Exploratory survey of access control paradigms and policy management engines

Pavlo Hlushchenko[1], Valerii Dudykevych[1]

[1] *Lviv Polytechnic National University, S. Bandery str. 12, Lviv, 79013, Ukraine*

Abstract

This article presents an in-depth exploration of access control paradigms and policy management engines, aiming to provide insights into their functionalities, strengths, and limitations. The study examines three prominent access control paradigms: Role-Based Access Control (RBAC), Attribute-Based Access Control (ABAC), and Relationship-Based Access Control (ReBAC), elucidating their principles and real-world applications. Furthermore, the article evaluates major policy management engines, including Google Zanzibar, AWS Cedar, and Open Policy Agent (OPA),delineating their architectures, purposes, and strengths. Through this comprehensive analysis, key observations emerge regarding the evolving landscape of access control mechanisms and the critical role of policy management engines in orchestrating access control policies. The findings underscore the importance of aligning access control mechanisms with organizational requirements and use case scenarios, while also highlighting avenues for further research and experimentation in exploring novel approaches to access control and policy management with direct or complementary use of AI.

Keywords

Access control, rbac, abac, rebac, open policy agent, aws cedar, google zanzibar, policy engines

1. Introduction

The ever-expanding digital landscape necessitates robust mechanisms to safeguard access to sensitive information and resources. Access control paradigms and policy management engines emerge as indispensable tools in this context, enabling organizations to establish fine-grained control over access privileges while mitigating security vulnerabilities and upholding regulatory compliance.

This article embarks on a comprehensive exploration of these critical technologies. It delves into the diverse landscape of access control paradigms, their historical context, and introduces prominent models like Role-Based Access Control (RBAC), Attribute-Based Access Control (ABAC), and Relationship-Based Access Control (RBAC). Subsequently, the article examines the critical role of policy management engines in facilitating the implementation of these paradigms. It delves into the functionality and significance of these engines, showcasing specific examples such as Google Zanzibar, Open Policy Agent (OPA), and AWS Cedar.

Through a high-level comparative analysis of these paradigms and engines, this article aims to equip readers with a foundational understanding of this crucial facet of cybersecurity and the current state of things in the field. This knowledge empowers organizations to make informed decisions regarding the selection and implementation of solutions that best align with their specific security needs and operational requirements.

Implementing access control, particularly within a zero trust architecture, serves as a proactive defense against a variety of cyber threats, notably ransomware attacks [1]. By adhering to the principle of least privilege, access control limits unauthorized access to sensitive data, mitigating the impact of ransomware incidents. Additionally, robust access control measures, such as multi-factor authentication, bolster defenses against ransomware by restricting unauthorized entry and reducing the attack surface. In a zero trust framework, where trust is never assumed, continuous verification of user identities and device integrity reinforces security measures, safeguarding critical assets against ransomware and other cyber threats [2].

CMIS-2024: Seventh International Workshop on Computer Modeling and Intelligent Systems, May 3, 2024, Zaporizhzhia, Ukraine

pavlo.k.hlushchenko@lpnu.ua (P. Hlushchenko); valerii.b.dudykevych@lpnu.ua (V. Dudykevych)

0000-0002-1262-5484 (P. Hlushchenko); 0000-0001-8827-9920 (V. Dudykevych)

2. Access Control Paradigms

Access control paradigms serve as foundational frameworks for managing and enforcing authorization policies within systems and applications. These paradigms delineate the rules and principles governing access to resources based on various attributes such as user roles, attributes, or contextual information.

The development of access control paradigms has evolved over time, reflecting the growing complexity of systems and escalating security concerns. Early efforts focused on simple mechanisms like password-based authentication, which later progressed to more sophisticated models offering granular control and flexibility.

Some of the prominent access control paradigms include:

- Discretionary Access Control (DAC): This traditional model allows the owner of a resource to grant access permissions to other users. Emerging in the early days of computing, DAC was well-suited for smaller, less complex systems. However, its reliance on individual users to manage access permissions can lead to scalability challenges and potential security vulnerabilities in larger environments
- Mandatory Access Control (MAC): In contrast to DAC, MAC enforces centrally defined access control policies that are pre-determined and not modifiable by individual users. This model, often found in military and government systems, prioritizes security over flexibility. However, its centralized management can be less adaptable in dynamic and complex environments
- Role-Based Access Control (RBAC): This widely adopted model assigns users to pre-defined roles, with every role containing specific individual permissions. By grouping users who have similar access requirements, this method streamlines the management of access. RBAC emerged as a response to the limitations of DAC in larger organizations requiring a more structured approach to authorization [3]
- Attribute-Based Access Control (ABAC): This dynamic model grants access based on a combination of attributes associated with the user, the resource, the environment, and the requested action. This flexibility allows for fine-grained control and policy enforcement based on various contextual factors. ABAC emerged as a response to the increasing need for more granular and adaptable access control in complex systems with diverse access requirements
- Relationship-Based Access Control (ReBAC): This emerging model focuses on the relationships between entities (users, resources, etc.) when making authorization decisions. Access rights are granted based on established relationships, such as ownership, membership in groups, or hierarchical structures. ReBAC offers a dynamic and context-aware approach to access control, particularly relevant in social network and collaborative environments

These paradigms have evolved in response to the changing landscape of technology and security requirements. RBAC, initially introduced in the 1990s to address the complexities of access management in large organizations, laid the groundwork for subsequent advancements in access control methodologies. Its flexibility allows it to implement MAC and DAC. ABAC emerged as a response to the need for more dynamic and context-aware access control mechanisms, accommodating the intricacies of modern computing environments. ReBAC represents a specialized approach tailored to organizations where access decisions are heavily influenced by inter-entity relationships and dependencies. While these approaches excel at their use cases, they can also be used together to complement each other. In the subsequent sections, we will delve deeper into each access control paradigm, examining their principles, strengths, and limitations, to provide a thorough understanding of their role in shaping access control strategies.

2.1. Role-Based Access Control (RBAC)

RBAC is a widely adopted access control paradigm that revolves around the concept of an organization and the roles defined in it. Access rights in RBAC are assigned to roles, and roles are then assigned to users based on their respective job functions and responsibilities. This hierarchical approach simplifies access management and enhances security by following the principle of least privilege, where users are granted only the permissions they need for their job. To make the access control management workflow easier and more consistent, it is a best practice to create user groups, add users to the respective groups and then assign roles to the groups. This way the onboarding/offboarding process is straightforward as well as movement of a user between different teams or projects inside an organization.

