Um Método para Extração e Refinamento de Políticas de Acesso baseado em Árvore de Decisão e Algoritmo Genético

Bruno Cremonezi[1], **Alex Vieira**[2], **José Nacif**[3], **Edelberto F. Silva**[2], **Michele Nogueira**[1,4]

[1]Depto. de Informática – Universidade Federal do Paraná (UFPR)

[2]Depto. de Ciência da Computação – Universidade Federal de Juiz de Fora (UFJF)

[3]Instituto de Ciências Exatas e Tecnológicas – Universidade Federal de Viçosa (UFV)

[4]Depto. de Ciência da Computação – Universidade Federal de Minas Gerais (UFMG)

cremonezi@ufpr.br, {alex,edelberto.franco}@ufjf.edu.br

jnacif@ufv.br, michele@dcc.ufmg.br

Abstract. *Attribute-Based Access control (ABAC) is becoming more and more popular. However, its adoption comes with a series of challenges that we must face off. Among them, the definition and maintenance of access policies are presented as one of the most complex challenges cause it is a time-demanding and challenging task. In this paper, we propose a method to mining access policies in access logs to simplify the definition and maintenance of access policies. Our proposal utilizes a supervised learning algorithm to create a decision tree that will be used to identify access patterns and build access policies. In this paper, we also propose a refinement method of the generated policies. Our refinement method uses a genetic algorithm to insert controlled distortions on the policy to get better results. Our evaluation uses synthetic and real data. The results show that our method creates valid policies, with an accuracy 10% higher than a similar method presented in the literature.*

Resumo. *O controle de acesso baseado em atributos (ABAC) está cada vez mais popular. No entanto, sua adoção possui uma série de desafios a serem enfrentados. Dentre eles, a definição e manutenção das políticas de acesso se mostra como um dos maiores desafios, pois essas são tarefas complexas. Este artigo propõe um método de mineração de políticas que utiliza registros de acesso para simplificar o processo de criação e manutenção de políticas de acesso no ABAC. A proposta utiliza um algoritmo de aprendizado de máquina supervisionado para gerar uma árvore de decisão, que será utilizada para identificar padrões presentes no registro e gerar uma política de acesso. Esse trabalho também apresenta um método de refinamento das políticas através de um algoritmo genético que insere distorções controladas na política para melhorar os resultados da política. A avaliação foi realizada com dados sintéticos e reais. Os resultados mostram que o método proposto é capaz de gerar políticas corretas, com uma acurácia 10% superior em relação ao estado da arte.*

Os autores agradecem o apoio da UFPR e auxílio financeiro do CNPq e da CAPES, processo número 426701/2018-6, 313844/2020-8 e 432204/2018-0.

1. Introdução

Um importante requisito de qualquer sistema de informação é o de proteger os seus recursos contra acessos não autorizados, enquanto simultaneamente assegura sua disponibilidade para usuários autorizados [Samarati and de Vimercati 2000]. Para que tal requisito seja atendido, é necessária a definição de políticas de acesso (*Access Control Policies* – ACP) a fim de estipular um conjunto de regras, as quais os usuários devem atender, para acessar um determinado recurso [Hu et al. 2013]. A definição de ACPs é uma tarefa crítica para a segurança das organizações e empresas como um todo, uma vez que ACPs errôneas podem resultar em acessos indevidamente aceitos ou rejeitados.

Dentre os diversos modelos de controle de acesso presentes na literatura para se estipular essas ACPs, o modelo baseado em atributos (ABAC) se destaca como um padrão em rápida ascensão. Em 2013, o *National Institute of Standards and Technology* (NIST) reconheceu o ABAC como o principal substituto para os mais populares modelos de controle de acesso até então: o baseado em papéis (RBAC) e o baseado em listas de acesso (ACL) [Hu et al. 2013]. A popularização do ABAC ocorreu pois, neste modelo, as regras que garantem a permissão (ou negação) de um pedido de acesso passaram a ser baseadas nos atributos (características) das entidades envolvidas desse pedido, o que permite a criação de condições de acesso flexíveis e dinâmicas [Abirami and Venkataraman 2019, Cotrini et al. 2018, Ding and Ma 2021]. Ainda que os benefícios em se utilizar o ABAC sejam claros, sua adoção e implantação são marcadas por um conjunto de desafios. A migração de outros mecanismos de controle de acesso para o ABAC, por exemplo, se mostra uma tarefa difícil e complexa, o que demanda uma grande quantidade de tempo e recursos humanos para a determinação das ACPs [Zhu et al. 2014]. Além disso, mesmo após esse alto custo de definição, a complexidade em realizar a manutenção e auditoria dessas ACPs se mostrou como uma tarefa ainda mais complexa e sujeita a erros humanos. Usualmente, as ACPs tendem a se tornar obsoletas devido a mudanças organizacionais e operacionais. Logo, para que essa ACP seja capaz de proteger de maneira eficaz o acesso aos recursos das empresas, é necessária uma constante atualização e revisão de suas regras. Destaca-se que as ACPs do ABAC tendem a se tornar cada vez mais complexas e difíceis de se manusear, visto que suas regras podem se alterar ao longo do tempo e novas condições de acesso podem ser adicionadas [Hu et al. 2013, Qiu et al. 2020].

Em resposta a esses desafios, uma forma de se evitar a especificação e manutenção manual de ACPs é a mineração de políticas a partir de registros de acesso [Xu and Stoller 2014]. Tendo em vista que esses registros apresentam o comportamento de usuários e dispositivos que acessam um determinado recurso e registros das atuais ACPs utilizadas na prática, a mineração de políticas utiliza técnicas de aprendizado de máquina para identificar e refatorar ACPs complexas, além de corrigir ACPs excessivamente permissivas que possibilitam o acesso de usuários e dispositivos de forma errônea [Cotrini et al. 2018]. Assim, os métodos de mineração de ACPs possuem o potencial para reduzir o custo e a complexidade no processo de especificação e manutenção de ACPs [Xu and Stoller 2014].

Atualmente, é possível encontrar uma extensa lista de trabalhos que oferecem métodos de mineração de políticas. Algumas técnicas que criam ACPs muito especializadas para uma determinada função, tornando-as inúteis para serem utilizadas em outras situações de acesso [Cotrini et al. 2018]. Em outros métodos, minera-se ACPs somente quando o registro possui uma grande quantidade de situações de acesso, pois, caso con-

trário, o método apresentará ACPs regras muito específicas e incapazes de generalizar situações [Xu and Stoller 2014]. Por fim, em outro extremo, há técnicas excessivamente permissivas, o que pode gerar permissões de acessos errôneos e sem qualquer evidência no registro que coloca em risco a segurança do sistema [Bui and Stoller 2020].

