するシステムについて説明する。

```
{
  "iss": "https://cap.demo",
  "aud": "https://rp.demo",
  "sub": "alice",
  "events": {
    "cap:user:location:raw":{
      "latitude": 1234,
      "longitude": 6789
    }
  }
}
```

図 5　SET クレーム

4.1　問題設定

ZTF において、RP は CAP からコンテキストを提供され、ゼロトラスト認証認可に基づくアクセス制御を行うことができる。しかし、コンテキストとはユーザのプライバシーを含む情報であるため、コンテキストを扱う際にはユーザの同意が必要である。そのために、コンテキスト共有をする前にユーザの同意を得る方法を考える。

4.2　設計方針

4.2.1　ユーザ同意取得フロー

ユーザ同意を伴うコンテキストの共有とは、コンテキストの共有がユーザの同意を受けて初めて行われ、同意を受けた範囲内でコンテキストを共有することで実現される。従って、ZTF でプライバシーを考慮する際は、CAP から RP へのコンテキスト提供、RP から CAP へのコンテキスト提供双方についてそれぞれユーザの同意があれば良い。

4.2.2　同意の程度

コンテキスト共有に対して行う同意の選択肢が「はい」か「いいえ」の 2 値では、ユーザは共有される情報を細かく制御することができない。コンテキストにはユーザに関する様々な情報が含まれる。それぞれの情報に対するプライバシーの感覚はユーザにより異なると想定されるため、それぞれの情報に対してどの程度の粒度で共有を許すかをユーザが制御できることが望ましい。

例えば、ユーザが最近いた地理的場所についての情報をコンテキストに含めて共有したいと RP が求めてきたとする。この時、緯度経度レベルの情報が必要なこともあれば、国レベルの情報で十分であるかもしれないし、「ユーザは日本にいるかどうか」のような条件式に対する真理値で十分であるかもしれない。各 RP が必要とする情報の粒度はユースケースによって様々であるため一様に定めることはできないが、ユーザがその粒度を制御できることは必要である。

そこで、用意したコンテキスト情報のそれぞれに対して、raw、predicate という 2 段階の粒度を用意することにした。raw レベルで同意した場合、CAP や RP が保存している情報をそのまま共有する。ユーザの最近いた地理的場所を緯度経度で保存していれば 2 値座標を共有することになり、国を単位として保存していれば国情報を共有する。predicate レベルで同意する場合、CAP が用意しているどの条件式を用いてコンテキスト情報を共有するかも一緒に指定する。

例えばユーザの地理的情報に関して、1）ユーザが日本にいたか、2）ユーザが京都大学にいたかといった 2 つの条件式を CAP が用意していたとする。この場合、ユーザが predicate レベルで同意する際は 1 、2 どちらを使うかも一緒に指定する。

4.2.3　同意の撤回

例えば、コンテキストの共有に同意した RP を信頼できなくなった場合や、RP のサービスの利用を停止する場合など、ユーザが自分のタイミングでコンテキストの共有を停止させたい場合がある。よって共有を停止、すなわち CAP には同意を任意のタイミングで撤回できるような仕組みが必要となる。一方で、同意の撤回が攻撃者を利することはあってはならない。つまり、同意が撤回されコンテキストの提供が停止されたことで RP がアクセスを誤って許可することはあってはならない。

そこで、同意が撤回され共有が停止された場合、同意が撤回されたコンテキストを最悪のケースとして RP が処理するようにした。例えば、RP がコンテキストの一つとしてユーザの現在地を使って、「日本からのアクセスであればアクセスを許可」のようなポリシーを運用していた場合を考える。この時 RP はユーザの現在地追跡を行う CAP からコンテキストを提供されるようにユーザに対して同意を求める。一度同意を得られた後に、ユーザが何らかの理由でその同意を撤回したとする。すると、RP はそのユーザの現在地に関するコンテキストを収集できないため、最悪ケース、つまり海外からのアクセスであるとみなしアクセス制御を行う。