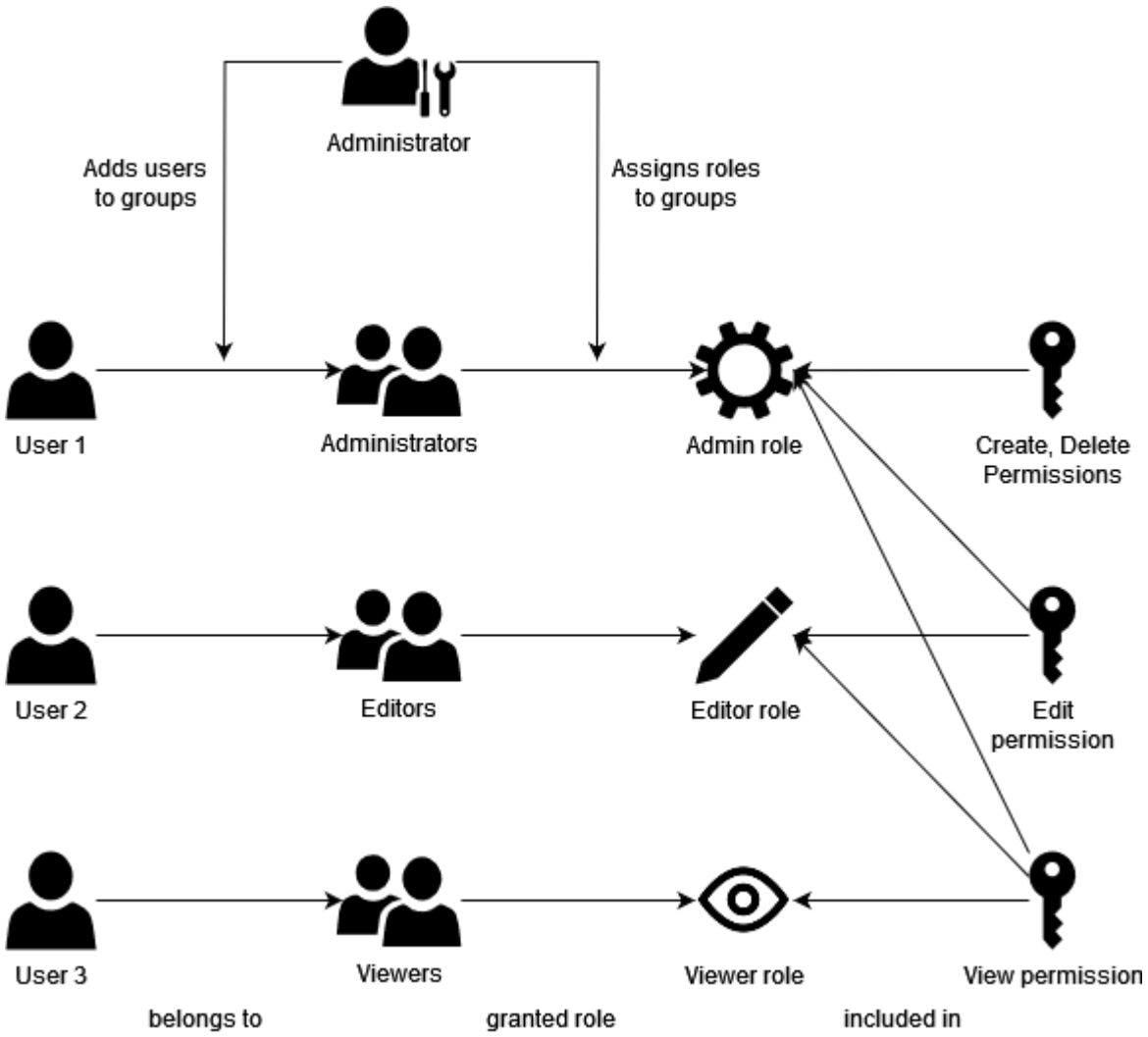

Figure 1: RBAC architecture

Key concepts in RBAC are:

- Roles: Represent a collection of permissions associated with a specific job function or responsibility within an organization. Examples include Administrator, Editor, Viewer, etc.
- Users: Individuals who are assigned to one or more roles based on their job responsibilities and access requirements.
- Permissions: Define the specific operations a user can perform on a resource, such as create, view, edit, delete, etc.
- Resources: Represent entities within the system that require access control, such as files, data, applications, services, etc.

RBAC model has several upsides that make it a preferred choice for access management within organizations. Firstly, its simplicity streamlines access control by organizing permissions into roles, significantly reducing administrative overhead and complexity. This structured approach also enhances scalability, allowing for seamless adaptation to organizational growth. With RBAC, new users can effortlessly be added to existing groups with pre-assigned roles, while the creation of new roles can be easily accommodated as organizational needs evolve. Furthermore, RBAC promotes consistency in access control policies throughout the organization, ensuring uniformity and minimizing the likelihood of errors or discrepancies. Lastly, RBAC enhances security by enforcing the principle of least privilege, thereby reducing the risk of unauthorized access and potential security breaches.

While RBAC offers significant benefits, it also presents several limitations that organizations should consider. The practical implementation of RBAC in real-world scenarios presents several challenges, notably the issue of role explosion. This occurs when the number of roles becomes excessively large and unmanageable, complicating the administration and efficiency of the access control system. It makes managing a multitude of roles challenging and can lead to a significant administrative overhead.

Aaron Elliott and S. Knight (2010) [4] highlight this problem and propose new approaches based on real-world practices to mitigate it. Secondly, role maintenance, including tasks such as role creation, modification, and deletion, can be time-consuming and resource-intensive, particularly in environments with high dynamics. Additionally, RBAC may exhibit limited flexibility in accommodating dynamic access control requirements or complex authorization scenarios involving contextual attributes or relationships. Lastly, there is a potential for abuse, as improperly defined roles or assigning users to inappropriate roles can introduce security vulnerabilities, underscoring the importance of careful role management practices.

Another significant challenge in RBAC implementation is adapting the system to context-sensitive environments, such as cloud and fog networks, where dynamic access control is crucial. A.S.M. Kayes et al. (2020) [5] discuss these context-aware access control mechanisms, noting the difficulties in applying traditional RBAC systems effectively in such rapidly changing settings. They suggest enhancements to address these issues, providing a detailed taxonomy of solutions based on real-world application case studies. These studies illustrate both the potential and the complexities of RBAC systems, emphasizing the need for continuous adaptation and innovation to overcome practical barriers in diverse operational contexts.

RBAC's simplicity, scalability, and security benefits have cemented its position as a foundational access control paradigm in many organizations and cloud environments. However, as computing environments evolve, organizations may supplement RBAC with more dynamic and context-aware access control mechanisms to address emerging challenges, requirements.

RBAC simplifies permission management by assigning permissions to roles rather than individuals, yet it faces specific challenges and potential vulnerabilities. The phenomenon of role explosion, where the number of roles grows unwieldy, can complicate the management and increase the risk of errors and security gaps. RBAC's static nature does not account for contextual factors, potentially leading to over-privileged accounts and security risks. Furthermore, RBAC may struggle with enforcing complex segregation of duties (SoD) policies, increasing the risk of fraud or misuse if roles are not meticulously defined. Administrative overhead is another concern, as managing a large number of roles and permissions can become labor-intensive and prone to delays, impacting both security and operational efficiency. Finally, scalability issues can arise as the system expands, potentially leading to inconsistencies in enforcing security policies across an organization.

Mitigating these vulnerabilities in RBAC involves regular audits of roles and permissions, implementing role hierarchies to streamline management, utilizing automated tools to reduce errors, and potentially integrating RBAC with dynamic models like ABAC or ReBAC to enhance flexibility. Such strategies help maintain security and efficiency in RBAC systems, even as organizational complexity grows.

2.2. Attribute-Based Access Control (ABAC)

Attribute-Based Access Control (ABAC) is a flexible, extensible and dynamic access control paradigm that grants or denies access based on a combination of attributes associated with various entities and the requested action. These attributes can include:

1. Subject attributes: Characteristics of the user, such as job title, department, location, security clearance, etc.
2. Object attributes: Characteristics of the resource being accessed, such as data classification level, creation date, location, etc.
3. Action attributes: Characteristics of the operation being requested, such as read, write, delete, etc.
4. Environment attributes: Contextual factors like time of day, network location, device type, etc.

