Computer Security Symposium 2021
26 - 29 October 2021

# エラー状態や競合発生を防止するユーザ認可方式の提案と実装

森 拓海[1,*] 藤田 真浩[1] 柴田 陽一[1] 松田 規[1]

**概要**：ファイルサーバ上で企業機密を扱う場合，強固な保護が必要である．最近では，オンプレミスや社内クラウド上の企業機密を，様々な環境（社内/社外，昼/夜，国内/海外，など）からアクセスする．そのため，利用者らしさ（信頼度）に応じて認証やアクセス制御を厳格に行い，情報漏洩リスクを低減するニーズがある．従来の固定的な ACL(アクセス制御リスト)による認可は，アクセス制御対象のファイルの内容変更による認可の見直しを実施できない．そこで，ファイルの属性や内容，アクセス経路等の認可条件を整理してオントロジーを構成し，ファイルアクセス時にオントロジーを用いて動的にアクセスポリシーを生成・適用する方式を開発した．この方式を用いることで，動的アクセスポリシーが矛盾する(エラー状態)場合でも可用性を損ねず，アクセス対象のファイルの内容が書き替わってもアクセスポリシーを追従可能となる．また，OSS の認可サーバである KeyCloak を用いたプロトタイプシステムを開発し，企業におけるリモートアクセスを想定した評価環境でフィージビリティ確認を行った．この評価を通じ，オントロジー入力 I/F や入力されたオントロジーの妥当性検証など，実用化の課題を明らかにした．

**キーワード**：動的認可，オントロジー，ゼロトラスト，企業機密管理

# Proposal and Implementation of User Authorization Method Preventing Error State and Confliction

Takumi Mori[1,*] Masahiro Fujita[1] Yoichi Shibata[1] Nori Matsuda[1]

***Abstract***: Strong data protection is required for handling confidential data on file servers. Recently, it has increased to access confidential data stored into on-premises servers or private cloud from various environments such as inside/outside a company, day/night, domestic/overseas. Therefore, reducing the risks of information leakage by stronger authentication and stricter access control using user reliability is important. A conventional fixed ACL (access control list) authorization cannot change access rules dynamically with file contents changes. We developed a method that integrates authorization conditions such as file attributes, contents, and access routes to configure an ontology, and dynamically generates and applies access policies using the ontology when accessing a file. Our method realizes that availability is not impaired even if the dynamic access policy is inconsistent (error state), and the access policy can be followed even if the contents of files are modified. In addition, we developed a prototype system on KeyCloak (an OSS authorization server) and verified the feasibility in an evaluation environment assuming remote access in a company. Through implementation and evaluation, we clarified the issues for practical use such as the ontology input I/F and the validation of the input ontology.

***Keywords***: Dynamic authorization, Ontology, Zero trust, Confidential data management

## 1. はじめに

企業のリモートワークやクラウド活用の拡大に伴い，機密情報を扱う IT システムに求められるセキュリティ要件は厳しくなっている．従来は社内において厳重に保護されていた企業機密は，社内/社外，昼/夜，国内/海外などからアクセスされ，時間や場所，デバイスの制限なく扱える IT システムが求められるようになった．

このような変化に対し，固定的なアクセスルールによる境界防御の対策では対応することが難しいため,「ゼロトラスト」の概念が登場した[1]．この概念では，信頼されたネットワークをセキュリティ対策によって守るのではなく，様々なセキュリティ機能によって，状況に応じて信頼を獲得する(すなわち，誰も信用しない)モデルである．ゼロトラストは NIST SP800-207[2]で定義化されており，以下 7 つの原則がまとめられている．

(1) 全てのデータソースとコンピューティングサービスをリソースとみなす．
(2) ネットワークの場所に関わらず，全ての通信を保護する．
(3) 企業リソースへのアクセスは，セッション単位で付与する．
(4) リソースへのアクセスは，クライアント ID，アプリケーション/サービス，リクエストする資産の状態，その他の行動属性や環境属性を含めた動的ポリシーにより決定する．
(5) 全ての資産の整合性とセキュリティ動作を監視し，測定する．
(6) 全てのリソースの認証と認可を動的に行い，アクセスが許可される前に厳格に実施する．
(7) 資産，ネットワークインフラストラクチャ，通信の状況について可能な限り多くの情報を収集し，セキュリティ体制の改善に利用する．

1 三菱電機株式会社 情報技術総合研究所
Mitsubishi Electric Corporation, Information Technology R & D Center
* Mori.Takumi@db.MitsubishiElectric.co.jp

© 2021 Information Processing Society of Japan

この原則のうち(1)～(3)はアクセス対象の考え方を示している．(4)(6)は動的な認証・認可，(5)(7)はアクセス要求の信頼性を担保するための入力情報の提供方法を示している．これらの原則は実現するための具体的な技術には言及していないため，本稿では(4)～(7)に着目し，動的な認可として利用できる情報をオントロジーにより整理することによって，実際に認可処理に利用する仕組みを検討する．なお，本稿ではオントロジーを用いた動的認可のスキームの提案と，評価環境におけるフィージビリティ検証までを対象としている．従来方式との認可精度の比較や，実用的な認可ルールにおける評価は今後の研究課題とした．