Este trabalho apresenta um método de mineração de políticas que utiliza registros de acesso e aprendizado de máquina supervisionado para extrair ACPs precisas e sucintas. O principal objetivo é oferecer um método que seja capaz de extrair diversas ACPs presentes em um registro de entrada e que, ao contrário da literatura, possua mecanismos efetivos para remover ACPs exageradamente complexas ou ACPs super permissivas. O método apresentado neste artigo gera um conjunto de ACPs candidatas a partir de árvores de decisão. Em seguida, utiliza uma métrica de qualidade para avaliar a corretude das ACPs criadas. Essa métrica atribui uma pontuação para cada regra e aplica uma penalidade caso a regra seja altamente permissiva ou tão especializada a ponto de ser uma ACP inútil para uma grande quantidade de acessos. Logo, o método seleciona as melhores ACPs e aplica um cruzamento das soluções através de uma heurística genética com o intuito de encontrar a melhor solução possível para aquele registro.

O presente trabalho avalia a qualidade das ACPs geradas por nossa proposta de mineração de políticas através de uma comparação direta com um método tradicional apresentado na literatura. Verificamos a qualidade das políticas geradas por ambos métodos em relação a sua complexidade e as métricas tradicionais, como acurácia, precisão e revocação. A avaliação dos métodos ocorreu através de dois grandes registros, sendo um deles real e o outro sintético, construído conforme as instruções de outro trabalho da literatura. Foi verificado em nossos resultados que nossa proposta apresentou um melhor desempenho em relação todas métricas tradicionais, em ambos registros. Em alguns casos, nossa proposta apresentou uma acurácia 10% superior ao método comparado.

2. Fundamentos

Nesta seção, inicialmente, descrevemos os fundamentos necessários para a compreensão deste artigo. Inicialmente, descrevemos conceitos básicos sobre o funcionamento do ABAC e os possíveis riscos inerentes de uma autorização de acesso incorreta. Em seguida, descrevemos o problema de mineração de políticas. Por fim, apresentamos uma visão geral sobre algoritmos genéticos e como podem ser empregados em nosso contexto.

2.1. Controle de Acesso ABAC

O ABAC é uma estratégia de autorização que define permissões com base em atributos dos usuários e recursos. Uma ACP no ABAC é um conjunto de regras nas quais, para cada regra, existe um conjunto de condições baseadas em valores de atributos que permitirá ou não um determinado acesso [Hu et al. 2013]. Para que um acesso ocorra, é necessário que haja uma requisição de acesso que demonstre que um usuário deseja realizar uma ação sobre um determinado recurso. Para que essa requisição seja aceita, o usuário deve realizar um pedido de acesso e se identificar no sistema. Após a identificação do usuário, os valores de seus atributos e os do recurso são analisados em cima de uma ACP. Caso os atributos dos usuários e recursos satisfaçam todas as condições de pelo menos uma regra da ACP, o acesso é permitido, caso contrário, o acesso é negado [Hu et al. 2013].

Neste trabalho, define-se que todas ACPs devem ser auditoradas e, devido a esse fator, ACPs extremamente grandes e complexas são indesejadas. Além disso, devido a

mudanças organizacionais ou até mesmo erros humanos, a implementação e manutenção dessas ACPs podem ser imprecisas. Essas imprecisões podem resultar em dois tipos de erros: autorizações incorretas, em que um usuário que não deveria possuir acesso conseguiu satisfazer ao menos uma regra da ACP; e negações incorretas, em que um usuário que deveria possuir acesso não consegue satisfazer nenhuma regra da ACP implementada. Normalmente, uma autorização incorreta (usualmente definida como precisão da autorização) é a mais problemática, pois pode ser usada como porta de entrada para diversos ataques de segurança, como invasões de privacidade e escalada de privilégios. Uma negação incorreta (usualmente definida como revocação da autorização), usualmente não representa um risco tão grande para segurança, uma vez que o usuário pode simplesmente reportar seu erro para o administrador do recurso, que por sua vez pode corrigir ou adicionar uma exceção para a ACP de acesso. Porém, ainda que não possua um desfecho tão problemático, essa manutenção de política e criações de exceções podem levar a mais erros humanos e ACPs complexas com baixa auditoria [Cotrini et al. 2018].

2.2. Mineração de Políticas

Um registro de acesso contém informações de todas requisições de acesso feitas no sistema por um determinado período de tempo. Essas informações podem ser processadas para gerar estatísticas de acesso e caracterizar como eles ocorrem. Usualmente, cada linha deste registro é formada pela junção de uma decisão de acesso (acesso válido e inválido) com todos os atributos que influenciaram ou não aquela decisão. Sendo assim, a mineração de políticas consiste em determinar uma ACP para um registro de acesso, de forma que, para cada linha presente naquele registro, a ACP seja capaz de fornecer uma decisão de acesso correta. Logo, é necessário identificar e definir um conjunto de valores de atributos que influenciam uma decisão de acesso. Assim espera-se que, para todo registro, a ACP resulte em acessos corretos. Como esperado, a elaboração de tais ACPs com todas as possíveis combinações de atributos presentes no registro de acesso pode ser uma tarefa complexa. De acordo com [Xu and Stoller 2014], esse problema é da classe NP-difícil, sendo uma variação do problema *Edge RMP* [Lu et al. 2008].

2.3. Algoritmos Genéticos

Os algoritmos genéticos são uma classe particular de algoritmos que se baseiam nos princípios Darwinianos para resolver um conjunto de problemas [Kramer 2017]. De maneira geral, o primeiro passo dessa classe de algoritmos é gerar uma população inicial de cromossomos, sendo cada cromossomo uma possível solução para o problema a ser resolvido. Esses cromossomos são compostos por um conjunto de genes, que são características pontuais daquela solução. Cada gene é determinado por um alelo, que representa um valor que um gene pode assumir. Durante o processo evolutivo, cada cromossomo recebe uma nota de aptidão, que reflete a qualidade da solução. Os cromossomos mais aptos devem ser selecionados e gerar soluções descendentes, enquanto os menos aptos devem ser descartados. Logo, para se obter essas soluções descendentes mais aptas a solucionar o problema, operações de cruzamento e mutação são empregadas. De maneira geral, a operação de cruzamento seleciona dois ou mais cromossomos e os combina para gerar um cromossomo descendente. A operação de mutação, por sua vez, ocorre quando um gene particular de um cromossomo é submetido a uma mudança de alelo. É importante ressaltar que as soluções geradas pelos algoritmos genéticos não garantem a obtenção de

resultados ótimos. Porém, problemas que envolvem grandes espaços de busca são particularmente adequados para os algoritmos genéticos, uma vez que estes são capazes de obter soluções adequadas em um tempo de busca viável [Thengade and Dondal 2012].

