DTW-ABAC: A DYNAMIC TRUST WEIGHTED-ATTRIBUTE BASED ACCESS CONTROL HYBRID SECURITY MODEL FOR CLOUD APPLICATIONS

by

Saurabh Kulkarni

B.E., Savitribai Phule Pune University, 2017
Diploma., MSBTE, 2014

THESIS SUBMITTED IN PARTIAL FULFILMENT OF
THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE
IN
COMPUTER SCIENCE

UNIVERSITY OF NORTHERN BRITISH COLUMBIA

September 2025

© Saurabh Kulkarni, 2025

ABSTRACT

Modern digital infrastructures require access control systems that protect sensitive data as well as adapt to evolving contexts and user behaviour. While foundational models like Discretionary Access Control (DAC), Mandatory Access Control (MAC), and Role-Based Access Control (RBAC) provide basic enforcement, they lack flexibility, granularity and real-time responsiveness. Attribute-Based Access Control (ABAC) improves granularity by using attribute-driven policies, but standard implementations (XACML and NGAC) each have critical limitations. XACML, though powerful in static policy expression, lacks real-time contextual awareness, while NGAC offers dynamic evaluation but struggles with policy standardization, over-permissiveness and transparency. To bridge these gaps, this research proposes DTW-ABAC (Dynamic Trust Weighted-Attribute Based Access Control), a hybrid framework that combines XACML's structured policy logic with NGAC's dynamic evaluation capabilities. The framework leverages Microsoft Entra ID for consistent and secure identity and attribute management, and introduces a trust scoring system that adjusts user access based on behavioural consistency and historical risk. Weighted attribute evaluation ensures policy flexibility, while scenario-driven testing and detailed audit logs increase transparency and accountability. Comparative analysis shows that the hybrid model delivers more accurate, adaptive, and explainable decisions than standalone XACML or NGAC, making it a strong candidate for enterprise and cloud-scale deployment where contextual nuance and high security reliability are essential.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

GLOSSARY

- **ABAC (Attribute-Based access control):** The access control method utilizes the subject attributes, object attributes and environmental attributes to determine whether a subject can perform any action on the requested object. It is a superset of RBAC because roles are just one type of attribute in ABAC.
- **Access control (AC):** "The process of granting or denying specific requests to 1) obtain and use the information and related information processing services and 2) enter specific physical facilities (e.g., federal buildings, military establishments [1])
- **Access control mechanism (ACM):** A logical component that receives access requests, evaluates access based on the defined architecture and enforces decisions.
- **Attributes:** The characteristics of the object, such as file name, application name, application ID, security clearance, etc., or subject, such as username, user ID, job location, associated project name, etc. or environmental, such as time of request, geographical location, etc. Usually, these are defined by security systems.
- **Authentication**: The process of verifying a subject's identity by relying on one or more factors [2][3], such as**:**
    - **Something you know** – A secret such as a password, PIN, or passphrase.
    - **Something you have** – A physical device, such as a smart card, security token, or mobile authenticator.
    - **Something you are** – Biometric characteristics (e.g., fingerprint, retina scan, or facial recognition).
    - **Something you do** – Behavioural biometrics, such as keystroke dynamics, typing speed, or gait recognition.

- **Somewhere you are** – Location-based authentication using IP address, GPS, or network context.

Additionally, static authentication typically occurs at the initial stage of the access control process (e.g. at login), whereas continuous authentication happens throughout the session (e.g., access and behavioural patterns which result in risk/trust).