## 2. 関連研究

動的な認可を実現する上でのアプローチは[3]～[12]のように多くの方式が提案されているが，膨大なデータから機械学習を用いて認可を行う分類器を構成する方法と，従来の ACL(アクセス制御リスト)の概念を拡張したオントロジーを用いる方式の 2 つに分類できる．

機械学習を用いる方法は，適応型ユーザ認証をサーベイした論文[3]にまとめられている．この論文では，適応型ユーザ認証の共通定義を示し，その定義に基づいて既存の研究論文を以下の視点で調査している．

① Authenticator (例：パスワード)
② What to adapt (例：信頼度，認証の要素)
③ Why to adapt (例：場所の変化)
④ How to adapt (例：自宅では認証要素を増やす)
⑤ When and Where to adapt (例:システム/アプリレベル)

これらの視点で，動的な認証・認可を実現している例として,推論エンジンや機械学習を利用しているものがある．Cerberus[4]や CARS-AD [5]は，推論エンジンを用いて，ユーザコンテキストを推定することで，必要な認証・認可を実施している．ConXsense [6]や[7]は，GPS や WiFi などのセンサデータから機械学習によりユーザコンテキストの分類器を構成し，分類結果に応じた認証・認可を行う．ASSO[8]や[9]は，時間や位置情報をサポートベクタマシン(SVM)で学習させることで，未知の状態における分類精度を向上させている．

一方，オントロジーを用いた方式は，オントロジーベースアクセス制御方式(以降，OBAC と表記)と呼ばれる．OBAC を利用した方式である[10][11]はオントロジー形式と親和性の高いデータ構造であるグラフデータベースに対して適用した研究である．また, [12]は本稿の提案方式と目的を同じく，ユーザや環境コンテキストをオントロジー形式で管理するスキームを提案している．しかし，概念の提案のみとなっており，実用性の評価は実施していない．

企業のリモートアクセスのユースケースを 5W1H の観点で要求分析し，[3]で参照される各認可技術を整理した結果，実用性の面では次のような課題があることがわかった．

***課題1. 認証・認可の実施結果に対する説明責任***

機密情報の漏えいのようなインシデントが発生した場合，原因究明のために，どのような認証・認可が実施されていたかを説明可能とすることが，その後の企業の信頼回復に重要な役割を果たす．

機械学習により構成された分類器では，適用するアクセスポリシーの妥当性が評価できず，インシデント発生時の説明責任を果たすことができない．OBAC は，オントロジーに意味があるため，説明責任を果たしやすい．

***課題2. 動的な認可処理に対する計算コストの抑制***

アクセスポリシーは時間経過で変化していくが，認可処理は動的に実施する必要がある．そのため，変化するポリシーに追従するための計算コストを抑制すべきである．

機械学習では，再学習して分類器を再構成する必要があるため，計算コストが高い．OBAC では，不変的な情報からオントロジーを形成すれば再構成は最小限にできる．

***課題3. 認可処理におけるエラー状態への事前対処***

機械学習はエラー状態の事前定義が困難である．エラー状態(適用するポリシーが無い)場合の挙動が未定義となる場合,安全面からアクセスを拒否すると可用性に影響する．

機械学習では，エラー状態であっても分類器により何等かのポリシーが適用されるが，事前に期待するポリシーが適用されるかどうかは不定となる．OBAC では，すべてのエラーパターンを細かく定義することが難しいため，定義に工夫が必要である．

***課題4. 複数条件の競合を考慮したアクセス制御***

ゼロトラストの概念では，複数の種類の認可条件が動的に判断される．認可条件の選択の仕方によっては矛盾する場合もあるため，矛盾を解消する仕組みが必要である．

機械学習では，入力条件を増やすことで対応するが，正確な分類器の構成にはさらに多くの学習データが必要になる．OBAC では，複数の条件を体系的に整理する方法を検討しなければならない．

これらの課題と関連研究の位置づけを表 1 にまとめた．本稿では，課題 1 及び課題 2 を既に解決している OBAC 方式をベースに，既存研究で解決されない課題 3，課題 4 を中心に提案方式の特徴を述べる．

表 1 課題と関連研究の位置づけ

| | 機械学習方式 [4][5][6][7][8][9] | OBAC 方式(従来) [10][11][12] |
|---|---|---|
| **課題 1** | × | ○ |
| **課題 2** | × | ○ |
| **課題 3** | △※ | × |
| **課題 4** | △※ | × |