3. Trabalhos Relacionados

Os métodos de mineração de ACPs específicos para o ABAC foram inicialmente apresentados por [Xu and Stoller 2014]. Nele, os autores propuseram um método de mineração de ACPs que possui como entrada, um conjunto de registros de acesso. Esses registros são constituídos por um conjunto de identificadores de usuários, um conjunto de identificadores dos recursos acessados, as operações que foram realizadas em cada um daqueles recursos, quais usuários acessaram quais recursos e um conjunto de atributos associados de cada usuário. Com base nesses registros, os autores geram uma ACL, que é composta por diversas tuplas de usuário-permissão. De maneira resumida, cada tupla dessa lista indica que um identificador de usuário possui o direito de realizar uma operação específica sobre um determinado recurso. Para expandir essas regras, os autores oferecem um método que une diversas tuplas da ACL com o intuito de criar ACPs candidatas que sejam capazes de generalizar diversas situações de acesso e cobrir um maior número de usuários. Assim, o método proposto pega cada regra gerada e substitui o identificador do usuário por seus atributos. Após definir regras em relação aos atributos, o método proposto inicia um processo de união das regras. Nesse processo, regras redundantes são excluídas e regras únicas são unidas para formar uma ACP. Ainda que efetivo no processo de mineração de políticas, a abordagem apresentada é altamente baseada em uma heurística e gera regras altamente complicadas de se interpretar. Os autores afirmam que seu método possui uma complexidade algorítmica cúbica em relação ao número de usuários únicos, o que impede sua execução em registro com uma grande quantidade de usuários.

O trabalho de [Medvet et al. 2015] foi inspirado no trabalho de [Xu and Stoller 2014]. Nele, os autores propuseram um método evolutivo para lidar com o problema de geração de ACPs no mecanismo de controle de acesso ABAC. Nesse trabalho foi utilizada uma abordagem de dividir e conquistar em que a ACP é construída de maneira incremental. Em um primeiro momento, os autores seguiram de forma bem similar o método de [Xu and Stoller 2014], pois realizaram um processo de transformação de um registro de acesso em uma ACL trivial. Após a construção dessa ACL, o trabalho entra em um processo de repetição em que, em cada iteração, um conjunto de regras aleatórias com base nos atributos dos usuários e recursos são criadas. Para cada regra, avalia-se o número de regras triviais que ela cobre. Com todas regras avaliadas, o método seleciona a que cobre o maior número de casos e retira as regras triviais cobertas da ACL. Repetido esse processo inúmeras vezes, o método finaliza no momento em que ACL está completamente vazia e as regras são unidas para formar uma ACP. Assim como [Xu and Stoller 2014], este trabalho usa dados sintéticos para sua avaliação que, de acordo com os autores, não é o ideal para o contexto de mineração de políticas.

O método apresentado por [Iyer and Masoumzadeh 2018] se difere por ser capaz de minerar não só as ACPs tradicionais que permitem o acesso, mas também ACPs negativas de autorização, capazes de negar o acesso de usuários aos recursos de uma organização ou empresa. Para minerar esses dois tipos de ACPs, os autores optaram por utilizar e estender um algoritmo de mineração de dados chamado PRISM. O algoritmo PRISM é um algoritmo iterativo que possui uma abordagem separar-e-conquistar para a

obtenção de ACPs corretas. Ainda que esse trabalho apresente regras "corretas"tanto para as ACPs positivas quanto para as negativas, ele tende a gerar ACPs especializadas (*overfitting*) para o registro de entrada, o que gera regras complexas e incapazes de autorizar de forma correta acessos diferentes dos apresentados no registro de entrada.

Em [Cotrini et al. 2018], os autores propuseram um algoritmo chamado *Rhapsody* para mineração de ACPs em Logs de acesso. Nesse trabalho, todo o funcionamento do algoritmo é construído sobre fundamentos de conjuntos. Nele, os autores definem uma métrica chamada confiabilidade que define a aptidão de uma regra gerada. Em resumo, essa métrica mede o quão excessivamente permissiva uma regra extraída é, e avalia subconjuntos dentro desta regra. Ainda que apresente resultados satisfatórios em relação a precisão das ACPs geradas, quando se há um grande número de atributos e valores de atributos, a complexidade computacional para se extrair uma ACP cresce em exponencial.

4. Solução proposta

A Figura 1 ilustra o mecanismo de mineração de ACPs proposto neste trabalho. Esse mecanismo está dividido em 4 etapas: pré-processamento, extração, avaliação e refinamento.

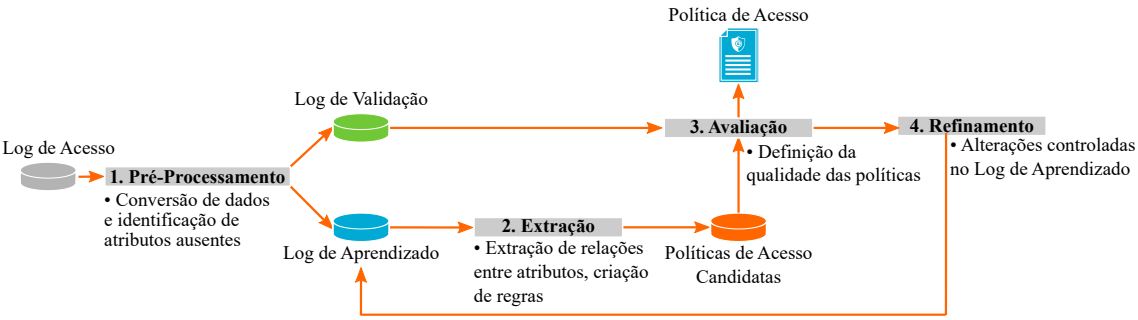

Figura 1. Visão geral do método de extração e refinamento de políticas.