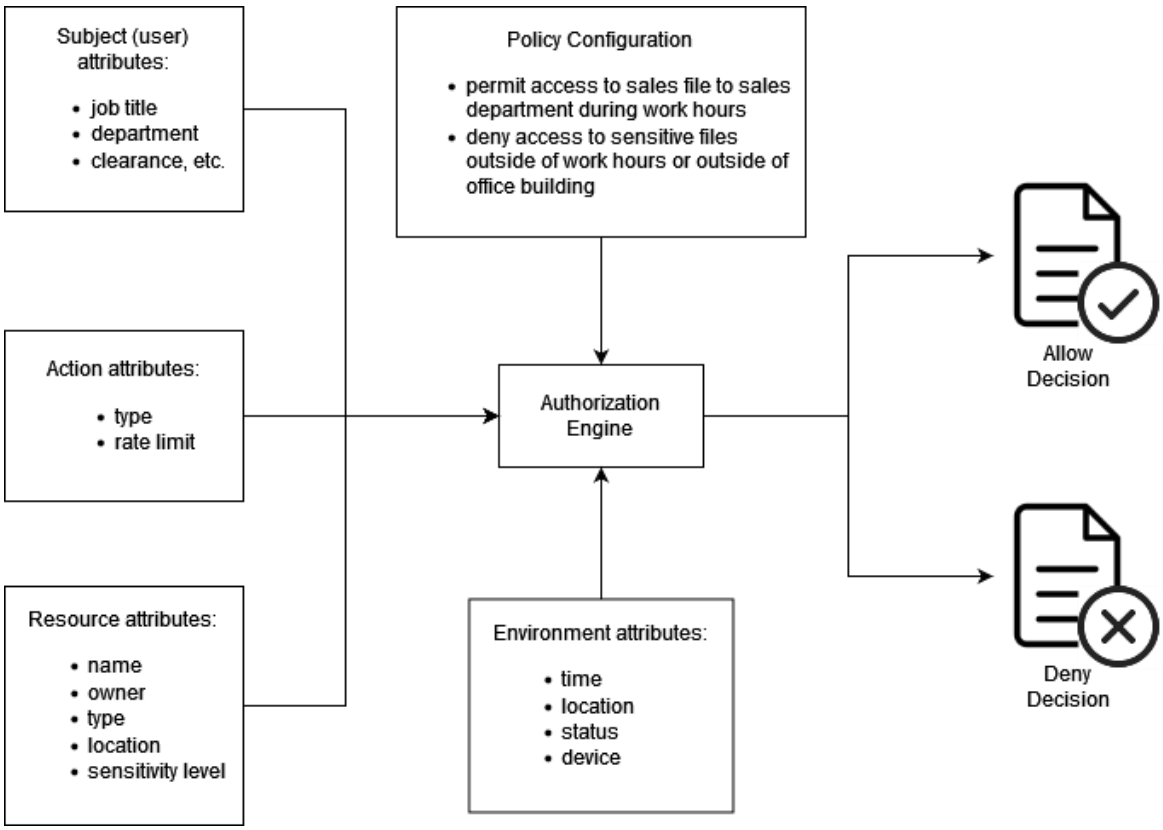

Figure 2: ABAC architecture

While the concept of ABAC existed for many years, it was endorsed by the Federal Chief Information Officers Council in 2011 as a model to adopt from considering growing complexity of systems and the need for more granular and adaptable access control [6] compared to traditional models like RBAC. While RBAC simplifies access management, it can be less flexible in dynamic environments where user and resource attributes are constantly changing as shown in Figure 2. The main concepts in ABAC model are:

- Attributes: Represent characteristics associated with users, resources, actions, and the environment.
- Policies: Define the access control rules based on the combination of relevant attributes. These policies dictate which combinations of attributes grant or deny access for specific actions on specific resources.
- Policy Engine: Analyzes the attributes of the user, resource, action, and environment to evaluate the applicable policies and make an access decision (grant or deny).

ABAC offers several strengths that make it a compelling approach to access control within organizations. Firstly, ABAC facilitates fine-grained control by allowing for highly granular access control based on various contextual factors, which are represented as attributes. This enables organizations to tailor access policies with precision, accommodating diverse scenarios and requirements. Secondly, the dynamic nature of ABAC enhances flexibility, as it enables the system to adapt to changing access requirements without the need to modify pre-defined roles. This adaptability simplifies the management of access control policies, particularly in dynamic environments where access needs may evolve rapidly. Lastly, ABAC can contribute to improved security by enabling the enforcement of more complex and nuanced access control policies. By considering multiple attributes and factors, ABAC can mitigate risks more effectively compared to simpler access control models, thereby enhancing overall security posture.

While ABAC offers significant advantages, it also presents certain concerns that organizations should consider. Firstly, managing and enforcing ABAC policies can be more complex compared to RBAC, primarily due to the dynamic nature of attributes and the requirement for robust policy engines. The dynamic nature of attributes introduces complexity in policy definition and enforcement, potentially leading to increased administrative overhead depending on the implementation. Secondly, evaluating complex ABAC policies can incur performance overhead compared to simpler access control models. The need to consider multiple attributes and factors during policy evaluation may result

in additional processing time, impacting system performance, particularly in high-throughput environments. Therefore, organizations implementing ABAC should carefully consider these potential challenges and ensure adequate resources and processes are in place to address them effectively.

The practical implementation of Attribute-Based Access Control (ABAC) in various real-world scenarios reveals several challenges, primarily associated with the complexity and dynamism of modern environments. ABAC systems, while flexible and powerful, often struggle with the nuanced requirements of fine-grained access control, particularly in complex infrastructures like cloud and fog networks, as well as IoT systems. A notable issue highlighted in studies such as by A.S.M. Kayes et al. [5] and S.C. Hubli and G.P. Potdar [7], is the management of attributes that dynamically change with context, complicating the enforcement of access policies across different scenarios.

Moreover, the integration of ABAC in environments like smart homes or military applications, as discussed by G. Goyal, P. Liu, and S. Sural [8], demonstrates challenges related to scalability and performance, where the systems must efficiently manage a vast number of attributes without compromising on security or operational speed. These challenges are compounded in settings that require robust security measures against evolving threats and complex, frequently changing access rules. The research by S. Parkinson and S. Khan [9] further illuminates these issues, pointing out that real-world applications of ABAC still face significant hurdles in terms of policy management and the real-time processing of attributes, emphasizing the need for ongoing development to enhance the practicality and effectiveness of ABAC systems.

ABAC offers flexibility and granularity in managing access based on attributes, but it also comes with potential vulnerabilities. The complexity of managing a large number of detailed policies can lead to misconfigurations and security loopholes, while the performance impact of dynamic attribute evaluation can affect system efficiency and scalability. Additionally, the integrity of the attributes is crucial; any compromise can lead to incorrect access decisions, potentially allowing unauthorized access and data breaches. ABAC systems are also susceptible to insider threats, as knowledgeable insiders could manipulate attributes or exploit policy gaps. Furthermore, ensuring compliance with regulatory requirements can be complex due to the dynamic nature of ABAC policies, posing additional challenges in proving consistent compliance across all scenarios.

To mitigate these vulnerabilities, organizations should develop robust policies, perform rigorous testing, ensure accurate and secure attribute management and verification, optimize performance, and regularly monitor and review ABAC systems. These measures help maintain the system's integrity and compliance while harnessing ABAC's potential for dynamic and flexible access control.

Overall, ABAC offers a powerful and versatile approach to access control, especially in complex environments and scenarios requiring fine-grained control and dynamic rules [10]. However, the increased complexity requires careful consideration of its implementation and ongoing management.