※エラー状態や競合状態にはならないが，意図した動作になるかは保証できない．

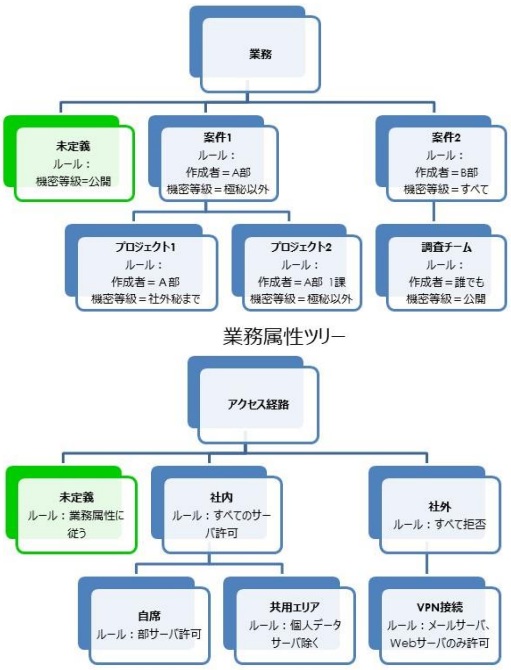



図 1　オントロジーの例

## 3. 提案方式

提案方式(以降，本方式)は，主に 2 章で述べた課題 3 及び課題 4 に対応するために，OBAC で利用するオントロジーにエラー状態時のルール（後述の「未定義ノード」）を追加する．さらに，オントロジーから選択されたルール候補の衝突・競合を解消するための事前定義ルールである「適用ルール」を導入する．

### 3.1 基本動作

本方式は準備フェーズと実施フェーズからなる．準備フェーズでは，アクセスポリシーツリー(オントロジー)を作成する．認可に利用するアクセスルールの属性ごとにツリーを作成することが特徴である(図 1)．このツリーには，各ノードにルールが設定されており，属性ごとに網羅的にルールを定義することができる．

次に，実施フェーズを説明する．実施フェーズでは，ファイルアクセス要求に対してアクセスポリシーを動的に生成し，適用する(図 2)．例えば，アクセス要求としてアクセス対象文書とアクセス経路の情報から，アクセスするユーザの業務がファイルアクセスに適合しているか，またファイルにアクセスするために利用する通信路が安全であるかを判断するため，それぞれ業務ツリーとアクセス経路ツリーを用いて，適用すべきアクセスルールの候補を選定する．選定したルール同士が矛盾する場合，後述する「適用ルール」を参照し，矛盾を解消した上で最終的なアクセスポリシーを決定する．

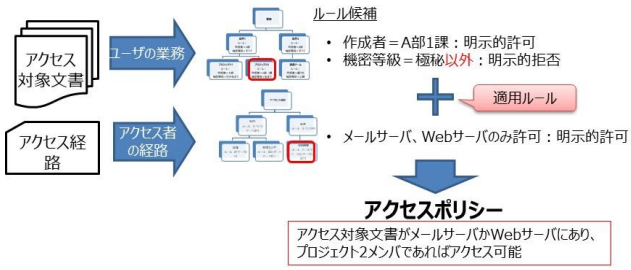



図 2 アクセスポリシーの生成と適用の流れ

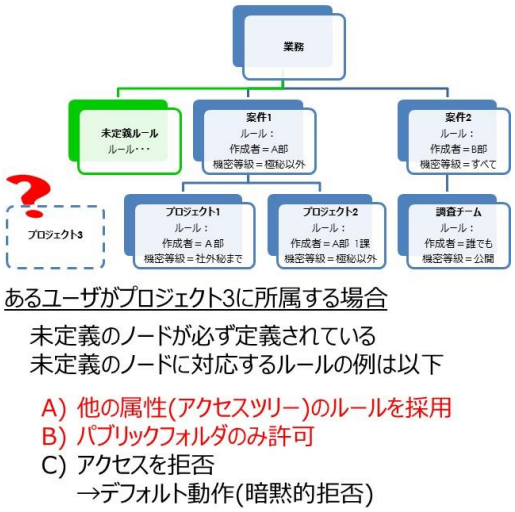



図 3 未定義ノードの例

以降の節では，本方式の特徴である「未定義ノード」と「適用ルール」について説明する．

### 3.2 未定義ノード

アクセスルールに例外が発生した場合に，取り得る手段として以下が想定される．

(1) 別のアクセスツリーのルールを採用する
(2) デフォルトルールを適用する
(3) アクセス拒否する

どのような手段を取るかは，状況により異なる．そこで評価対象の属性ごとに，ルールが存在しない場合のデフォルトルールを定義した「未定義ノード」を用意する．

ツリーで定義されない未知の属性情報を評価する場合は，「未定義ノード」に設定されたルールを参照する．例えば，図 3 のように，業務ツリーの中に，プロジェクト 1，プロジェクト 2 しか定義されていない状態で，プロジェクト 3 の属性を持つユーザがアクセスしてきた場合，適用するノードが無いため，「未定義」ノード(図中緑色のノード)を参照する．未定義ノードには，例えば，「他の属性ツリーのルールを参照」，「パブリックなフォルダのみアクセス許可」，「アクセスを拒否する」などをルールに設定する．