- **Authorization:** The process of determining and granting specific permissions or access rights to a subject for one or more objects, ensuring that the subject can only access resources they are authorized to use with the approved level of permissions (e.g., Read, Write, etc.).
- **Cloud computing:** The delivery of computing services, such as servers, storage, databases, networking, software, analytics, and intelligence, over the Internet.
- **Evaluation**: The final result after evaluating the access request by the AC model.
  - **Positive result**: The access request has been granted. From the context of this thesis:
    - **TP (True Positives)** - permitted access for legitimate users. It reduces unnecessary alerts.
    - **FP (False Positives)** - permitted access for unauthorized users. It can lead to a security breach.
  - **Negative result**: The access request has been denied. From the context of this thesis:
    - **TN (True Negatives)** - denied access to unauthorized users. It improves the security and access management process.
    - **FN (False Negatives)** - denied access to legitimate users. Erroneously preventing users from doing their work.

- **Firewall:** A security mechanism designed to regulate network traffic by monitoring data transmissions and determining whether to permit or restrict access based on predefined security policies.
- **IaaS (Infrastructure-as-a-Service):** Provides virtualized computing resources such as servers, storage, and networking on a pay-as-you-go basis.
- **Microsoft Azure Cloud:** A subscription-based cloud computing solution provided by Microsoft that is accessible via the internet dashboard. This is an IaaS and PaaS solution, but not SaaS.
- **Microsoft Entra ID (or Entra ID)** [4]**:** A cloud-based identity and access management service to manage Microsoft PaaS and SaaS applications. Certain features are license-based.
- **Multi-factor Authentication (MFA):** A process of authenticating a user's identity using two or more verification factors (mentioned in the authentication in glossary) to ensure a strict identity. For example, users must enter login credentials (something you know) and a one-time password (OTP – something you have) received on mobile phones, etc.
- **Object:** A logical entity or system resources such as devices, files, records, tables, processes, programs, networks, or domains containing or receiving information.
- **PaaS (Platform-as-a-Service):** Offers a managed environment with development tools, frameworks, and infrastructure to build, test, and deploy applications.
- **Performance metrics:** To assess how accurately and effectively a system makes decisions, such as granting or denying access.
    - **Accuracy**: Percentage of total predictions that are correct (TP or TN).

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

- **Precision**: Percent of permit accesses that are correct. If precision is low, legitimate users will be denied access and unable to do their work.

$$Precision = \frac{TP}{TP + FP}$$

- **Recall** (Sensitivity or True Positive Rate): It shows how well the model identifies scenarios where access should be granted. if recall is low, this means the system will grant access to illegitimate users, such as attackers.

$$Recall = \frac{TP}{TP + FN}$$

- **F1-Score or Harmonic mean**: Used when both precision and recall are important, balancing FP and FN, especially critical in access control where both denial of legitimate users and access to unauthorized users are risky.

$$F1 - Score = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

- **RBAC (Role-Based Access Control):** The access control method to restrict access to resources based on roles, a logical identity with a set of permissions, and associate it with the users.
- **SaaS (Software-as-a-Service):** Delivers fully functional applications over the internet, eliminating the user's need for installation or maintenance.
- **Subject:** a person or a system process, also known as a non-person entity (NPE), requesting access to the files, devices or applications. Also known as the requester.
- **Usage Frequency**: The count of how many times an attribute is used in the XACML-defined policies and NGAC graphs.

NOTATIONS

- **XACML Notations**
  - TF_{XACML} : XACML trust factor. (range: 0 to 100) Note: calculated using a formula.
  - C_{XACML} : XACML weight constant. (range: 0 to 1), assigned to the XACML trust factor to calculate the final trust factor. Note: assigned value derived from experiments.
- **NGAC Notations**
  - $TF_{NGAC_{Base}}$: NGAC Trust factor (base) without the confidence factor. Note: calculated using a formula.
  - H : Confidence factor. (range: 1 to 2) Used to adjust the NGAC trust factor (base) according to historical success/failure patterns. Note: calculated using a formula.
  - TF_{NGAC} : Final NGAC Trust factor. (range: 0 to 100) Note: calculated using a formula.
  - C_{NGAC} : NGAC weight constant, assigned to the NGAC trust factor to calculate the final trust factor. (range: 0 to 1) Note: assigned value derived from experiments.
- **Common Notations**
  - W_i: Weight associated with the attribute i. Note: assigned value derived from experiments.
  - M_i: Multiplication factor for the attribute i. Either 1 if the attribute value is matched with values defined in policies (i.e. incoming attribute value is equal to or in range of expected attribute value or range), or 0 if not matched.
  - TF: Final trust factor to decide the result. (range: 0 to 100) Note: calculated using a formula.