2.3. Relationship-Based Access Control (ReBAC)

Relationship-Based Access Control (ReBAC) is an access control paradigm that considers the relationships between entities within an organization when making access control decisions. Unlike traditional access control models that primarily focus on user roles or attributes, ReBAC takes into account the relationships between users, resources, and other entities to determine access permissions [11]. Key concepts and components of ReBAC include:

- Relationship Definition: ReBAC begins with the identification and definition of relationships between entities within an organization. These relationships can include hierarchical relationships (e.g., supervisor-subordinate), team-based relationships (e.g., project team membership), or custom-defined relationships based on organizational structure or business processes.
- Relationship-Based Policies: Access control policies in ReBAC are defined based on the relationships between entities. These policies specify the conditions under which access is granted or denied based on the nature and characteristics of the relationships between users, resources, and other entities.
- Contextual Access Control: ReBAC incorporates contextual information and relationship dynamics into access control decisions. Access permissions may vary based on the specific relationships between entities, the nature of the access request, and the contextual environment in which the request is made.
- Fine-Grained Control: ReBAC offers fine-grained control over access permissions based on relationship attributes and characteristics. Organizations can define complex access control policies

that reflect the nuances of relationships within the organization, enabling precise control over access to resources.

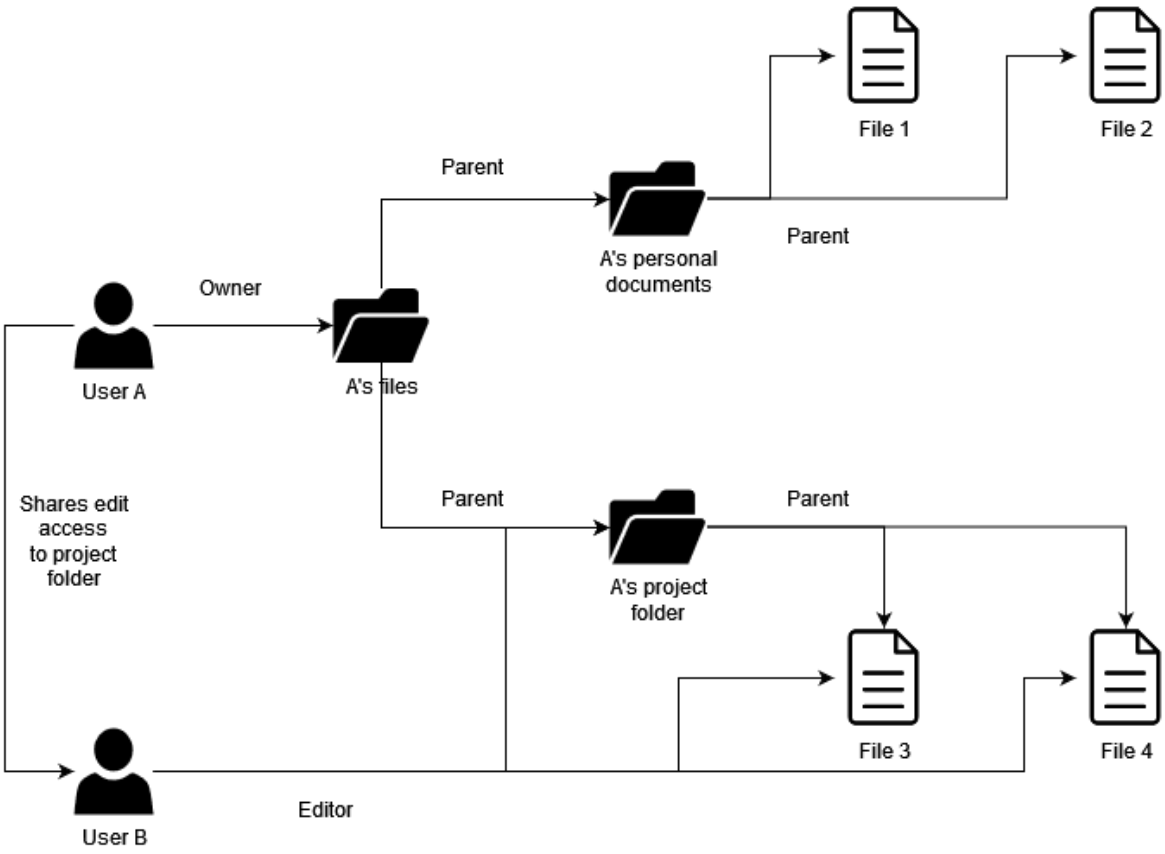

Figure 3: ReBAC architecture

ReBAC boasts several strengths that make it a formidable approach to access control within organizations. Firstly, it offers granular and comprehensive access control by providing precise control over access permissions based on the relationships between entities. This capability enables organizations to enforce tailored access control policies aligned with specific relationship dynamics, enhancing security and compliance. Secondly, ReBAC incorporates context awareness into access control decisions, allowing organizations to consider relationship dynamics, organizational structure, and business processes when granting or denying access. This contextual understanding enhances the accuracy and relevance of access control decisions, bolstering overall security posture. Additionally, ReBAC exhibits adaptability to changing organizational structures and relationship dynamics, enabling organizations to adjust access control policies dynamically as relationships evolve over time. This flexibility ensures that access control remains aligned with organizational needs and evolving business requirements. Moreover, ReBAC excels in environments where relationships play a significant role in access control, such as social media platforms or collaborative workspaces, providing improved control and security in such contexts. Lastly, ReBAC policies offer flexibility in capturing various types of relationships, allowing organizations to tailor access control based on specific contexts and requirements, further enhancing precision and effectiveness in access control enforcement.

Despite the notable strengths and a completely different approach to access control, ReBAC also presents several limitations that organizations should consider. Firstly, implementing ReBAC can be complex, especially in organizations with intricate relationship structures and dynamic relationship dynamics. Sophisticated policy management and enforcement mechanisms may be necessary to effectively implement ReBAC, adding complexity to the access control framework. Secondly, improperly defined relationships or policies can lead to unforeseen access control outcomes, potentially compromising security and compliance. Additionally, ReBAC requires capturing and representing various types of relationships, which may pose challenges in accurately modeling complex relationship dynamics. Furthermore, relationship management, including capturing and managing relationship data and ensuring consistency across systems, can be challenging, particularly in distributed or heterogeneous environments. Lastly, the dynamic evaluation of relationship-based policies in real-time may introduce performance overhead, particularly in high-throughput environments or systems with

large numbers of relationships, impacting system performance and responsiveness. Therefore, organizations considering the adoption of ReBAC should carefully assess these limitations and ensure appropriate strategies are in place to address them effectively.

The practical implementation of Relationship-Based Access Control (ReBAC) in real-world scenarios encounters several significant challenges, primarily arising from the complexity and dynamic nature of the relationships it needs to manage. One of the primary difficulties is the policy definition complexity, as highlighted by C. Arora (2022) [12]. This study discusses the challenges such as solving homomorphism problems that emerge when defining and enforcing relationships within access control frameworks, especially in settings where relationships can dynamically change, thus complicating the access control process.

Moreover, the scalability of ReBAC in distributed systems like IoT, blockchain, and cloud environments presents another layer of complexity. L. Golightly et al. (2023) [13] explore these challenges in the context of IoT, emphasizing issues related to policy management, enforcement efficiency, and the system's ability to handle a vast network of dynamic relationships without performance degradation. Similarly, ASM Kayes et al. (2020) [5] point out the difficulty in expressing and managing access control policies that effectively capture the fluid nature of user relationships and roles within decentralized networks such as cloud and fog computing environments. These environments require a more flexible and scalable approach to access control, which current ReBAC implementations struggle to provide.

These studies collectively indicate that while ReBAC is theoretically robust in utilizing relationships for access control, its practical application is often hindered by issues related to complexity in policy formulation, scalability, and the dynamic management of relationships in high-demand environments.

Managing complex and dynamic relationships within large organizations can become cumbersome and error-prone, leading to potential misconfigurations and security breaches. Additionally, ReBAC systems face scalability challenges as they must update access rights in real-time based on changing relationships, which can strain system resources and lead to latency issues.