4.3　実装

4.3.1　ユーザ同意

コンテキスト共有のユーザ同意については OAuth2.0 と呼ばれる権限移譲プロトコル [6] を用いて実装する。

4.2 節で述べたように、実装すべきユーザ同意は 2 種類ある。1）RP から CAP へコンテキスト提供に関する同意と、2）CAP から RP へコンテキスト提供に関する同意である。

1）について、ユーザは RP に対して RP が収集しているコンテキストを CAP へ提供することに対して同意する。

2）について、ユーザは CAP に対して RP がコンテキストを CAP から取得することに同意するため、同意結果を委譲するためのプロトコルが必要となる。よって、OAuth2.0 という権限委譲プロトコル [6] を用いる。2）でユーザの同意を RP が取得できるまでのフローを図 6 に示す。

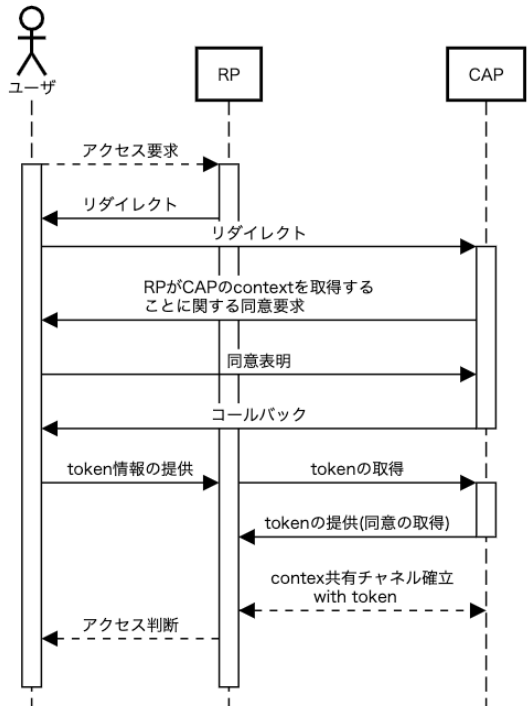

図 6　ユーザ同意取得フロー

4.3.2　同意の撤回

同意の撤回については、OAuth2.0 の補足として提案されている OAuth2.0 Token Revocation[9] に従って実装する。

5.　ユースケースと考察

5.1　プロトタイプ

今回、3.4 節と 4.3 節で示した実装を満たすプロトタイプを用意した。なおプロトタイプは Golang [*6] を用いて作成し、Golang 標準パッケージ群と外部ライブラリとして jwt-go [*7] と呼ばれる JWT[8] 用ライブラリを用いて作成した。また、`macOS Mojave10.14.6` のローカル環境で動作を確認した。今回作成したソースコードは GitHub[*8] で公開している。

5.2　ユースケース

5.1 節のプロトタイプを用いて、次のようなユースケースでコンテキストの共有が行えていることを確認した。CAP、RP1、RP2、IdP の 4 つの entity とユーザがいる状況において、次のユースケースを考える。

(1) ユーザは RP1 にアクセスする
(2) ユーザは RP2 にアクセスする
(3) ユーザは RP1 の管理下でコンテキストを更新する
(4) RP2 は CAP を介して更新されたコンテキストを共有する

このシナリオにより、RP2 が ZTF に参加していなければ認識できないコンテキストをユーザの同意のもと共有できていることを確認した。動作例を HTTP メッセージとともに説明したものは Github[*9] で公開している。

5.3　考察

前節で確認したように、本提案によってユーザの同意を伴うコンテキスト共有を行うことができた。コンテキストもユーザの属性であると考えると、uApproveJP[14] というユーザ同意を伴う属性情報連携システムと本提案を比較することができる。uApproveJP も本提案もユーザの同意を得てコンテキスト (ユーザ属性) を提供するという仕組み自体は同じであるが、扱う情報の更新頻度に違いがある。今回提案している ZTF においては、RP や CAP 間で共有するコンテキストは頻繁に変化すると考えられる。uApproveJP には同意を取得するユーザ属性の値はあまり更新されないという前提があり、デフォルトの設定として uApproveJP では属性の値が変化するたびに同意を取り直すための仕組みが用意されているため、uApproveJP を使ってしまうと頻繁に再同意する必要があり、ユーザの利便性を低下させてしまう。一方、本提案ではユーザの同意の撤回はユーザに委ねられており、その点でユーザの利便性を低下させることはない。