$$TF = \frac{C_{XACML} \times TF_{XACML} + C_{NGAC} \times TF_{NGAC}}{C_{XACML} + C_{NGAC}} \times 100$$

ACKNOWLEDGEMENT

I would like to express my sincere and heartfelt gratitude to my supervisors, Dr. Waqar Haque and Dr. Andreas Hirt, for their invaluable guidance, technical expertise, and unwavering support throughout the duration of this research. Their mentorship was pivotal in shaping the conceptual and methodological foundation of this work and refining its execution and presentation. Their timely feedback, encouragement, and willingness to engage deeply with both theoretical and practical aspects greatly enriched the quality of this thesis.

I am also thankful to the committee member, Dr. Peter Jackson, for their thoughtful insights and constructive feedback. Their engagement and perspective helped strengthen the rigour and relevance of this work.

I wish to acknowledge the faculty and staff of the Department of Computer Science, UNBC, for providing a collaborative and intellectually stimulating environment that supported the successful completion of this research. I also thank my peers and colleagues for the many helpful discussions, shared learning experiences, and moral support.

Lastly, I am deeply grateful to my family and friends for their constant encouragement, patience, and belief in me throughout this journey.

Chapter 1

Introduction

A relentless pursuit of information accessibility and operational agility characterizes the contemporary business landscape. Over the last few decades, data and applications have experienced tremendous growth in different sectors such as information technology, finance, healthcare and governance. Enterprises use several information and communication technologies (ICT), such as company-wide private networks, industry-specific software, regular information exchange over the Internet or cloud computing. Cloud computing has played a significant role in this digital transformation due to its remote access features, scaling capability, security and pay-as-you-go cost model. A 2023 survey published by Statistics Canada [5] shows that cloud computing has become the most commonly adopted ICT, as depicted in Figure 1.

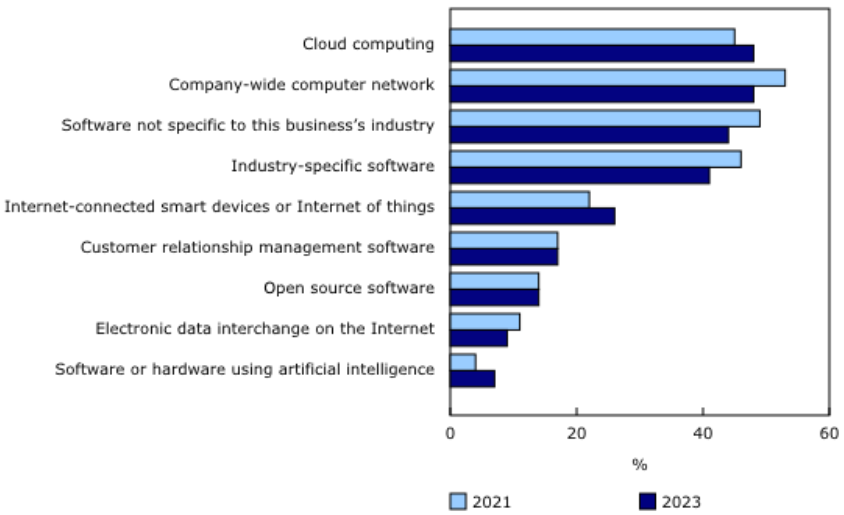


Figure 1 Information and communication technologies most commonly used by businesses, Canada, 2021 and 2023 [5]

In 2023, cloud computing reached a usage rate of 48%. Particularly, businesses in the information and cultural industries sector had the highest adoption rate, with 81% using cloud computing in 2023.