The specification and enforcement of ReBAC policies are complex due to the intricate, multi-layered nature of relationships and their context dependency. Any inaccuracies or manipulations in relationship data can compromise access control decisions, leading to unauthorized access and increasing the risk of insider threats. Furthermore, the fluidity of ReBAC makes auditing and compliance particularly challenging, as demonstrating consistent application of access controls in a continually adapting environment can be difficult. To mitigate these vulnerabilities, organizations should implement robust management tools for relationships, sophisticated policy evaluation engines, and extensive auditing features to maintain continuous integrity and compliance in the ReBAC system.

As shown in Figure 3, User A is assigned the role Owner of Files folder, which means he is also the owner of every file and folder for which Files is the parent folder. ReBAC's focus on relationships between entities within an organization offers a unique perspective on access control, enabling organizations to enforce fine-grained access control policies that reflect the nuances of relationship dynamics. By considering relationships alongside user roles and attributes, ReBAC enhances security and enables organizations to enforce more context-aware and adaptable access control policies.

3. Overview of Major Policy Management Engines

In the realm of access control and authorization, policy management engines play a crucial role in enforcing access control policies and facilitating the implementation of various access control paradigms. A policy management engine can be defined as a software component or system that is responsible for the administration, enforcement, and evaluation of access control policies within an organization's IT infrastructure.

The primary purpose of policy management engines is to provide a centralized platform for defining, managing, and enforcing access control policies across diverse computing environments. These engines enable organizations to translate high-level security requirements and access control policies into actionable rules and mechanisms that govern access to resources.

Policy management engines facilitate the implementation of access control paradigms such as RBAC, ABAC, and ReBAC by providing the necessary infrastructure for policy definition, evaluation, and enforcement.

Policy management engines have evolved in response to the growing complexity of IT environments, the proliferation of access control paradigms, and the increasing demands for fine-grained access control. Initially, access control policies were often implemented through ad-hoc mechanisms embedded within individual applications or systems, leading to fragmented and inconsistent access control practices.

As organizations recognized the need for centralized policy management and enforcement, dedicated policy management engines began to emerge. These engines provided standardized interfaces and tools for defining, managing, and enforcing access control policies across heterogeneous IT environments.

Over time, policy management engines have evolved to incorporate features such as policy modeling languages, policy evaluation engines, and integration with identity and access management (IAM) systems. Modern policy management engines leverage technologies such as machine learning and artificial intelligence to enhance policy enforcement and adaptability to changing access control requirements.

Policy management engines are utilized across various industries and sectors, including healthcare, finance, government, and e-commerce, to enforce access control policies and ensure compliance with regulatory requirements. These engines are often integrated with existing IT infrastructure, including directory services, authentication systems, and application servers, to provide seamless access control enforcement across the organization.

The future of policy management engines is likely to be shaped by advancements in areas such as cloud computing, AI, Internet of Things (IoT), and edge computing. As organizations embrace distributed and hybrid IT environments, policy management engines will need to evolve to support dynamic and scalable access control mechanisms that span across disparate computing platforms and environments.

Additionally, the convergence of access control, identity management, and security analytics is expected to drive the development of more intelligent and context-aware policy management engines that can adapt to evolving threat landscapes and user behavior patterns. As organizations grow, get more complex and dynamic and operate at ever-increasing scale, AI is going to be a key part of such systems in the future.

3.1. Open Policy Agent (OPA)

Open Policy Agent (OPA) emerged as a notable solution for policy-based authorization in the context of modern, cloud-native architectures and microservices. Developed against the backdrop of evolving cloud-native technologies and the challenges inherent in managing access control in distributed systems, OPA embodies a departure from traditional access control mechanisms towards a more declarative and flexible approach. [14]

The genesis of OPA lies in the recognition of the limitations of existing access control mechanisms in addressing the complexities of cloud-native environments. As organizations embraced microservice architectures and containerization, the need for a unified and adaptable policy enforcement solution became apparent. OPA's development represents a response to this growing demand for a policy engine capable of addressing the dynamic and heterogeneous nature of modern computing environments.

Advantages of leveraging OPA include its utilization of Policy-as-Code (PAC), where policies are defined through Rego code, providing all the benefits associated with code, such as versioning, reviews, and auditing. OPA also supports fine-grained permissions, accommodating RBAC, ABAC, and ReBAC (with certain adjustments), thus enabling granular permissions management. Furthermore, OPA offers a high degree of flexibility and extensibility, empowering organizations to tailor access control policies to their specific requirements and use cases. With a robust open-source community and widespread adoption across various industries, OPA benefits from continuous development and contributions, ensuring ongoing enhancements and improvements. Moreover, OPA's architecture is engineered for scalability and performance, leveraging in-memory caching to scale efficiently and deliver optimal performance.

Using OPA may also entail certain disadvantages. Firstly, there is a learning curve associated with mastering the Rego policy language and understanding OPA's functionalities, which may be challenging for users unfamiliar with declarative policy languages and policy-based authorization concepts. Secondly, operational complexity may arise when managing and maintaining OPA instances in production environments, especially in distributed or heterogeneous environments with multiple

OPA instances and policy repositories. This complexity can pose challenges in ensuring consistency and efficiency across OPA deployments, requiring careful planning and management to mitigate potential issues.

Additional factors to consider include the complexity associated with real-time policy updates, the reliance on multiple data sources in rules, which often requires bundling not easily achieved, and the synchronization challenges of multiple OPA instances when operating more than one concurrently.

While OPA alone does not directly address these challenges, they can be effectively resolved by leveraging OPAL [15] in conjunction with OPA. OPAL serves as an administrative layer atop OPA, facilitating real-time updates and offering solutions to address synchronization issues.

In conclusion, OPA is a well-established and broadly used solution that provides substantial benefits including versatility, separation of code and policy, ability to integrate with various systems, and policy evaluation done in an efficient manner. When enhanced with OPAL, OPA becomes even stronger, successfully tackling further complexities found in intricate access control settings.

3.2. AWS Cedar

Cedar, recently released by AWS in open-source, is a policy-as-code language that is designed to enhance management of IAM and access control processes. It offers a well-organized and scalable approach to permissions management, particularly impactful for application-level permissions management.

Cedar presents several advantages that contribute to its effectiveness in access control management within organizations. Firstly, akin to OPA, Cedar adopts a policy-as-code paradigm, employing a policy language for defining access control policies. This approach facilitates the treatment of policies as code, enabling versioning, auditing, and transparent review processes, thereby enhancing policy management practices. Secondly, Cedar prioritizes readability in its design, ensuring accessibility for both technical and non-technical team members. This emphasis on readability fosters collaboration and understanding across diverse teams within an organization. Additionally, Cedar adopts a structured and safety-oriented design approach [16], emphasizing safety by default. This commitment to security and correctness provides confidence through development practices guided by verification. Furthermore, Cedar is specifically designed for application-level authorization, catering to the unique access control requirements within applications [17]. Its design ensures effective enforcement of permissions at the application level, thereby enhancing security and compliance. Lastly, Cedar supports a variety of access control paradigms, including RBAC, ABAC, and ReBAC, enabling granular permissions management to accommodate diverse use cases and access control requirements effectively.

Although Cedar boasts significant benefits, it also entails certain drawbacks that organizations need to take into account. Firstly, Cedar currently lacks an extensive ecosystem of tools and modules compared to more established policy languages. This constraint could require extra work or custom solutions for particular tasks, which may affect the efficiency of policy management workflows. Additionally, given that it is a newly introduced policy language, Cedar boasts an active yet smaller community compared to, for instance, OPA. This smaller community might lead to less documentation, resources, and support available from community channels, which could present challenges for organizations needing help with Cedar-related initiatives. Therefore, organizations considering the adoption of Cedar should carefully assess these limitations and ensure appropriate strategies are in place to address them effectively.