4.1. Pré-Processamento

O pré-processamento é um conjunto de sub-etapas que envolvem preparação, organização e estruturação do registro de acesso. Esta etapa precede a extração de política e está dividida em 5 sub-etapas: Entrada, Categorização, Complementação, Transformação e Divisão. Uma visão geral e exemplo do pre-processamento pode ser visto na Figura 2. É importante notar que, uma vez que a sub-etapa “entrada” possui grande simplicidade, sendo constituída apenas da leitura dos registros de acesso, seus detalhes serão ocultados na explicação das sub-etapas de pré-processamento.

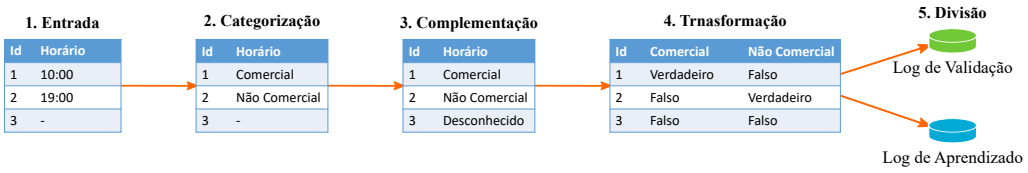

Figura 2. Exemplo das etapas do pré-processamento.

Categorização: Neste trabalho, todas as regras devem ser criadas tendo como base dados categóricos. Assim, o primeiro passo é pré-processar o registro de acesso

de forma a converter todas as variáveis numéricas para variáveis categóricas. Como o ABAC se apropria de atributos dinâmicos como localização e horário. O nosso pré-processamento se inicia na transformação desses atributos de variáveis contínuas para variáveis discretas. Um determinado acesso pode possuir o atributo horário que define a exata hora em que o acesso ocorreu, neste caso, acessos ocorridos entre 09:00 e 18:00 são transformados para uma variável discreta como "comercial", enquanto acessos fora deste horário estão definidos como "Não comercial", como exemplificado na Figura 2 (2).

Complementação: Usualmente, registros de acesso podem apresentar informações faltantes. Logo, como exemplificado na Figura 2 (3), caso o registro apresente ausência de informação em alguma linha, o valor faltante é substituído por "Desconhecido" e esse valor será desconsiderado na execução.

Transformação: Após realizar a categorização e substituição dos valores faltantes, o pré-processamento realiza uma operação de transformação dos dados chamada *one-hot-encoding*. De maneira geral, essa transformação cria uma nova coluna com uma pergunta binária que indica a presença de cada possível valor dos atributos. Por exemplo, na Figura 2 (4), cada categoria do atributo horário foi atribuído em uma nova coluna. Sendo assim, criou-se a coluna "Comercial" e a coluna "Não Comercial". Note-se que caso o valor do atributo seja comercial, a coluna comercial é preenchida com o valor "verdadeiro" e a coluna não comercial é preenchida com o valor "falso". Caso o valor seja não comercial, nota-se que ocorre o oposto. É importante ressaltar que ao realizar essa transformação, é possível criar "perguntas" atômicas binárias, para cada coluna, as quais transmitem informações sobre o acesso. Por exemplo, na linha com identificador (id) 1 da Figura 2, podemos definir duas perguntas: "O horário de acesso é comercial?", "O horário de acesso é não comercial?". Por meio dessas perguntas, a próxima etapa do método de mineração de políticas (extração) gerará as ACPs.

Divisão: A etapa da divisão é a etapa final do pré-processamento e consiste no particionamento do registro de acesso de treinamento em dois outros registros: registro de validação, que contém um subconjunto do registro de acesso que será utilizado para a avaliação da ACP, e o registro base de aprendizado, que contém o restante do registro de acesso e que será utilizado como base para extrair as ACPs.

4.2. Extração

A etapa de extração de políticas recebe um registro de aprendizado pré-processado e entrega uma ACP condizente com tais dados. Para realizar essa tarefa, a extração foi dividida em outras 2 sub-etapas: inferência e interpretação. Neste trabalho, a etapa de inferência se utiliza das árvores de decisão para extrair regras de classificação dos dados, e a interpretação realiza a transformação dessa árvore para uma ACP.

Inferência: Ao receber o registro de acesso pré-processado, as informações contidas nele estão na sua forma atômica, ou seja, definidas em uma função binária que estabelece ou não a presença de um determinado atributo em uma linha do registro. Para gerar as ACPs positivas e negativas, optou-se por utilizar as árvores de decisão para classificar os acessos presentes no registro. Optou-se por esse modelo pois as árvores de decisão são bastantes estudadas na área de aprendizado de máquina e utilizadas para problemas de classificação [Umadevi and Marseline 2017]. Logo, deseja-se criar um modelo de treinamento que pode ser utilizado para determinar se um acesso deve ser permitido.

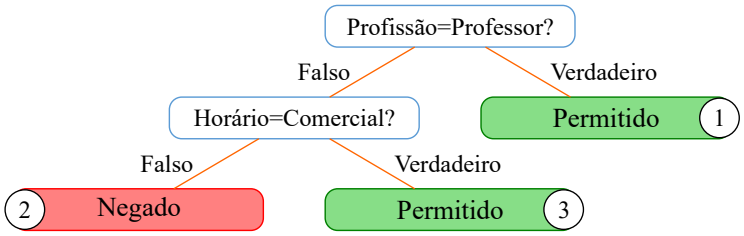

Figura 3. Exemplo de uma árvore de decisão gerada na sub-etapa de inferência.

Interpretação: Uma vez que a árvore de decisão está construída, o processo de interpretação gera uma ACP ABAC através da união dos caminhos até as folhas de uma determinada classe. Por exemplo, na Figura 3, observa-se que as folhas 1 e 3 são da classe "Permitido" e a folha 2 é da classe "Negado". O processo de interpretação, seleciona cada folha e gera o caminho até a raiz selecionando o nó superior e adicionando o valor do ramo. Quando o caminho até a raiz tem caminho maior que 1, expressão lógica "AND" é adicionada para cada novo salto até a raiz. Novamente utilizando a Figura 3 de exemplo, a folha 1 possuí distância 1 da raiz e gera uma regra simples: "Se (Profissão=Professor)==Verdadeiro então Acesso permitido"; a folha 2, possui distância 2 e gera a seguinte regra "Se ((Profissão=Professor)==Falso AND (Horário=Comercial)==Falso) então Acesso negado"; por fim, a folha 3, que possui distância 2, gera a seguinte regra "Se((Profissão=Professor)==Falso AND (Horário=Comercial)==Verdadeiro) então Acesso permitido". Logo, ao utilizar a expressão lógica "OR" para realizar a união das folha de uma mesma classe, temos que a política que permite o acesso é definida pela "folha 1 OR folha 3" e a política que nega o acesso é definida pela "folha 2". Logo, a política resultante que permite o acesso é definida por "Se ((Profissão=Professor)==Verdadeiro) OR ((Profissão=Professor)==Falso AND (Horário=Comercial)==Verdadeiro) então Acesso permitido". Enquanto a política resultante que nega o acesso é definida por "Se ((Profissão=Professor)==Falso) AND (Horário=Comercial)==Falso) então Acesso negado".