ACL によるデフォルトルールとの違いは，評価対象の属性ごとにルールを決められることである．ACL のデフォル

トルールでは，安全サイドに倒して(3)の「アクセス拒否」を採用することが多い．単一の属性しか評価しない場合は，「拒否」でも問題ないが，アクセス拒否が多発するとデータの可用性が損なわれる．ゼロトラストではユーザを動的に識別し，コンテキストに応じてアクセス可能な範囲を決定するため，デフォルト「拒否」は適さない．OBAC では，認可に用いるコンテキストは複数あるため，ツリーの属性に応じてデフォルトのルールを整理しておくことが効果的である．以上のようにエラー状態の事前定義を漏れなく実施することができ，課題 3 が解決される．

### 3.3 適用ルール

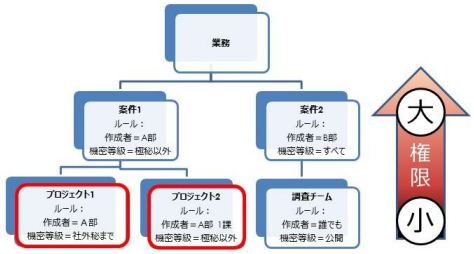
「適用ルール」は，オントロジーから選択される複数のルールが矛盾する状態に陥った場合に，それを解消するためのルールである．複数の属性情報を用いて認可可否を判断する際，いくつかのルール候補を選定した後，最終的なルールを生成して評価する．このとき，候補としたルール同士が衝突する場合があり，取り得る手段は以下がある．

(1) アクセスを拒否する

(2) ルールに優先順位を設ける

これらのどちらの手段を採用するか，また，(2)を採用する場合の詳細動作を予め定義しておくのが「適用ルール」の役割である．

例えば，図 4 のように，あるユーザが業務としてプロジェクト 1，プロジェクト 2 の両方に所属(兼務)していた場合，この例ではプロジェクト 1 のルール「A 部作成で社外秘の等級まで」とプロジェクト 2 のルール「A 部 1 課作成で極秘以外」に対し，「A 部 1 課作成の秘等級」は許可すべきかすべきではないかが判断できない(プロジェクト 1 のルールでは NG で，プロジェクト 2 のルールでは OK)．このような場合に，予め図 4 の A)～D)のようなルールを用意しておく．このルールを用いることで，どのようにルールを適用すべきかが一意に解釈できる．このようにすることで，課題 4 を解決することができる．




あるユーザがプロジェクト1、プロジェクト2の両方に所属する場合

適用ルール例

A) あるノードの全リーフの権限を持つ場合、上位のノードのルールを適用
→「案件1」の権限でアクセス

B) 明示的拒否＞明示的許可＞暗黙的拒否の順で優先、同一順の場合はそれぞれのルールのORとする
→優先順位は同じなので、両方のルールのORでアクセス

C) ユーザに対し、どちらの所属でアクセスするか問い合わせる

D) アクセスを拒否

図 4 ルールが衝突する場合の適用ルールの例

## 4. 評価

本章では，本方式の実用性評価の一環として，企業における機密情報へのリモートアクセスを想定した動的認可システムの試作・評価した結果を示す．試作では，以下 2 点を目的とする．

- 本方式が動作可能かを評価(フィージビリティ評価)
- 実用化にむけた実装方法や課題の抽出

### 4.1 基本設計

本方式を試作するにあたり，開発が必要なコンポーネントと機能要件を検討した．ファイルサーバに対するアクセス認可をユースケースとし，図 5 のように機能要件をまとめた．まず，ユーザ(管理者)は認可サーバに対し，認可条件を登録する．この時，認可サーバは認可に必要な認証方式や，認証要素としてユーザ情報とともにパスワードや認証デバイスなどのクレデンシャルを登録する．次に，ユーザ(利用者)はファイルサーバ(オンプレ/社内クラウド)に対し，データ及び関連するメタデータを登録する．利用者はファイルサーバに保管されたデータにアクセスするために，アクセス要求を作成する．その際，認可に必要なコンテキスト情報を収集する．認可サーバは，利用者のアクセス要求に対し，その要求を分析し，必要に応じて認証を要求する．利用者が認証要求に応答すると，認可サーバは認証サーバに対し，本人・所有物の確認を実施し，その結果を認可サーバに伝える．認可サーバはアクセス要求に含まれるユーザのコンテキスト情報と認証結果から認可可否の判断を行う．ファイルサーバは，認可可否の判断結果としてデータへのアクセスが許可されれば，ユーザにデータを開示する．