In traditional perimeter-based security models, companies protected their digital assets by keeping them behind firewalls and inside secure data centers. Cloud computing changes this by moving assets to internet-accessible services, especially in public cloud environments. Because of this shift, businesses can no longer rely only on internal network security and must rethink how they protect their systems. As companies spread their workloads across different cloud platforms, security is no longer just about firewalls. Instead, protection is now based on identity-based security, often described as "identity is the new firewall." This makes authentication and access management more important than ever, as they now play a key role in securing cloud environments. To further enhance security, Intrusion Detection System (IDS) and Intrusion Prevention System (IPS) technologies can be integrated alongside authentication and access management, providing real-time detection and automatic prevention of suspicious activities and potential breaches.

With this growth, cyber-attacks have also increased, involving malicious traffic from untrusted sources, network attacks, unauthorized access and insider threats to enterprise resources and services [6]. Strong security systems are required to counter those attacks and provide access to only legitimate users. For decades, firewalls have been a crucial safeguard in protecting networks from unauthorized access and potential threats. However, the firewall cannot handle unauthorized access to data or insider threats, primarily carried out by compromised user/admin

accounts. Privileged Access Management (PAM) is a security solution that monitors and controls access to critical resources. It limits administrative privileges, tracks user activity during privileged sessions, and adds layers of protection to reduce the risk of unauthorized access and data breaches. A report published by Netwrix Research Labs, recognized as a visionary in Gartner's 2023 Magic Quadrant for PAM [7], highlighted the most common security incidents, with attacks linked to cloud account compromise ranked in second position (Figure 2) [8]. In 2020, just 16% of respondents reported this kind of cloud incidence; by 2024, that number had risen to 55%.

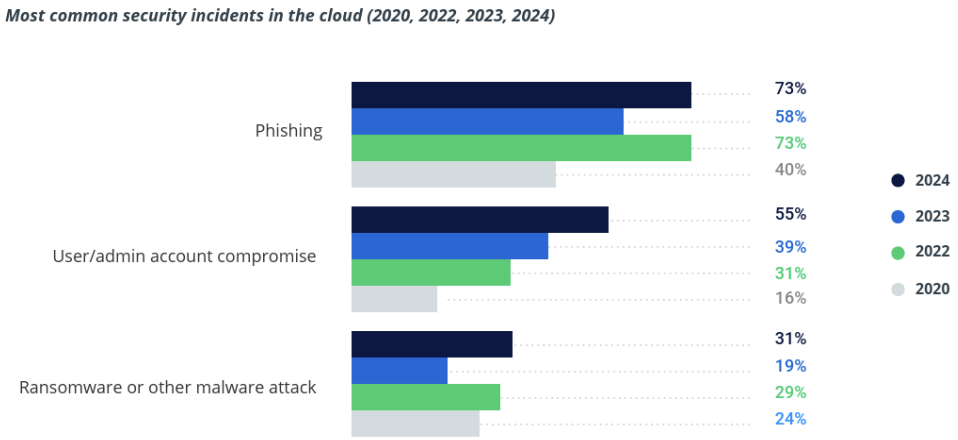


Figure 2 Most common security incidents in the cloud (2020, 2022, 2023, 2024) [8]

The same report also compares the security incidents in on-premises and cloud-based infrastructures in the healthcare sector (Figure 3). The statistics reveal that the user account compromise in the cloud environment is higher than on-premises, as well as other most common security incidents. The report further links this to contributing factors, including remote access to patient records, shared documents, weak authentication and misconfiguration of access management policies. Since cloud computing is a shared security model [4], both the cloud vendor and organizations are responsible for security and compliance.