4.3. Avaliação

A etapa de avaliação se utiliza do registro de validação para testar a ACP gerada na etapa anterior. Para cada ACP de acesso presente nos conjuntos de ACPs candidatas, a avaliação testa todo registro de validação na ACP e define um valor de aptidão com base no tamanho da ACP, precisão e revocação. Essa aptidão possui como objetivo remover ACPs exageradamente complexas ou ACPs super permissivas. Como apresentado na sub-seção 3.1, define-se que o objetivo deste trabalho é apresentar regras de baixa complexidade e revocação, e essas regras devem possuir alta precisão. Assim, como este trabalho possui 3 objetivos diferentes, foi definido uma soma ponderada entre esses objetivos com o intuito de transformá-los em objetivo único. A métrica que definimos para unir esses objetivos é chamada de aptidão e está definida como: $Aptidão = A.(Precisão)^x + B.(Revogação)^y$

Essa métrica de aptidão varia entre os valores 0 e 1, sendo 0 ACPs completamente inaptas e 1 ACPs "perfeitas" para o registro de validação. Como a precisão é um fator de maior criticidade, assim é estipulado que A>B e x>y, ou seja, nessa função a precisão possui um peso maior na revocação, porém a punição ao permitir um acesso inválido é maior do que negar um acesso válido. Caso duas ACPs possuam o mesma aptidão, o tamanho da ACP, definido pela quantidade (ANDs e ORs)+1, é definido como desempate.

4.4. Refinamento

A ACP gerada é altamente dependente do registro de aprendizado. Como esse registro de aprendizado contém apenas um conjunto de possibilidades de acesso, a ACP gerada por ele pode não ser a mais acurada. Assim, para remediar tal problema, a etapa de refinamento possui como objetivo injetar distorções intencionais no registro de aprendizado, com o intuito de se obter novas ACPs candidatas. Visto que o espaço de busca para as novas ACPs candidatas é grande, ao invés de realizar essas injeções de forma aleatória, optamos por realizá-las conforme uma heurística genética.

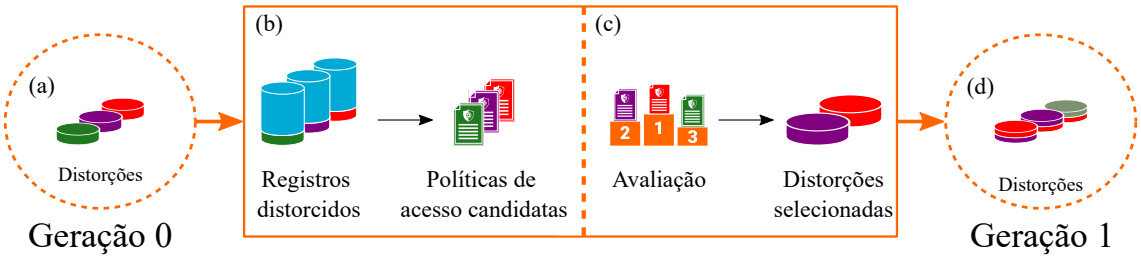

Figura 4. Exemplo das etapas do refinamento.

Neste trabalho, a fase de refinamento se apoia em um algoritmo genético. Em sua primeira etapa é gerado um conjunto de acessos aleatórios (i), que serão agrupados para gerar um conjunto inicial de distorções (ii). Esse conjunto de distorções é chamada de geração 0 e cada elemento desse conjunto será injetado no registro base de aprendizado para se obter um registro distorcido. Por exemplo, na Figura 4 apresenta-se uma representação da fase de refinamento, observa-se 3 elementos na geração 0 (Figura 4a), de modo que cada um desses elementos será introduzido no registro base de aprendizado e gerará um novo registro de aprendizado distorcido. É importante notar que esse registro distorcido é majoritariamente composto pelo registro de aprendizado disponibilizado na etapa de pré processamento, apenas unido com um conjunto de entradas distorcidas. Neste trabalho, estipulou-se que as distorções possuem 20% do tamanho de entradas do registro de aprendizado. Para cada registro distorcido, o processo de extração é executado, o que resulta em uma nova ACP (Figura 4b). Essa ACP então é avaliada, de forma que as distorções que geraram as ACPs com os melhores valores de aptidão são selecionadas (Figura 4c) para gerar as distorções da próxima geração. Assim, com um conjunto de distorções que geraram melhores ACPs, aplica-se técnicas de cruzamento (iii), mutação (iv) e recombinação (v) para gerar uma nova geração de distorções, com base na geração anterior (Figura 4d). Esse processo se repete até um determinado número de gerações, ou quando as ACPs conseguem atingir um nível aceitável na etapa de avaliação. A seguir, apresentamos detalhes sobre a geração de acessos aleatórios, geração do conjunto inicial de distorções, cruzamento, mutação e recombinação.

i **Geração de Acessos Aleatórios:** Dado que o registro é composto por um conjunto de atributos e decisões de acesso, um acesso aleatório analisa os possíveis valores de cada atributo e gera uma decisão de acesso aleatória. Por exemplo, suponha um registro composto por acessos que possuem 2 atributos A e B. Suponha que o atributo A pode assumir os valores $\{a1, a2\}$, atributo B pode assumir os valores $\{b1, b2\}$ e a decisão de acesso assumir os valores $\{Negado, Permitido\}$. Um acesso aleatório é gerado ao selecionar um elemento aleatório de A, um elemento aleatório de B e uma

decisão de acesso. Por exemplo, um conjunto de acessos aleatórios pode ser definido por $\{a1, b1, Negado\}$,$\{a1, b2, Permitido\}$,$\{a2, b1, Negado\}$,$\{a2, b2, Permitido\}$.