6.　結論

本研究では、Identity Federation 下でゼロトラストネットワーク (ZTN) を構築するために必要な連携 (ゼロトラスト認証認可連携; ZTF) と、連携するためのシステムについて提案し、プロトタイプを作成し実際に動作を確認した。

Identity Federation 下の ZTN は、アクセス要求者の振る舞いに基づいたコンテキストを組織間で連携することで実現される。コンテキスト間で連携する際は、コンテキストの更新をリアクティブに共有すること、コンテキストの共有はユーザの制御下にあること、の 2 つが重要である。前者については CAEP という現在標準化が進められているプロトコルがあり、本論文執筆時に明らかになっている情報をもとに設計した。後者については OAuth2.0 という権限委譲プロトコルを用いて、共有を始める前にユーザの同意を必ず取れるように設計した。また、ユーザの同意を取る際は、同意する内容をユーザがきめ細かく選択できることが重要である。そのため本提案ではコンテキストの情報ごとにそもそも同意を拒否するか、同意する場合は生のデータのまま提供するか、何らかの形で加工した結果で提供するか選択できるようにした。

さらに、提案手法により ZTF においては組織間でコンテキストを共有しアクセス要求の検証に用いることができ

[*6] `https://golang.org/`
[*7] `https://github.com/dgrijalva/jwt-go`
[*8] `https://github.com/hatake5051/ztf-prototype/tree/dicomo2020`
[*9] `https://github.com/hatake5051/ztf-prototype/blob/dicomo2020/doc/usecase.md`

ることを確認した。

今後の課題として、1）コンテキストの標準化、2）同意のきめ細やかさをどう定義するか、3）ZTF での名寄せの防止がある。

1）に関して、実際に RP-CAP 間でコンテキストを連携し合う際、何をコンテキストとして共有し合うかについて明確な基準が必要である。それは学認において提供する属性に標準[*10]があるように、コンテキストに対しても RP 側が使いやすいような標準を定める必要がある。

2）に関して、本提案では raw と predicate という 2 種類のレベルを用意した。特に predicate レベルでは条件式というものを導入し、コンテキストを生のまま共有することを避けるような同意の方法を用意することができた。この考え方を発展させ、コンテキスト処理の結果が真理値以外のものなども用意できれば、よりユーザの意図に近い同意を表明することができると考えられる。

3）に関して、Identity Federation では同じユーザでも異なる RP に対して異なるユーザ ID(仮名 ID) を提供することがある。これは、複数の RP が結託してユーザの名寄せを行い、プライバシーを侵害してしまうことを防ぐためにある。ZTF でも同じように異なる RP に対して仮名 ID を提供し、名寄せの防止を行う必要がある。しかし、ZTF では異なる RP に対してユーザに関するコンテキストを提供するため、ユーザ ID の仮名性に加えて名寄せを防止できるような別の仕組みを検討する必要がある。

参考文献

[1] Covington, M. J., Long, W., Srinivasan, S., Dev, A. K., Ahamad, M. and Abowd, G. D.: Securing Context-Aware Applications Using Environment Roles, *Proceedings of the Sixth ACM Symposium on Access Control Models and Technologies*, New York, NY, USA, Association for Computing Machinery, p. 10–20 (online), DOI: 10.1145/373256.373258 (2001).

[2] Diep, N. N., Lee, S., Lee, Y.-K. and Lee, H.: Contextual Risk-Based Access Control., *Security and Management*, Vol. 2007, pp. 406–412 (2007).

[3] Frank, M., Biedert, R., Ma, E., Martinovic, I. and Song, D.: Touchalytics: On the Applicability of Touchscreen Input as a Behavioral Biometric for Continuous Authentication, *IEEE Transactions on Information Forensics and Security*, Vol. 8, No. 1, pp. 136–148 (online), DOI: 10.1109/TIFS.2012.2225048 (2013).