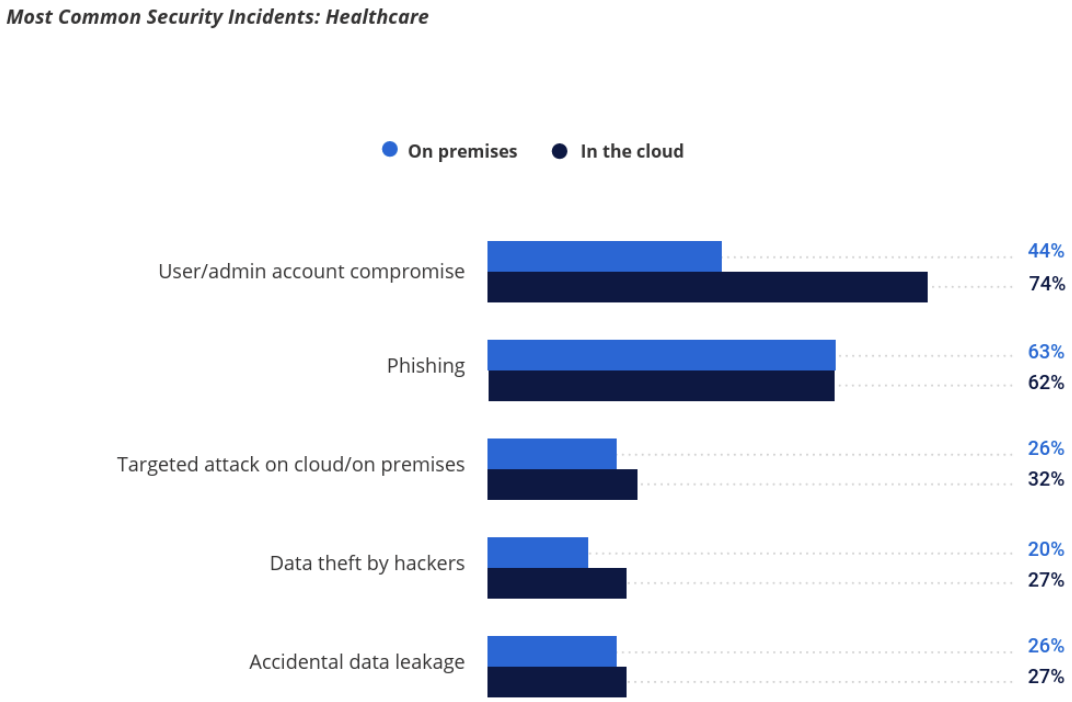


Figure 3 Most common security incidents: Healthcare [8]

1.1. Access control (AC) levels

The compromised user identity with legitimate access can still exploit the system, and this is where the access control (AC) mechanisms (in the context of logical operations) play a crucial role. AC works as the background process when the user initiates an access request, and it fundamentally ensures that only a legitimate user can access the authorized resources. In addition to the authentication and authorization process, AC performs access enforcement, logging events or activities, session management, and adaptive security responses such as multi-factor authentication (MFA), temporary account disablement, password resets, or escalation to a security help desk. Also, it is necessary to distinguish between static access control and continuous access evaluation, especially in modern cloud services and hybrid work environments.

1) Static Access Control determines permissions at the start of a session and retains them until the session ends. This is common in traditional systems but lacks adaptability to real-time threats.
2) Continuous Access Evaluation (CAE), on the other hand, continuously monitors access during a session and can adapt decisions based on changing factors, like geolocation, device status, risk scores, or user behaviour [9]. CAE enables real-time enforcement, such as revoking access mid-session if a risk is detected.

Context in Access Control refers to environmental or situational information that influences access decisions. It answers questions about when (such as time of day or day of week), where (like IP address or physical location), how (including device type or encryption), and under what conditions (such as risk score, threat level, or compliance requirements). By evaluating these dynamic attributes, policies can adapt to changing conditions and enforce more precise security controls. This shift toward dynamic, context-aware access control is especially important as identity threats and session hijacking increase in frequency. These mechanisms can be applied at various points and levels in the system, from application-specific controls to enterprise-wide policy layers. AC can be applied to multiple places and different levels, such as:

1) System-level access control [10]: Restricting access to the operating system or computing environment through login permissions or administrator access control. It deals mainly with the authentication process of verifying identity using credentials, one-time passwords (OTP) or Multi-Factor authentication (MFA) to enter the system.