図 5 に示した要件を満たす試作システム構成を図 6 に示す．本方式の要であるオントロジー管理プログラムと認証認可判断のロジックを中心に，4.3 節で説明する．

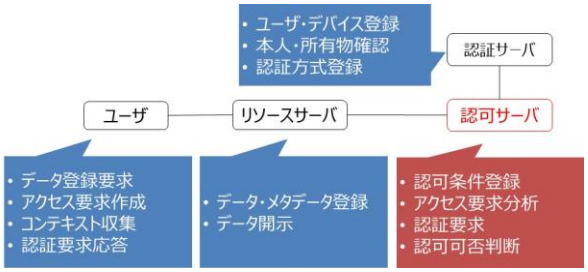



図 5 機能要件

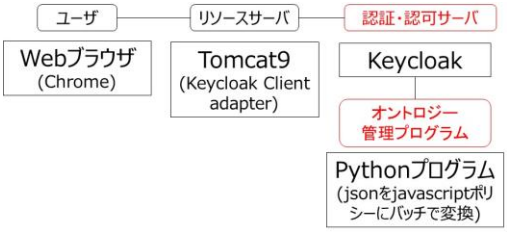



図 6 試作システム構成

### 4.2 開発環境

今回は試作のため，利用 S/W のバージョンは最新ではない．なお，試作には Java の javascript エンジンである Nashorn を利用しているが，この機能は Java11 で非推奨，Java 15 から機能削除となっていることに注意する[a]．

### 4.3 実装手順

本方式を具体的に実装するための手順を示す．本試作では，フィージビリティ評価を目的とするため，本方式を実現する最も簡易的な手段を採用した．一方で，理想的な実装方法についても検討している．KeyCloak に対する実装の流れは以下の通りである．

【手順1.】 各属性のポリシーを設定(ユーザ，ロール，時間，グループ等)．必要に応じて，独自属性用 SPI を追加．

【手順2.】 必要に応じて集約ポリシー(属性ポリシーの組み合わせ)を作成．

【手順3.】 オントロジー用 SPI(試作では javascript ポリシーで代用)を作成し，【手順 1.】【手順 2.】で作成したポリシーをノードとするツリーを作成．

【手順4.】 これをオントロジーポリシーとして Keycloak の内部データベースに保管して管理(試作では javascript ポリシーを Keycloak にデプロイ)．

【手順5.】 アクセス対象のリソースに応じて，パーミッションとしてオントロジーを指定する(集中管理の場合)．また，User Managed Access(UMA)を利用すれば，パーミッション管理を分散型かつ動的に管理可能．

本試作において重要な【手順 3.】【手順 4.】に対応する認可ルールの管理認各手順に関する詳細を述べる．認可に使用する属性は認可ルールとして管理する．表現形式としてツリー状のオントロジーを用いる．

ツリー状のデータ構造は json や XML といった形式があるが，今回は json 形式のデータを採用した．

json 形式の認可条件は，Keycloak に直接読み込ませることはできないため，javascript ベースのポリシーに変換する(図 7)．出力された javascript ベースポリシーを，Keycloak サーバにデプロイし javascript ベースポリシーを Keycloak 管理コンソールの認証フローで有効化する．

表 2 開発環境

| 項目 | 利用 S/W・バージョン |
|---|---|
| 統合開発環境 | IntelliJ IDEA<br>バージョン: 2020.3.2<br>ビルド: 203.7148.57 |
| Java | OpenJDK11 |
| 認可サーバ | Keycloak 12.01<br>(ライセンス：Apache License 2.0) |
| リソースサーバ | Apache Tomcat 9 |

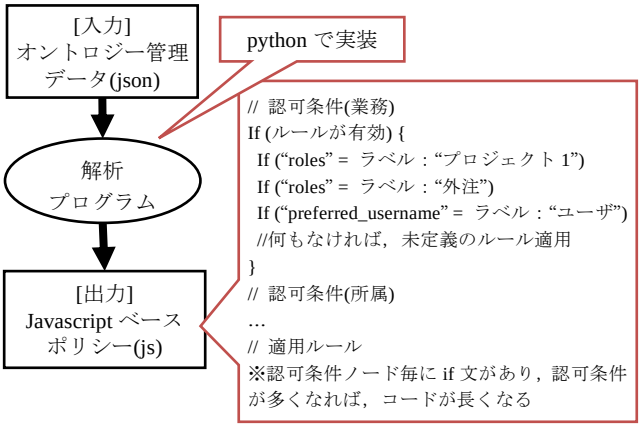



図 7 オントロジー形式の認可ルールの変換方法

表 3 ユーザーとユーザコンテキスト一覧

| ユーザ ID | grade | department | role |
|---|---|---|---|
| au | 課長(アクセスレベル 50) | /SecurityDepartment | pj1 |
| bu | 主任(アクセスレベル 30) | /SecurityDepartment/SecurityOffice1 | pj2 |
| cu | 担当(アクセスレベル 20) | /SecurityDepartment/SecurityOffice2 | pj3 |