ii **Conjunto Inicial de Distorções:** Neste trabalho, cada distorção representa um cromossomo. Esse cromossomo é modelado por meio de um vetor de tamanho N, em que $N > 1$, de forma que cada elemento desse vetor representa um acesso. O alelo de cada gene é representado por cada combinação de atributos e decisão de acesso possível de ocorrer no registro. Sendo assim, o conjunto inicial de distorções gera um conjunto de acessos aleatórios e os agrupa em diversos cromossomos.

iii **Cruzamento:** O cruzamento ocorre quando dois cromossomos são selecionados e combinados entre si. Para que isso ocorra, dado dois cromossomos A e B de tamanho N, um número aleatório i é selecionado, para $i \leq N$. Após definido o valor de i, o cruzamento une os elementos de 0 até i do vetor A com as posições $i + 1$ até N do vetor B e une os elementos de 0 até i do vetor B com as posições $i + 1$ até N do vetor A. Como resultado, dois novos vetores são gerados.

iv **Mutação:** Cada cromossomo possui um conjunto de genes, sendo cada gene uma decisão de acesso e um conjunto de atributos relacionados. Logo, a mutação seleciona um ou mais acessos de um cromossomo e substitui por outros gerados aleatoriamente.

v **Recombinação**: Cada cromossomo é modelado em um vetor de tamanho N. O processo de recombinação seleciona um um número aleatório i, para $i \leq N$, e inverte a ordem do vetor. Ou seja, após definido o valor de i, os elementos entre $i + 1$ e N, passam a ser os de posição 0 e i, e vice-versa.

5. Experimentos

Neste trabalho, vamos comparar o método proposto (A.Dec+Gen) com uma abordagem que utiliza árvore de decisão tradicional (A.Dec), similar ao trabalho [Bui and Stoller 2020], porém aplicada para o ABAC. Os experimentos realizados para avaliar o desempenho de ambos métodos foram realizados através de simulações computacionais. Os métodos foram desenvolvidos em *Python*, utilizando a biblioteca de aprendizado de máquina *sklearn* [Pedregosa et al. 2011]. Os métodos foram testados em 2 bases de dados diferentes. A primeira base é constituída de dados sintéticos, sendo construídos conforme as especificações de [Talukdar et al. 2017]. A segunda base é constituída de dados reais da *Amazon*, disponibilizados na plataforma *Kaggle* [Amazon 2013]. Em cada base de dados, optamos por uma divisão de 80% de dados para treino e 20% de dados para teste. É importante notar que tanto a base de dados sintética quanto a base de dados da *Amazon* são desbalanceadas. Foram utilizadas três técnicas para lidar com desbalanceamento: nula (N), *oversampling* (O) e *undersampling* (U). A nula (N) consiste em executar ambos os métodos com os dados desbalanceados e ignorar o problema de desbalanceamento. O *oversampling* (O) consiste em replicar amostras aleatórias da classe minoritária, enquanto o *undersampling* (U) consiste em reduzir de forma aleatória os exemplos da classe majoritária. Neste trabalho, o nosso método foi avaliado em máquinas de CPUs com 4 núcleos de 2.9 GHz e 8 GB de RAM. Em relação à árvore de decisão utilizada em nosso método, limita-se sua altura em 6, para evitar árvores que gerem regras super especializadas. Na parte evolutiva do método, o número máximo de gerações foi especificado em 200, sendo definido como o critério de parada o limite de gerações ou aptidão igual a 1.

5.1. Bases de Dados

A base de dados sintética foi dividida em 4 instâncias, sendo cada instância construída conforme as especificações de [Talukdar et al. 2017]. Nessa base, deseja-se simular requisições de acesso que ocorreram de um conjunto de usuários sobre um determinado conjunto de objetos. Sendo assim, para a criação dessa base, gera-se 1000 usuários e 100 objetos. Cada usuário e objeto possui 5 atributos únicos de forma que, para cada atributo, existem 5 valores diferentes. Quando uma requisição de acesso ocorre, essa requisição une todos os valores de atributos do usuário e todos os valores de atributos do objeto. Para determinar se esse acesso foi válido, foram geradas 30 regras aleatórias que contêm uma combinação de 5 valores de atributos (tanto do usuário quanto do objeto), que estipulam se o acesso é valido ou não. Uma vez estabelecidas as regras, cada usuário realiza uma requisição de acesso sobre cada objeto e, caso alguma regra seja satisfeita, o acesso é permitido, caso contrário, o acesso é negado. Para a construção dessa base, todas as n amostras de acesso permitido foram selecionadas, enquanto somente um valor de n/5 amostras de acesso negado foram selecionadas. Ao total, cerca de 1800 acessos foram selecionados para cada instância da base de dados sintética.

A base de dados disponibilizada pela *Amazon* é constituída de informações reais de dados coletados entre 2010 e 2011. Nela existem diversos acessos de usuários aos diferentes recursos da empresa, sendo todos esses acessos manualmente definidos por um administrador de rede. Ao todo, essa base de dados possui 32 mil acessos de 12 mil usuários únicos em 7 mil recursos diferentes. Nessa base de dados, cada acesso possui 9 atributos relacionados com os usuários e recursos. Esta base se destaca por ser uma base com grande desbalanceamento, possuindo cerca de 93% dos acessos permitidos. Para esta base de dados, a metodologia de separação das instâncias apresentada por [Cotrini et al. 2018] foi seguida, de modo que todos os acessos que ocorreram sobre cada recurso foram agrupados, o que permitiu a criação de 7 mil instâncias de entrada, uma para cada recurso. Neste trabalho, selecionamos as mesmas instâncias utilizadas no trabalho de referência.

5.2. Métricas de Avaliação

Para a avaliação de desempenho dos métodos, cada um foi mensurado, conforme as seguintes métricas, comumente utilizadas nesse contexto:

i **Acurácia**: Fração de acessos corretamente classificados pela ACP em relação a todos os acessos (tanto acessos permitidos quanto negados);
ii **Precisão**: Fração de acerto de todos os acessos permitido, ou seja, dentre todas as classificações de acesso permitido que o modelo fez, quantas estão corretas;
iii **Revocação**: Fração de acerto dentre todos os acertos possíveis, ou seja, dentre todas as situações em que o acesso permitido é o valor esperado, quantas estão corretas;
iv **F1 score**: Média harmônica entre Precisão e Revocação;
v **Taxa de Verdadeiro Negativo (TNR):** Proporção de acessos negados que devem ser genuinamente negados;
vi **Complexidade (Comp):** Somatório do tamanho de todas as regras da ACP.