2) Data level access control [11]: Restricting access to files, databases and structured/ unstructured data. It ensures that only authorized users can view, edit or delete the data using

file permissions, policies and encryption methods. A set of rules known as an Access Control List (ACL) is associated with each logical object that defines which users or system processes are allowed to access a specific object. It is most commonly used in security situations, where an object owner allocates access permissions.

3) Application-level access control [12]: Governs access to multiple applications within a specific infrastructure, determining the features and functionalities users can utilize to perform certain operations. It typically operates through a role-based access matrix, where permissions or privileges are assigned to roles rather than directly to individual users, improving scalability and manageability. Each role aggregates a set of access rights, and users are mapped to roles based on their responsibilities. Systems such as Customer Relationship Management (CRM) software use role-based permissions to provide controlled access at this level [13]. Access rules differentiate users (e.g., administrators, managers, customer service representatives) by assigning them predefined roles with distinct privilege sets. For instance, a sales executive may only view customer contact and deal progression data in a CRM environment. In contrast, administrative roles may allow configuration of system-wide settings. However, in modern zero trust models, even administrators typically do not have unrestricted access to sensitive data by default; instead, elevated access is granted only through controlled, time-limited processes that require approval and generate audit logs to maintain security and compliance.

4) Network-level access control [14]: Determines whether a user or machine can be connected to a specific network and what resources they can access. It prevents unauthorized network access and manages incoming and outgoing network activity. Usually, firewalls or virtual

private networks (VPN) are used to establish secure, encrypted tunnels over the Internet. Traditional VPNs are commonly used to connect remote workers securely to an organization's internal network (intranet). Additionally, site-to-site VPNs have become increasingly popular to securely connect multiple office locations over public networks, allowing them to operate as a unified private network across geographically distributed sites [15].

5) Cloud-based and IoT-level access control [16] [17]: This emerging solution controls how users interact with IoT devices and cloud-based resources, including data, applications and networks. This is a custom-designed access control mechanism specific to the vendor to manage access to their cloud services by internal or external users of the client organization. It involves both the authentication and authorization process.

1.2. Access control (AC) traditional models

Numerous models have been developed to manage access control for data and applications more efficiently. The traditional access control models are as follows [18]:

1.2.1 Discretionary Access Control (DAC)

An access control model where the owner of a resource has the authority to grant or restrict access to other users based on defined permissions. Access permissions, such as read, write, and execute, are assigned at the owner's discretion rather than enforced by a central security policy. However, this type of access control comes with some security weaknesses, as follows [18].

(1) Transitive Access: A user who is granted read access can copy and share the information with others.

(2) Trojan Horse Attacks: Malicious programs can inherit user permissions. This can lead to unauthorized actions under the user's identity [19].

(3) Lack of Centralized Security Policy: Since individual owners control access, it may not align with organization-wide security policies. Standardization is not possible across the organization and can create compliance issues.

1.2.2 Mandatory access control (MAC): MAC is a strict security model where access decisions are enforced by a centralized authority based on predefined security policies. Unlike discretionary models, individual users, including object owners, have no control over who can access resources. Instead, both users and data objects are assigned security labels, such as classification levels (e.g., Top Secret, Secret, Confidential), which are then used to enforce access. This model is widely used in environments requiring high assurance, such as military and government systems, where maintaining strict confidentiality and data integrity is critical. MAC operates on key principles:

(1) Clearance-based access: Users must possess appropriate security clearance to access classified data.

(2) "No read up, no write down": A user cannot read data above their clearance level (preventing unauthorized disclosure) nor write data to a lower classification level (preventing data leaks), known as the Bell-LaPadula model [20].