[4] Gilman, E. and Barth, D.: *Zero Trust Networks Building Secure Systems in Untrusted Networks*, O'Reilly Media (2017).

[5] Goldberg, D. S.: The MITRE security perimeter, *Tenth Annual Computer Security Applications Conference*, pp. 212–218 (online), DOI: 10.1109/CSAC.1994.367306 (1994).

[6] Hardt, D.: The OAuth 2.0 Authorization Framework, RFC 6749 (2012).

[7] Hunt, P., Jones, M., Denniss, W. and Ansari, M.: Security Event Token (SET), RFC 8417 (2018).

[8] Jones, M., Bradley, J. and Sakimura, N.: JSON Web Token (JWT), RFC 7519 (2015).

[9] Lodderstedt, T., Dronia, S. and Scurtescu, M.: OAuth 2.0 Token Revocation, RFC 7009 (2013).

[10] Minami, K. and Kotz, D.: Secure context-sensitive authorization, *Pervasive and Mobile Computing*, Vol. 1, No. 1, pp. 123 – 156 (online), DOI: "https://doi.org/10.1016/j.pmcj.2005.01.004" (2005).

[11] Northcutt, S., Zeltser, L., Winters, S., Kent, K. and Ritchey, R. W.: *Inside Network Perimeter Security (2nd Edition) (Inside)*, Sams, USA (2005).

[12] OASIS Security Services TC: Security Assertion Markup Language (SAML) v2.0, http://docs.oasis-open.org/security/saml/v2.0/sstc-saml-approved-errata-2.0.pdf (2005).

[13] OpenID Foundation: Final: OpenID Connect Core 1.0 incorporating errata set 1, https://openid.net/specs/openid-connect-core-1_0.html (2014).

[14] Orawiwattanakul, T., Yamaji, K., Nakamura, M., Kataoka, T. and Sonehara, N.: User consent acquisition system for Japanese Shibboleth-based academic federation (GakuNin), *International Journal of Grid and Utility Computing*, Vol. 2, No. 4, pp. 284–294 (2011).

[15] Osborn, B., McWilliams, J., Beyer, B. and Saltonstall, M.: BeyondCorp: Design to Deployment at Google, *;login:*, Vol. 41, pp. 28–34 (online), available from ⟨https://www.usenix.org/publications/login/spring2016/osborn⟩ (2016).

[16] Preuveneers, D. and Joosen, W.: SmartAuth: Dynamic Context Fingerprinting for Continuous User Authentication, *Proceedings of the 30th Annual ACM Symposium on Applied Computing*, New York, NY, USA, Association for Computing Machinery, p. 2185–2191 (online), DOI: 10.1145/2695664.2695908 (2015).

[17] Rose, S., Borchert, O., Mitchell, S. and Connelly, S.: Zero Trust Architecture, NIST SP 800-207 (2019).

[18] Spear, B., Beyer, B. A. E., Cittadini, L. and Saltonstall, M.: Beyond Corp: The Access Proxy, *;login:*, Vol. 41, No. 4, pp. 28–33 (2016).

[19] Traore, I., Woungang, I., Obaidat, M. S., Nakkabi, Y. and Lai, I.: Combining Mouse and Keystroke Dynamics Biometrics for Risk-Based Authentication in Web Environments, *2012 Fourth International Conference on Digital Home*, pp. 138–145 (online), DOI: 10.1109/ICDH.2012.59 (2012).

[20] Tulshibagwale, A.: Re-thinking federated identity with the Continuous Access Evaluation Protocol, https://cloud.google.com/blog/products/identity-security/re-thinking-federated-identity-with-the-continuous-access-evaluation-protocol.

[21] Ward, R. and Beyer, B.: BeyondCorp: A New Approach to Enterprise Security, *;login:*, Vol. Vol. 39, No. 6, pp. 6–11 (2014).

[22] 宮川寧夫，砂原秀樹：代理アクセスを認可する OAuth 仕様の進展，コンピュータソフトウェア， Vol. 29, No. 1 (2012).

[*10] https://meatwiki.nii.ac.jp/confluence/pages/viewpage.action?pageId=12158166