5.3. Resultados

As Tabelas 1 e 2 sumarizam os resultados obtidos do método de comparação (A.Dec) e nosso método (A.Dec+Gen) em relação ao dataset sintético e o da *Amazon*, respectivamente. Nelas, apresentamos cada instância gerada a partir dos datasets apresentados

Instâncias	Técnica	Acurácia	Precisão	Revocação	F1	TNR	Comp.
Sintético#1	A.Dec (N)	0.818	0.868	0.900	0.884	0.544	**35**
	A.Dec (O)	0.832	0.915	0.862	0.888	**0.733**	58
	A.Dec (U)	0.795	0.904	0.820	0.860	0.711	72
	A.Dec+Gen (N)	0.882	0.904	**0.947**	**0.925**	0.667	74
	A.Dec+Gen (O)	0.858	0.912	0.902	0.907	0.711	60
	A.Dec+Gen (U)	**0.876**	**0.917**	0.922	0.920	0.722	59
Sintético#2	A.Dec (N)	0.846	0.887	0.925	0.905	0.543	57
	A.Dec (O)	0.860	0.900	0.927	0.913	0.602	58
	A.Dec (U)	0.857	0.905	0.916	0.910	0.626	**55**
	A.Dec+Gen (N)	0.875	0.882	**0.974**	0.925	0.496	89
	A.Dec+Gen (O)	**0.907**	**0.930**	0.955	**0.942**	**0.72**	70
	A.Dec+Gen (U)	0.863	0.910	0.919	0.914	0.650	59
Sintético#3	A.Dec (N)	0.793	0.812	**0.951**	0.876	0.264	83
	A.Dec (O)	0.759	0.934	0.740	0.826	0.826	**58**
	A.Dec (U)	0.802	0.910	0.825	0.865	0.727	59
	A.Dec+Gen (N)	0.873	0.927	0.907	0.917	0.760	69
	A.Dec+Gen (O)	**0.892**	0.918	0.945	**0.931**	0.716	73
	A.Dec+Gen (U)	0.826	**0.951**	0.816	0.879	**0.859**	77
Sintético#4	A.Dec (N)	0.812	0.850	0.919	0.883	0.457	64
	A.Dec (O)	0.809	0.845	0.921	0.882	0.437	64
	A.Dec (U)	0.798	0.874	0.861	0.868	0.586	**50**
	A.Dec+Gen (N)	0.867	**0.907**	0.921	0.914	**0.685**	84
	A.Dec+Gen (O)	**0.876**	0.874	**0.980**	**0.924**	0.526	84
	A.Dec+Gen (U)	0.846	0.893	0.909	0.901	0.636	73

Tabela 1. Dataset Sintético

e mostramos a desempenho de ambos os métodos em relação às métricas apresentadas anteriormente. Nestas tabelas, ambos os métodos foram utilizados em conjunto com o balanceamento nulo (N), *oversampling* (O) e *undersampling* (U).

Na Tabela 1 é possível observar que a acurácia do método proposto é superior em todas as instâncias. Destaque para a instância Sintético#3, em que nosso método, aplicado com a técnica de balanceamento *oversampling*, apresentou uma acurácia aproximadamente 10% superior em relação ao método de comparação. Sobre a precisão e revocação, ambas apresentaram resultados superiores em nosso método. Consequentemente, o *F1 score* apresentou um melhor desempenho em nossa abordagem, ficando acima de 0.9 em todas as instâncias. Ter um valor alto nessa métrica é importante, pois nossa base de dados é bastante desbalanceada. Em situações como essa, um baixo valor do *F1 score* significa que a ACP criada está enviesada para aceitar mais acessos do que negar. Essa característica foi evidenciada na taxa de verdadeiro negativo (TNR), em que nosso método foi capaz de aumentar a capacidade da ACP de negar acessos corretamente em todas as instâncias, exceto a primeira. Em relação à complexidade da ACP, é possível observar um movimento contrário, em que nosso método apresentou regras mais complexas do que o comparado. Esse resultado foi esperado, uma vez que a complexidade não está diretamente introduzida na métrica de aptidão apresentada na Seção 4.3. Sendo assim, uma vez que a complexidade foi utilizada apenas como critério de desempate, nota-se que nosso método priorizou a acurácia e a revocação.

A Tabela 2 mostra que, nosso método apresenta ganhos mesmo quando utilizado em dados reais. Em relação à acurácia, nosso método apresenta melhores resultados em todas as instâncias. Em relação à precisão, nota-se que, ainda que nosso método apresen-

Instâncias	Técnica	Acurácia	Precisão	Revocação	F1	TNR	Comp.
Rec.25993	A.Dec (N)	0.911	0.957	0.949	0.953	0.167	**5**
	A.Dec (O)	0.935	0.958	0.974	0.966	0.167	**5**
	A.Dec (U)	0.919	0.957	0.957	0.957	0.167	**5**
	A.Dec+Gen (N)	**0.967**	**0.975**	0.991	**0.983**	**0.500**	14
	A.Dec+Gen (O)	0.959	0.959	**1**	0.979	0.167	10
	A.Dec+Gen (U)	0.951	0.959	0.991	0.975	0.167	7
Rec.4675	A.Dec (N)	0.988	0.992	0.996	0.994	0.333	**1**
	A.Dec (O)	0.988	0.992	0.996	0.994	0.333	2
	A.Dec (U)	0.984	0.992	0.992	0.992	0.333	**1**
	A.Dec+Gen (N)	0.988	0.992	0.996	0.994	0.333	5
	A.Dec+Gen (O)	**0.992**	0.992	**1**	**0.996**	0.333	12
	A.Dec+Gen (U)	0.988	**0.996**	0.992	0.994	**0.667**	3
Rec.75078	A.Dec (N)	0.968	0.984	0.984	0.984	0.333	3
	A.Dec (O)	**0.976**	0.984	**0.992**	**0.988**	0.333	3
	A.Dec (U)	0.944	0.991	0.951	0.971	**0.667**	**2**
	A.Dec+Gen (N)	**0.976**	0.984	**0.992**	**0.988**	0.333	6
	A.Dec+Gen (O)	**0.976**	0.984	**0.992**	**0.988**	0.333	10
	A.Dec+Gen (U)	0.968	**0.992**	0.975	0.983	**0.667**	**2**
Rec.79092	A.Dec (N)	0.979	**0.986**	0.993	0.989	**0.600**	**5**
	A.Dec (O)	0.959	0.979	0.979	0.979	0.400	**5**
	A.Dec (U)	0.945	0.978	0.965	0.971	0.400	10
	A.Dec+Gen (N)	**0.986**	**0.986**	**1**	**0.993**	**0.600**	10
	A.Dec+Gen (O)	**0.986**	**0.986**	**1**	**0.993**	**0.600**	13
	A.Dec+Gen (U)	0.945	0.978	0.965	0.971	0.400	12