3.3. Google Zanzibar

Google Zanzibar represents a pivotal advancement in access control systems, developed by Google to address the complex access control requirements of its distributed services and platforms. Originating from Google's internal needs for a scalable, centralized access control solution, Zanzibar was designed to meet the challenges posed by Google's vast and diverse ecosystem of services, users, and resources.

Zanzibar possibly draws its name from the historic island known for its significance as a trading hub, reflecting its role as a central hub for managing access control policies across Google's services and infrastructure. Historically, Google utilized a decentralized approach to access control, where each service managed its own access policies. However, as the company's services and user base grew exponentially, this decentralized model became increasingly challenging to manage and scale effectively.

Google Zanzibar represents a departure from this decentralized model, offering a centralized, globally-distributed access control system capable of managing access policies for millions of resources and users across Google's infrastructure. By consolidating access control policies into a unified system, Zanzibar enables Google to enforce consistent, fine-grained access control across its services, while also providing scalability, fault tolerance, and adaptability to changing access requirements [18].

Google Zanzibar has several advantages that contribute to its effectiveness in access control management within large-scale distributed environments. Firstly, it offers centralized access control, providing a unified platform for managing access control policies. This centralized approach enables consistent enforcement of access policies across distributed services and platforms, enhancing security and compliance. Secondly, Zanzibar is designed to scale seamlessly to handle millions of policy evaluations per second, catering to Google's vast ecosystem of services and users. This scalability ensures that access control remains efficient and responsive even as the organization grows. Additionally, Zanzibar employs a globally-distributed architecture with built-in fault tolerance mechanisms, ensuring high availability and reliability of access control services. This fault tolerance enhances system resilience and minimizes the risk of service disruptions. Lastly, Zanzibar enables fine-grained access control, allowing Google to define and enforce access policies based on diverse attributes, relationships and conditions. This granularity enables precise control over access permissions, enhancing security and enabling organizations to tailor access policies to specific requirements and use cases.

On the other hand, Zanzibar has several drawbacks for consideration. Firstly, Zanzibar is a proprietary system developed by Google for internal use, which limits its accessibility and interoperability with non-Google systems. This proprietary nature may hinder organizations seeking to integrate Zanzibar into their own infrastructure or collaborate with external partners. Secondly, external documentation and insights into Zanzibar's architecture and implementation details are limited, making it challenging for organizations to gain a comprehensive understanding of the system and its capabilities. This limited documentation may impede broader adoption outside of Google's ecosystem. Lastly, organizations leveraging Zanzibar may face vendor lock-in, as the system is tightly integrated with Google's internal infrastructure and services. This dependency on Google's ecosystem may restrict organizations' flexibility and ability to migrate to alternative solutions in the future. Therefore, organizations considering the adoption of Zanzibar should carefully evaluate these limitations and weigh them against the system's advantages.

Implementing a graph based on Google Zanzibar into the cloud environment adds complexity, frequently depending on hosted services. This reliance can raise concerns regarding latency and scalability, as dependencies on external services may impact system performance and introduce additional points of failure. Moreover, graph-based systems such as Zanzibar might encounter deployment challenges at the edge owing to their large size and non-local characteristics. These limitations could hinder their effectiveness in edge computing environments, where low latency and efficient access control are essential.

Compared to other policy management engines, Google Zanzibar distinguishes itself through its centralized, globally-distributed architecture tailored for large-scale, distributed services. While OPA offers flexibility and extensibility through its open-source nature, and AWS Cedar provides seamless integration with AWS services, Zanzibar excels in scalability, fault tolerance, and fine-grained access control tailored for Google's unique infrastructure and requirements.

Google Zanzibar represents a groundbreaking approach to access control, addressing the scalability and complexity challenges inherent in managing access policies for large-scale, distributed systems. While proprietary in nature, Zanzibar's design principles and architectural innovations offer valuable insights and lessons for the broader community in developing scalable and resilient access control solutions for modern computing environments. Many other authorization engines were inspired by Zanzibar, such as SpiceDB [19].

3.4. Integration with existing infrastructure

Integrating RBAC, ABAC, and ReBAC within existing IT infrastructures involves adapting these access control models to meet the complex requirements of modern technological environments. The study by M. Penelova (2021) [20] discusses how these models can be applied to enhance security in enterprise software systems. RBAC's role-based restrictions, ABAC's attribute-based flexibility, and ReBAC's relationship-oriented permissions each offer unique advantages depending on the specific

needs of an organization. The integration often requires careful planning to align with existing security policies and IT architecture while ensuring scalability and adaptability to future changes. Integration involves mapping organizational roles to access permissions, ensuring that these roles reflect actual job functions and access needs within the IT infrastructure. RBAC should be implemented with a central administration point that manages role assignments and permissions, facilitating updates and maintenance. For ABAC, the focus is on defining attributes (such as department, time of access, and type of data accessed) that control access decisions. Integration requires setting up a policy enforcement point that dynamically evaluates attributes against policies before granting access. This might involve modifying existing data schemas to include necessary attribute data. ReBAC requires identifying and managing relationships between entities (like user-to-user or user-to-data relationships) within the IT system. Integration can be complex, involving the creation of a relationship graph that must be maintained and referenced by the access control system. Existing applications might need to be extended to support these access control models or, in case it is not possible, a middleware application can be implemented to handle access control.

The survey by L. Golightly et al. (2023) [13] extends the discussion to distributed systems such as cloud services, blockchain, IoT, and Software Defined Networks (SDN). These environments pose additional challenges due to their decentralized nature and the dynamic interactions between their components. The study highlights how RBAC, ABAC, and ReBAC need to be adapted to manage access control efficiently in these settings, focusing on the need for models that can dynamically adjust to changes in user roles, attributes, and relationships without compromising the security or performance of the system. This could involve automated tools to monitor changes in the operational environment and adjust access controls accordingly. Additionally, in distributed systems, central management of access controls might not be feasible. Instead, a distributed approach where each node or service manages its own access controls based on synchronized policies could be more effective.

Furthermore, J. Park et al. (2021) [21] explore the application of these access control models in smart environments, where the integration of IoT and cloud computing introduces new dimensions of complexity. Their work discusses the principles of next-generation access control systems that not only secure but also enhance the functionality and user collaboration in smart infrastructures. The integration of RBAC, ABAC, and ReBAC in these systems is shown to be critical in managing diverse and dynamically changing access requirements while maintaining a high level of security. Access control systems need to be integrated with cloud-based services and IoT devices. This includes the ability to handle the high volume of access requests typical in these environments and ensuring that access control decisions are made swiftly to avoid delays in service. In smart environments, access control systems need to manage not only user access but also device interactions. This requires policies that can handle complex scenarios where devices act on behalf of users or interact with other devices autonomously.

3.5. Summary

After examining three prominent policy management engines, namely Google Zanzibar, Open Policy Agent (OPA), and AWS Cedar, each offers distinct features and functionalities tailored to specific use cases and organizational requirements.

Google Zanzibar stands out for its centralized access control system, designed for large-scale, distributed services. Its strengths lie in scalability and fault tolerance, making it ideal for organizations operating expansive, complex ecosystems where centralized access control is paramount. However, its proprietary nature and limited external documentation may pose challenges for organizations seeking interoperability and comprehensive understanding outside of Google's infrastructure.

Open Policy Agent (OPA) offers unparalleled flexibility and extensibility, empowering organizations to define and enforce access control policies across diverse cloud-native environments. Its open-source nature, coupled with a vibrant community, makes it suitable for organizations prioritizing flexibility and community-driven development. OPA excels in scenarios where fine-grained control and policy-as-code capabilities are essential, such as Kubernetes-based environments and microservices architectures.