なお，今回は試作のため，認可に使用するオントロジー(json ファイル)は 1 つのみとした．また，オントロジーのノード内に設定する条件も単一属性の AND のみである．また，今回は適用ルールよりも，エラー状態の挙動(例外処理)に対応する方が優先度が高いため，本稿ではまず課題 3 に対応できているかを検証した．

本試作では，javascript ベースのポリシーを介すことでオントロジーベースのポリシーを解釈したが，理想的には独自の Authentication SPI として実装すべきである．

### 4.4 試作結果

本節では，実際に試作したシステムに対してオントロジーで表現した認可ルールを適用・実施した結果を示す．

表 3 は，試作システムで使用したユーザ一覧である．それぞれのユーザは，役職を表す grade と所属を表す

a https://openjdk.java.net/jeps/372
Java の最新バージョンは 15 なので，基本的には利用できない機能である．Nashorn は，ECMAScript-262 5.1 の完全実装である Rhino scripting engine に置き換えることができる．今回の試作と同じ方法で実用化を検討する場合，この実装で置き換えることで対応することができる．

department，及び役割を表す role が紐づいている．
また，今回は環境コンテキストとして，ユーザの置かれた状況を示す area という属性をブラウザの拡張機能(Chrome の ModHeader)を使って Http ヘッダに付加した(図 8)．area は level1～level4(level が高いほどセキュアな場所)を指定する想定である．本来はユーザの環境から決定するが，今回は(ブラウザごとに)固定値を設定する形とした．

今回の試作で，実際に動作を確認したオントロジーは図 9 の通りである．各ノードに設定された認可ルールは Positive ルールと解釈するため，いずれかのノードに合致した場合にファイルのアクセスを許可する．

上記オントロジーで利用した属性を図 9 に示した．業務ノードには，ユーザの役職(grade)に紐づく認可ルールを表す secretLevel と，ユーザのアクセス場所の信頼度を表す areaLevel を用いている．所属ノードには，ユーザの所属からノードを特定し，アクセス可能なディレクトリ(directory)を指定する．アクセス経路は，IP アドレスによる場所の特定と，時間帯(time)による認可ルールを扱う．

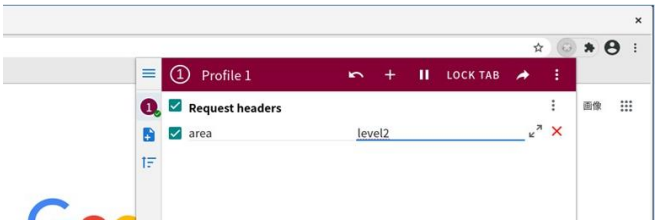



図 8 ModHeader による環境コンテキストの設定 (area=level2)

表 4 オントロジーに使用した属性

| ノード(name) | 属性 | 値の利用方法 |
|---|---|---|
| 業務<br>ユーザの role からノードを決定 | secretLevel | ユーザコンテキストを利用．ユーザの grade に応じたアクセスレベルの閾値を示す．ユーザが secretLevel 以上のアクセスレベルを保持する場合はアクセスを許可する． |
| | areaLevel | 環境コンテキストを利用．ユーザから受信した Http リクエストに含まれる area の値(level1～level4)が areaLevel 以上であれば，アクセスを許可する． |
| 所属<br>ユーザの所属情報からノードを決定 | directory | ユーザコンテキストを利用した．directory に示されたディレクトリのアクセスを許可する． |
| アクセス経路<br>アクセス元 IP アドレスからノードを決定 | time | 環境コンテキストを利用した．時間帯によって，勤務場所(アクセス元)が異なることを想定し，時間帯によって認可可否を決定する． |

オントロジーによる認可ルールをリソースサーバに適用したディレクトリ構造を図 10 に示した．ディレクトリごとに適用する認可ルールを変えている．

以上の設定で動作させた試作システムのデモ画面を図 11 に示した．本試作システムにアクセスすると，ユーザ未認証の場合は Keycloak 認証画面が表示される．表 3 のいずれかのユーザでログインし，認証が成功するとファイル選択画面が表示される．一覧表示されたファイルは前述したオントロジーで表現されたアクセス条件により認可処理が実施される．実際に，試作システムでログインしたユーザのコンテキストによってファイルへのアクセス可否が決定することを確認した．

```
{  "name": "root",
   "children": [
      {    "name": "業務",
           "parent": "root",
           "evaluation": "true",
           "children": [ {
                    "name": "pj1",
                    "parent": "業務",
                    "areaLevel": "level4",
                    "secretLevel": "50",
                    "children": [ {
                             "name": "pj3",
                             "parent": "pj1",
                             "areaLevel": "level2",
                             "secretLevel": "10"
                         },
                          {  "name": "pj2",
                             "parent": "pj1",
                             "areaLevel": "level3",
                             "secretLevel": "15"
                         } ]
                },
                {    "name": "未定義",
                     "parent": "業務",
                     "rule": "拒否"
                } ]
       },
        {  "name": "所属",
           "parent": "root",
           "evaluation": "true",
           "children": [ {
                    "name": "/SecurityDepartment",
                    "parent": "所属",
                    "directory": "ontology/public",
                    "children": [ {
                             "name":
                                "/SecurityDepartment/SecurityOffice1",
                             "parent": "SecurityDepartment",
                             "directory": "ontology/security1"
                         },
                         { "name": "/SecurityDepartment/SecurityOffice2",
                           "parent": "SecurityDepartment",
                           "directory": "ontology/security2"
                         } ]
                },
                 {  "name": "未定義",
                    "parent": "所属",
                    "rule": "拒否"
                } ]
       },
       {  "name": "アクセス経路",
          "parent": "root",
          "evaluation": "true",
          "children": [ {
                    "name": "社外",
                    "parent": "アクセス経路",
                    "time": "10-14"
                },
                {    "name": "社内",
                     "parent": "アクセス経路",
                     "time": "08-20",
                     "children": [ {
                              "name": "自席",
                              "parent": "社内",
                              "time": "08-20"
                         } ]
                },
                {    "name": "未定義",
                     "parent": "アクセス経路",
                     "time": "拒否"
                } ]
       } ]
}
```