Tabela 2. Dataset Amazon

tasse melhores resultados, os valores são novamente próximos ao do método comparado. Com relação à revocação, entretanto, notamos uma melhora em todos os casos, sendo a instância Rec.75078, a única que não apresentou a métrica de revocação perfeita. Consequentemente, o F1 score apresentou um melhor desempenho em nossa abordagem, ficando acima do método comparado em todas as instâncias. Quanto a taxa de verdadeiro negativo, é importante notar que nosso método obteve resultados expressivos se contrastado com o método de comparação. O método de comparação assume como base uma árvore de decisão simples e gera ACPs com base nessa informação. Em nosso método, por outro lado, alguns dados sintéticos são adicionados para realizar a etapa de evolução. Consequentemente, nosso método é capaz de extrapolar informações não apresentadas no dataset de entrada, gerando um conjunto de ACPs mais complexas em relação ao método de comparação, porém com uma chance de, em um momento da evolução, gerar um ACP correta a partir de dados não apresentados na entrada. Uma vez que esses dados conseguem gerar uma regra de negar um acesso corretamente e a base de dados possui poucos exemplos dessa negação, o nosso método foi capaz de manter essa regra na ACP gerada. Devido essa característica, notamos que os nossos ganhos vem acompanhado de um pequeno aumento na complexidade da regra, uma vez que as distorções inseridas acabaram sendo adicionadas na ACP. Porém, notamos que esse aumento não é exponencial e permanece na mesma ordem de grandeza do método comparado.

6. Conclusões

Este trabalho apresentou um método de mineração de políticas que se utiliza de logs de acesso e aprendizado de máquina supervisionado para extrair ACPs precisas e sucintas. Mostrou-se que, através da união das árvores de decisão com uma heurística genética,

é possível obter ACPS que ofereçam melhor desempenho na permissão ou negação de um acesso. Os resultados indicaram que nossa proposta possuiu melhores resultados em relação à acurácia, à precisão, à revocação e à taxa de verdadeiro negativo. Neste trabalho, foi possível observar que em bases extremamente desbalanceadas, com pouco exemplos de acesso negado, ambos os métodos comparados possuíram dificuldade em identificar regras de negação. No entanto, como nosso método possui uma característica evolutiva, ele foi capaz de gerar políticas que permitem negar acessos corretamente com base no que foi aprendido na injeção de distorções nas bases de dados. Foi identificado que essas distorções melhoram nossos ganhos, porém ao ocorreu um aumento na complexidade nas regras. Porém, esse aumento na complexidade não é exponencial e ainda está na mesma ordem de grandeza do que o método comparado. Como trabalhos futuros, deseja-se aplicar diferentes técnicas para a geração de políticas e uma análise mais aprofundada em relação a complexidade das regras. De modo geral, deseja-se aplicar técnicas para reduzir o impacto das distorções que foram inseridas na política resultante.

Referências

Abirami, G. and Venkataraman, R. (2019). Performance analysis of abac and abac with trust (abac-t) in fine grained access control model. In *IEEE 11th ICoAC*, pages 372–375.

Amazon (2013). Amazon.com - Employee Access Challenge. `https://www.kaggle.com/c/amazon-employee-access-challenge/`. [Último acesso em 21/02/2021].

Bui, T. and Stoller, S. D. (2020). Learning attribute-based and relationship-based access control policies with unknown values. In *Int. Conf. on Information Systems Security*, pages 23–44.

Cotrini, C., Weghorn, T., and Basin, D. (2018). Mining abac rules from sparse logs. In *IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 31–46.

Ding, S. and Ma, M. (2021). An attribute-based access control mechanism for blockchain-enabled internet of vehicles. In *Advances in Computer, Communication and Computational Sciences*.

Hu, V. C., Ferraiolo, D., Kuhn, R., Friedman, A. R., Lang, A. J., Cogdell, M. M., Schnitzer, A., Sandlin, K., Miller, R., Scarfone, K., et al. (2013). Guide to attribute based access control (abac) definition and considerations (draft). *NIST special publication*, 800(162).

Iyer, P. and Masoumzadeh, A. (2018). Mining positive and negative attribute-based access control policy rules. In *ACM on Symposium on Access Control Models and Technologies*.

Kramer, O. (2017). Genetic algorithms. In *Genetic algorithm essentials*, pages 11–19. Springer.

Lu, H., Vaidya, J., and Atluri, V. (2008). Optimal boolean matrix decomposition: Application to role engineering. In *IEEE 24th Int. Conf. on Data Engineering*, pages 297–306.

Medvet, E., Bartoli, A., Carminati, B., and Ferrari, E. (2015). Evolutionary inference of attribute-based access control policies. In *Int. Conf. on Evolutionary Multi-Criterion Optimization*.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., et al. (2011). Scikit-learn: Machine learning in python. *the Journal of machine Learning research*, 12:2825–2830.

Qiu, J., Tian, Z., Du, C., Zuo, Q., Su, S., and Fang, B. (2020). A survey on access control in the age of internet of things. *IEEE Internet of Things Journal*, 7(6):4682–4696.

Samarati, P. and de Vimercati, S. C. (2000). Access control: Policies, models, and mechanisms. In *International School on Foundations of Security Analysis and Design*, pages 137–196. Springer.

Talukdar, T., Batra, G., Vaidya, J., Atluri, V., and Sural, S. (2017). Efficient bottom-up mining of attribute based access control policies. In *Int. Conf. on Collaboration and Internet Computing*.

Thengade, A. and Dondal, R. (2012). Genetic algorithm–survey paper. In *MPGI*.

Umadevi, S. and Marseline, K. J. (2017). A survey on data mining classification algorithms. In *Int. Conf. on Signal Processing and Communication*, pages 264–268.

Xu, Z. and Stoller, S. D. (2014). Mining attribute-based access control policies. *IEEE Transactions on Dependable and Secure Computing*, 12(5):533–545.

Zhu, Y., Huang, D., Hu, C.-J., and Wang, X. (2014). From rbac to abac: constructing flexible data access control for cloud storage services. *IEEE Transactions on Services Computing*.