AWS Cedar, recently released as an open-source solution by AWS, introduces a structured approach to IAM management and access control. It offers a seamless integration with AWS services and prioritizes readability and safety in policy enforcement. Cedar is well-suited for organizations heavily invested in the AWS ecosystem, particularly those requiring application-level authorization and compliance management within AWS environments. Having said that, it is not entirely AWS specific.

The project is currently available in open source as a collection of repositories and can be self-hosted using cedar-agent.

In summary, the choice of policy management engine should align with organizational requirements, system architecture, and cloud platform preferences as described in Table 1. Google Zanzibar is recommended for organizations with extensive, distributed infrastructures requiring centralized access control. OPA is suitable for organizations seeking flexibility, extensibility, and community support in managing access control policies across diverse cloud-native environments. AWS Cedar is ideal for organizations heavily reliant on AWS services, prioritizing readability and structured IAM management within AWS environments. Careful consideration of each engine's strengths and limitations is essential to selecting the most suitable solution for specific use cases and organizational needs.

Table 1
Policy management engines comparison table

Key Areas	Open Policy Agent	AWS Cedar	Google Zanzibar
Open Source	Yes	Yes	No (but alternative implementations are available)
Ability to self-host	Yes	Yes	No
Supported access control paradigms	RBAC, ABAC, ReBAC (with additional coding)	RBAC, ABAC, ReBAC	ReBAC
Language	Rego (Go-based)	Cedar (Rust-based)	ZCL
Authorization type	Policy as code (PAC)	Policy as code (PAC)	Graph-based
Scale	Scalable (with OPAL)	Scalable	Huge
Data volume	Lower	Lower	Higher
Performance	Higher	Higher	Lower compared to policy as code
Policy complexity	Higher	Higher	Lower
Deployment at edge	Supported	Supported	Infeasible due to size
Advantages	Policy as code, performance, adoption	Policy as code, application level policies, readability	Best fit for hierarchies and nested relationships, scale
Disadvantages	Learning curve, hard to use in production without OPAL	Limited tooling, community size, adoption	Cannot be deployed locally due to size, ReBAC only
Ease of Updates	Highly flexible (with OPAL)	Highly flexible	Less flexible
Deployment	Standalone	Standalone or as part of AWS services	Not available as a standalone deployment or service, although open source implementations exist

4. AI Applications in access control space

The application of Artificial Intelligence in the access control space holds significant promise for revolutionizing how organizations manage and enforce access policies within their digital ecosystems. Traditional access control mechanisms, while effective, often struggle to adapt to the increasingly dynamic and complex nature of modern computing environments. As organizations embrace cloud computing, microservices architectures, and distributed systems, the need for adaptive, intelligent access control solutions becomes more pronounced. AI can be leveraged in various ways for access control and policy management.

Some potential use cases include:

1. User and Entity Behavior Analytics (UEBA): AI can be used to establish baseline behavior patterns for users, devices, and applications within an organization. By continuously monitoring and analyzing activity logs, AI algorithms can detect anomalous behavior that deviates from the established baselines, potentially indicating unauthorized access attempts or policy violations. This can help identify insider threats, compromised accounts, or malicious activities.
2. Automated Policy Management: AI can assist in the creation, review, and maintenance of access control policies. It can analyze existing policies, identify inconsistencies, redundancies, or gaps, and suggest improvements. AI can also be used to automatically generate and update policies based on organizational requirements, industry regulations, and best practices.
3. Intelligent Access Governance: AI can support access governance by continuously evaluating user access privileges and entitlements against defined policies. It can identify excessive or unnecessary permissions, recommend access revocations or modifications, and automate the review and certification processes for user access rights.
4. Risk Analysis and Mitigation: AI can assess the potential risks associated with granting or denying access to specific resources or systems. By analyzing various factors such as user roles, resource sensitivity, threat intelligence, and historical data, AI can provide risk scores and recommendations to help organizations make informed access control decisions.
5. DLBAC (Deep Learning Based Access Control) implementation can augment, complement and in future possibly even replace the currently used access control approaches in large scale, complex and dynamic environments [22]. It involves a neural network that adapts to the environment and context and reduces the effort spent on policy engineering, changes and other related maintenance activities.
6. Continuous Monitoring and Adaptation: AI can continuously monitor changes in the organization's infrastructure, applications, and user population. It can proactively identify potential policy violations or access control issues and suggest appropriate remediation measures or policy updates to ensure ongoing compliance and security.
7. Intelligent Multi-Factor Authentication (MFA): AI can be used to enhance MFA mechanisms by analyzing contextual factors such as user behavior, device characteristics, location, and risk indicators. It can dynamically adjust the authentication requirements or challenge types based on the perceived risk level, providing an additional layer of adaptive access control.
8. Natural Language Processing (NLP) for Policy Interpretation: AI-powered NLP can assist in interpreting and understanding access control policies written in natural language. It can translate policies into machine-readable formats, identify ambiguities or inconsistencies, and facilitate better understanding and enforcement of policies across the organization.

The integration of AI technologies into access control systems has the potential to enhance security, streamline access management processes, and improve user experience in today's increasingly interconnected and dynamic digital landscape. By addressing challenges such as dynamic policy management, anomaly detection, and adaptive access control, AI-driven access control solutions can help organizations stay ahead of evolving security threats and compliance requirements.

5. Conclusions

In this exploratory survey, we delved into various access control paradigms and policy management engines, aiming to provide insights into their functionalities, strengths, and limitations. Through the examination of Role-Based Access Control (RBAC), Attribute-Based Access Control (ABAC), and Relationship-Based Access Control (ReBAC), alongside major policy management engines such as Open Policy Agent (OPA), AWS Cedar, and Google Zanzibar, several key observations and considerations have emerged.

Firstly, it is evident that access control paradigms have evolved significantly to address the diverse and dynamic requirements of modern computing environments. While RBAC offers simplicity and scalability, ABAC provides flexibility and granularity, and ReBAC introduces a nuanced perspective by considering relationships between entities within an organization. Each paradigm has its own set of strengths and limitations, necessitating careful consideration of organizational needs and use case requirements in selecting the most suitable approach.

Furthermore, policy management engines play a pivotal role in facilitating the implementation and enforcement of access control policies across distributed and heterogeneous IT environments. Open

Policy Agent (OPA), AWS Cedar, and Google Zanzibar offer distinct approaches to policy management, catering to different deployment scenarios and integration requirements. While OPA provides an open-source, flexible solution with community support, AWS Cedar and Google Zanzibar offer managed services tailored for integration with their respective cloud platforms.

In the realm of authorization systems, there exists universal solution to cater to the diverse range of scenarios. The decision to employ graph-based systems, such as Google Zanzibar or SpiceDB, or policy-as-code systems like Open Policy Agent or AWS Cedar, is not a matter of selecting a technologically superior approach. Instead, it hinges on aligning the chosen system with the specific requirements, scale, and complexity of the environment in question.

The choice of access control approach and policy management engine should be guided by factors such as organizational requirements, system architecture, and compliance considerations. Organizations must carefully assess the trade-offs between simplicity, flexibility, scalability, and vendor lock-in when selecting access control mechanisms and policy management solutions.

Graph-based systems demonstrate proficiency in managing voluminous data sets and effectively representing intricate relationships within large-scale applications. Conversely, policy-as-code systems offer a high degree of flexibility, facilitating seamless policy updates and exhibiting adeptness in handling complex policy structures. Each paradigm possesses its unique strengths and limitations, necessitating a judicious evaluation of system requirements and a careful weighing of trade-offs.