図 9 作成した認可ルール(オントロジー)

```
webapps
└── dynauthapp  (試作システムのホームディレクトリ)
      └ ontology
          |
          ├ security1
          |    |
          |    └protect1.txt ( bu が time:08-20 にアクセス可)
          |
          └ security2
               |
               └protect2.txt (cu が time:08-20 にアクセス可)
```

図 10 Webdav サーバのディレクトリ構成とアクセス制限




図 11 試作システムデモ画面

## 5. 考察

試作評価の目的に対する考察をまとめる．まず 1 つ目の目的である本方式のフィージビリティに関して述べる．評価環境では，企業のリモートアクセスを想定し，ユーザの所属や役割，リモート接続元の場所，時間帯といった基本的な要素を用いたアクセス制御がオントロジーにより記載可能であり，動的認可処理に適用することが可能であることがわかった．少なくとも，評価環境においてユーザのアクセス元に関する場所や時間帯といった動的に変化する要素にリアルタイムに追従することができる．既存の動的認可技術や製品でも同様の制御は実現できるものもあるため，提案技術が劣っているということは無いと言える．

次に，2 つ目の目的である実用化に向けた実装方法や課題について述べる．本方式の本試作を通して得られた課題を表 5 にまとめた．

1 つめの課題として，認可に使用する属性の決定がある．認可に必要な属性は，必ずしもシステム開始時に完全に決まっていない．その場合，属性の解析に Keycloak 側の改修が最小限になるよう，予め想定される属性を解析できるロジックを組み込んでおくことが望ましい．

表 5　本方式の実用化に向けた課題

| No. | 実用化課題 | 対応方針 |
|---|---|---|
| 1. | 認可に使用する属性の決定 | 予め認可で使用することが想定される属性を Keycloak 側で解析できるようにしておく． |
| 2. | オントロジーの入力 I/F | 漏れなく，ミスなくツリー状の構造データを入力する I/F を検討する． |
| 3. | オントロジーの正しさの検証 | オントロジー作成者以外の者による確認や，認可ログの確認によって実現する． |
| 4. | 認可ログのオントロジーへのフィードバック | 認可処理を Keycloak に集約し，認可ログからオントロジーの見直しを行う． |
| 5. | オントロジーノードに指定するアクセス条件の表現形式 | 複雑なアクセス条件を表現するには，BNF 記法などを採用することが望ましい． |
| 6. | 適用ルールの実装 | ユーザインタラクションの要否で，実装をポリシーエンフォーサーか，認証サーバ側にするかを決める． |

2 つめの課題は，オントロジーの入力 I/F である．本方式ではオントロジーはツリー形式となっている．試作では json 形式で直接記述しているが，実用化に当たっては，漏れがなく柔軟なルール定義がミスなく記述できる仕組みが不可欠である．

3 つめの課題は，認可ルールとして作成したオントロジーそのものの正しさの検証である．これは本方式に限った話ではないものの，オントロジーの定義やノードに設定される個別のルールの正しさは，何等かの形で検証すべきである．例えば，オントロジー作成者以外の者によるレビュー，認可ログの監査を通じたオントロジーの修正がある．

4 つめの課題は，認可ログのオントロジーへのフィードバックである．本方式は，複数の属性を用いた動的な認可を行うため，個々の認可アクションの妥当性に誤りがないかを確認する必要がある．認可サーバにはログが集約されるため，ログの監査自体は可能であるが，それをオントロジーの修正にフィードバックする仕組みは別途考える必要がある．ベースオントロジーを用いた認可ログからのオントロジー修正の一例は既に検討しているが，より実効性のある方法が必要になると考えられる．

5 つめの課題として，オントロジーノードに指定するアクセス条件の表現方法がある．アクセス条件は論理式の形式で指定できることが望ましい．任意の論理式の評価のための解析エンジンを自作するのは非常に難しいため，BNF 記法などのよく知られた形式で表現することを考える．た