1.2.3 Role-based access control (RBAC): It involves creating multiple roles that contain a logical identity associated with a set of permissions to resources, and then those roles can be assigned to the users (Figure 4). This model differs significantly from the

DAC and MAC in terms of who controls the access provision and how access decisions are made. It is more structured than DAC since access is determined by roles rather than individual users controlling it. Similarly, it is more flexible than MAC since the roles can be adjusted without changing the entire security policy.

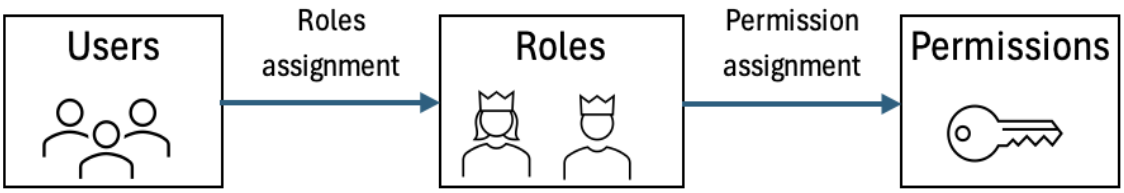


Figure 4 Role-based access control components

However, despite being flexible and structured, it lacks several essential features as described below:

(1) It has to deal with the "stronger security vs easier administration" problem [10]. To maintain stronger security, the roles need to be more granular, which requires creating as many roles as possible per user, which can create challenges in the administration. This can also lead to compliance and regulatory hurdles and raise the challenge of balancing security with operational costs.

(2) The rapid growth of web-based applications, which rely on interconnecting multiple components over the internet to deliver cohesive services, has significantly increased the complexity of access control. In such dynamic environments, RBAC often struggles to adapt due to frequent changes in roles and permissions, making its implementation challenging and, at times, impractical [21].

1.2.4 Attribute-based access control (ABAC): Among all traditional models, ABAC is more evolved and flexible and provides more granularity. It revolves around the attributes, policies and permissions. The attributes are subject (user) attributes, object attributes (application) attributes and environment attributes. Figure 5 shows the simplified version of how ABAC connects those attributes with policies and assigns permissions instead of predefined roles. Although the administrator controls the access policies, the attributes or tags for each user and registered application are typically not manually configured. Instead, they are automatically populated from upstream systems such as Human Capital Management Systems (HCMS) (e.g., job title, department, location). The resulting policy consists of a set of rules that define the required attributes and values to access the specific resource. The AC mechanism evaluates the attributes against the rules and enforces the policy. ABAC is a superset of RBAC since the roles can be defined as attributes in the user profiles or application attributes. It can replace RBAC and add more features to the ACM if needed.

While ABAC offers greater flexibility, granularity, and dynamic security control, its implementation can lead to complex policies and time-intensive processes. Various approaches exist for implementing ABAC, each with its own advantages and limitations. Ongoing advancements aim to simplify ABAC frameworks, making them more efficient and easier to adopt.