Moreover, it is imperative to acknowledge the fact that these systems can complement each other. By combining their respective capabilities, a hybrid system can be crafted, capitalizing on the strengths of both approaches and thereby addressing the diverse authorization needs that may arise. Ultimately, the overarching objective remains the effective control of access to services, ensuring data privacy, and upholding system integrity, irrespective of the chosen authorization strategy.

Areas for further research and experimentation include exploring the intersection of access control paradigms with emerging technologies such as blockchain, AI, and zero-trust security architectures, investigating the impact of dynamic policy enforcement mechanisms on system performance and scalability, and exploring novel approaches to policy modeling and enforcement in distributed and decentralized computing environments.

Future research could focus on the development of AI-driven models for predictive access control, where machine learning algorithms could predict and prevent unauthorized access based on behavior patterns and anomaly detection. This approach could significantly improve security in real-time applications and large-scale systems by allowing more proactive management of access permissions. Investigating the integration of AI to automate and optimize the management of complex access control policies would also be valuable, particularly in environments where access requirements are continually changing.

Blockchain technology could be explored further for its potential to facilitate decentralized access control mechanisms, particularly in distributed environments like the Internet of Things (IoT) and cloud computing. The immutable and transparent nature of blockchain could help in creating more secure and resistant access control systems that are less susceptible to tampering and insider threats. Research could also look into hybrid models that integrate blockchain with traditional access control systems to enhance trust and security in multi-party transactions or operations.

There is also a compelling case for advancing policy management engines to better handle the complexity and scale of modern IT environments. Enhancing the capability of these systems to dynamically adjust policies based on real-time data and contextual information could address many of the current limitations in scalability and flexibility. Additionally, exploring ways to simplify policy management while maintaining robust security controls in highly dynamic environments would be beneficial.

In conclusion, as the landscape of access control continues to evolve in response to emerging technologies and evolving threat landscapes, ongoing research and experimentation are essential to develop robust, adaptable, and scalable access control mechanisms that meet the evolving needs of organizations in an increasingly interconnected and dynamic digital ecosystem.

References

[1] D. Zhuravchak, T. Ustyianovych, V. Dudykevych, B. Venny and K. Ruda, Ransomware Prevention System Design based on File Symbolic Linking Honeypots, in: Proceedings of the 2021 11th IEEE International Conference on Intelligent Data Acquisition and Advanced

Computing Systems: Technology and Applications (IDAACS), IEEE, Cracow, Poland, 2021, pp. 284-287, doi: 10.1109/IDAACS53288.2021.9660913.
[2] D. Zhuravchak, P. Hlushchenko, M. Opanovych, V. Dudykevych, A. Piskozub, Zero Trust Concept For Active Directory Protection To Detect Ransomware, Cybersecurity: Education, Science, Technique, vol.2, no.22, pp.179-190, 2023.
[3] R. S. Sandhu, Role-based Access Control, in: M. Zelkowitz (Ed.), Advances in Computers, volume 46, Elsevier, College Park, Maryland, 1998, pp. 237-286. doi:10.1016/S0065-2458(08)60206-5.
[4] A. Elliott, S. Knight, Role explosion: Acknowledging the problem, in: Proceedings of Software Engineering research and practice, WORLDCOMP '10a, Las Vegas, NV, 2010, pp. 349-355.
[5] A. S. M. Kayes, R. Kalaria, I. H. Sarker, M. S. Islam, P. A. Watters, A. Ng, I. Kumara, A survey of context-aware access control mechanisms for cloud and fog networks: Taxonomy and open research issues. Sensors 20 (2020). doi: 10.3390/s20092464
[6] V. C. Hu, D. R. Kuhn, D. F. Ferraiolo, J. Voas, Attribute-Based Access Control, Computer 48(2) (2015) 85-88. doi: 10.1109/MC.2015.33.
[7] Hubli, S. C., Potdar, G. P. Security evolution in object-oriented software systems: shaping the future landscape through innovative patterns, IRJMETS 6(1) (2024). doi: 0.56726/IRJMETS48793.
[8] G. Goyal, P. Liu, S. Sural, Securing Smart Home IoT Systems with Attribute-Based Access Control, in: Proceedings of the 2022 ACM Workshop on Secure and Trustworthy Cyber-Physical Systems, Sat-CPS '22, Association for Computing Machinery, New York, NY, 2022. pp. 37–46. doi:10.1145/3510547.3517920
[9] S. Parkinson, S. Khan, A Survey on Empirical Security Analysis of Access-control Systems: A Real-world Perspective, ACM Computing Surveys 55 (2022) 1-28. doi:10.1145/3533703.
[10] J. A. Khan, Role-Based access Control (RBAC) and Attribute-Based Access Control (ABAC), in: P. Goel, H. Pandey, A. Singhal, & S. Agarwal (Eds.), Improving Security, Privacy, and Trust in Cloud Computing, 1st ed, IGI Global, Hershey, PA, pp. 113-126. doi:10.4018/979-8-3693-1431-9.ch005
[11] P.W.L. Fong, Relationship-based access control: protection model and policy language, in: Proceedings of the first ACM conference on Data and application security and privacy, CODASPY '11, Association for Computing Machinery, New York, NY, 191–202. doi:10.1145/1943513.1943539
[12] C. Arora, Higher-Order (Temporal) Relationship-Based Access Control, Master's thesis, University of Calgary, Calgary, Canada, 2022.
[13] L. Golightly, P. Modesti, R. Garcia, V. Chang, Securing distributed systems: A survey on access control techniques for cloud, blockchain, IoT and SDN, Cyber Security and Applications (2023). doi:10.1016/j.csa.2023.100015
[14] Daniel Bass, An intro to Open Policy Agent (OPA), 2022. URL: https://www.permit.io/blog/introduction-to-opa
[15] Daniel Bass, Real-time dynamic authorization - an introduction to OPAL, 2022. URL: https://www.permit.io/blog/introduction-to-opal
[16] J. Cutler, et al. Cedar: A New Language for Expressive, Fast, Safe, and Analyzable Authorization (Extended Version), arXiv preprint, 2024. doi:10.48550/arXiv.2403.04651.
[17] Mike Hicks, How we built Cedar with automated reasoning and differential testing, 2023. URL: https://www.amazon.science/blog/how-we-built-cedar-with-automated-reasoning-and-differential-testing
[18] R. Pang, R. Caceres, M. Burrows, Z. Chen, P. Dave, N. Germer, A. Golynski, K. Graney, N. Kang, L. Kissner, J. L. Korn, A. Parmar, C. D. Richards, M. Wang, Zanzibar: Google's Consistent, Global Authorization System, in: Proceedings of the 2019 USENIX Annual Technical Conference, USENIX ATC 19, USENIX Association, Renton, WA, 2019, pp. 33-46. isbn:978-1-939133-03-8
[19] Jake Moshenko, Google Zanzibar Open Source: The Architecture of SpiceDB, 2021. URL: https://authzed.com/blog/spicedb-architecture
[20] M. Penelova,Access Control Models, Cybernetics and Information Technologies 21(4) (2021) 77-104. doi:10.2478/cait-2021-0044
[21] J. Park, R. Sandhu, M. Gupta and S. Bhatt, Activity Control Design Principles: Next Generation Access Control for Smart and Collaborative Systems, IEEE Access 9 (2021), 151004-151022, doi: 10.1109/ACCESS.2021.3126201.

[22] M. N. Nobi, R. Krishnan, Y. Huang, M. Shakarami, R. Sandhu, Toward Deep Learning Based Access Control, in: Proceedings of the Twelfth ACM Conference on Data and Application Security and Privacy, CODASPY '22, Association for Computing Machinery, New York, NY, 2022, pp. 143–154. doi:10.1145/3508398.3511497.