だし，課題 2 のように分かりやすい I/F とする必要もあるため，具体的な作りこみについては今後も継続して検討する必要がある．

最後に 6 つめの課題は，適用ルールの実装である．本試作では，オントロジーノードが同時に 2 つ指定されることや，複数のオントロジーを同時に評価することができない仕様としたため，適用ルールを実装していない．机上検討では，アクセス条件の競合時の解消方法として，ユーザとのインタラクションを必要とする場合は，よりユーザに近いポリシーエンフォーサーによる実装が合理的である．しかし，ユーザとのインタラクションが発生しない場合や認可サーバから直接ユーザにインタラクションを取ることができるプロトコルを利用する場合は，ログ取得の観点からも認証サーバ側で管理するのが良いと考えられる．

## 6. まとめ

本稿では，ユーザのアクセスパターン等をリアルタイムに検証する動的な認証・認可の一つの実現方法として，オントロジーを用いた方式を実装・評価した．

オントロジーを用いることで，認可に複数の種類の情報を用いることが可能となり，動的に複雑なアクセスポリシーを適用させることが可能となった．ユーザの状況に応じて必要最小限のアクセスに制限することで，情報漏洩のリスクを低減させることができる．また，既存の OSS の認可サーバである Keycloak に実装することで，実用化の際に想定される課題を導出した．

本稿では，オントロジーを用いた動的認可のスキームの提案と，評価環境の範囲内でのフィージビリティ検証を実施した．今後は，試作システムに対して実際に想定されるアクセスルールを設定し，有効なパラメータを特定する検討を行う．その上で，適用ルールを用いることが効果的なユースケースや，従来技術との精度比較を行い，実用上耐用し得るかを確認する．

## 参考文献


[1] Kindervag, J.: Build Security Into Your Network's DNA: The Zero Trust Network Architecture, Forrester(online), available from < https://www.virtualstarmedia.com/downloads/Forrester_zero_trust_DNA.pdf>(accessed 2021-6-9).

[2] NIST: SP800-207 Zero Trust Architecture, NIST(online), available from <https://csrc.nist.gov/publications/detail/sp/800-207/finall> (accessed 2021-6-9).

[3] Arias-Cabarcos, P., Krupitzer,C., and Becker, C.:A Survey on Adaptive Authentication, ACM Computing Surveys, Vol.52, Issue 4, Article No.80(September 2019), pp.1-30(2019).

[4] Al-Muhtadi, J., Ranganathan, A., Campbell, R., and Mickunas, M.D.:Cerberus: A context-aware security scheme for smart spaces, In Proceedings of the IEEE International Conference on Pervasive Computing and Communications (PerCom'03), pp.489-496(2003).

[5] Lima, J.C.D., Rocha, C.C, Vieira, M.A, Augustin ,I., and Dantas, M.A.R.:CARS-AD: A context-aware recommender system to decide about implicit or explicit authentication in ubihealth, In Proceedings of the 9th ACM International Symposium onMobilityManagement and Wireless Access (MobiWac'11), ACM, NewYork,NY, pp.83-92(2011).

[6] Miettinen, M., Heuser, S., Kronz, W., Sadeghi, A.R., and Asokan, N.:ConXsense: Automated context classification for context-aware access control, In Proceedings of the ACM ASIA Conference on Computer and Communications Security (ASIACCS'14), pp.293-304(2014).

[7] Kayacik, H.G., Just, M., Baillie, L., Aspinall, D., and Micallef, N.:Data driven authentication:On the effectiveness of user behaviour modelling with mobile device sensors, arXiv preprint arXiv:1410.7743(2014).

[8] Liu, Z., Bonazzi, R., and Pigneur, Y.:Privacy-based adaptive context-aware authentication system for personal mobile devices, J. Mob. Multimed. 12, 1-2 (Apr. 2016), pp.159-180(2016), Retrieved from http://dl.acm.org/citation.cfm?id=3177177.3177187.

[9] Witte, H., Rathgeb, C., and Busch, C.:Context-aware mobile biometric authentication based on support vectormachines, In Proceedings of the 4th IEEE International Conference on Emerging Security Technologies(EST'13), pp.29-32(2013).

[10] Buffa, M., Zucker, F.Z.: Ontology-Based Access Rights Management, Advances in Knowledge Discovery and Management. Studies in Computational Intelligence, vol.398, Springer, pp.49-61(2012).

[11] Brewster, C., Nouwt, B., Raaijmakers, S. and Verhoosel, J.: Ontology-based access control for FAIR data, Data Intelligence2(5), pp.66–77(2020).

[12] Mohammad, A., Kanaan, G., Khdour, T. and Bani-Ahmad, S.: Ontology-Based Access Control Model for Semantic Web, Journal of Information and Computing Science, Vol.6, No.3, pp.177-194(2011).