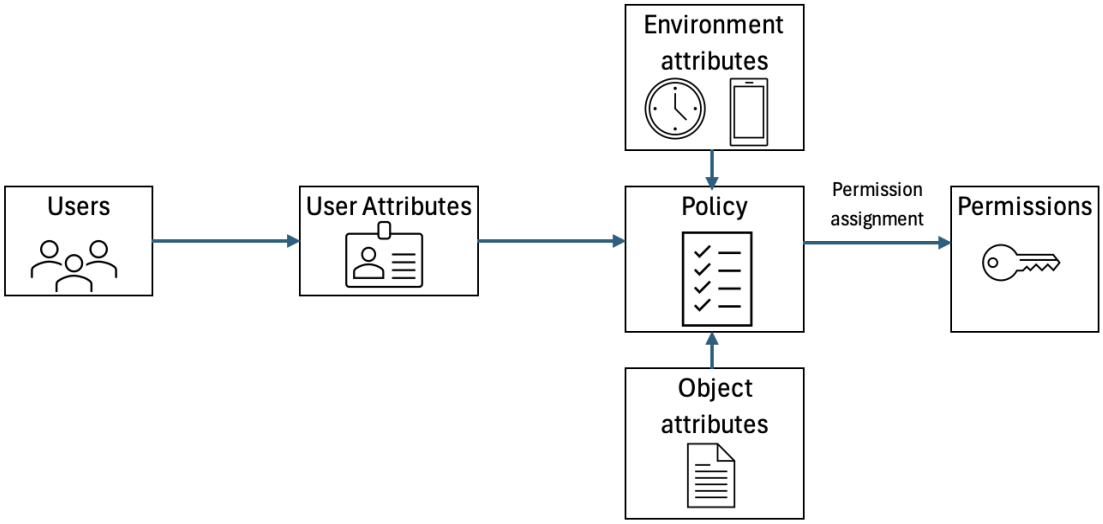


Figure 5 Attribute-based access control components

1.3. Comparison of Traditional Access Control Models across System-Level Metrics

Table 1 compares these traditional AC models over metrics such as flexibility, granularity, security level, ease of implementation, scalability, dynamic adaptation and use case suitability [22], [23].

Metric	**DAC**	**MAC**	**RBAC**	**ABAC**
Flexibility	High	Low	Moderate	Very High
Security Level	Low	Very High	Moderate	Very High
Granularity	Coarse	Coarse	Moderate	Fine-Grained
Ease of Implementation	Simple	Complex	Moderate	Complex
Scalability	Low	Low	High	Very High
Dynamic Adaptability	Limited	None	Limited	High
Use Case Suitability	Small Teams/ Personal	Government/ Military	Enterprises/ Organizations	Cloud/IoT/ Dynamic Systems

Table 1 Comparison of DAC, MAC, RBAC and ABAC models over system-level metrics

These metrics are described below:

i) **Flexibility:** It defines how the model can adapt to new scenarios and requirements as the environment changes. ABAC takes the edge here because it allows dynamic AC using multiple attributes.

   (a) For example, the AC can grant access to a resource only if the user with a specific role accesses it from a secure device during business hours. The attributes include user role, location, device type, and time. ABAC adapts to real-time conditions, making it ideal for complex, evolving systems like cloud environments and IoT.

ii) **Security level:** The level of security enforced by the access control (AC) model. ABAC enforces fine-grained policies using multiple attributes. This ensures the principle of least privilege and reduces the risks of being over-permitted.

   (a) For example, users may access a database only if their risk score is low and their department matches the resource sensitivity level. This gives tighter control due to the consideration of environmental and user-specific factors. Additionally, it minimizes unauthorized access risks.

iii) **Granularity:** It shows how detailed and precise AC is in specifying who can do what under specific conditions. ABAC comes with fine-grained access control as it evaluates multiple levels of attributes. Instead of only allowing the “developer to access the code files,” ABAC can enforce the rule that the “developer can edit sensitive code only when they are part of the specific project in the office network

and using a secure device." This level of control supports complex security needs and compliance with British Columbia's privacy legislation, including:

(1) The Personal Information Protection Act (PIPA) [24] governs the collection, use, and disclosure of personal information by private sector organizations in BC.

(2) The Freedom of Information and Protection of Privacy Act (FIPPA) [25] regulates how public sector organizations handle personal data and ensure security controls to prevent unauthorized access.

   (a) By implementing ABAC, organizations in BC can ensure compliance with these laws by enforcing strict, attribute-based access policies that align with data protection and privacy requirements.

iv) **Ease of Implementation:** This defines how easy it is to set up and maintain the AC model for a specific use case. This is where ABAC takes a backstep. It is complex to build since it requires identifying many attributes, writing policies and setting up the infrastructure for dynamic evaluation.

v) **Scalability and dynamic adaptation:** It defines how well the model will perform when the system grows in size and complexity. ABAC, not depending on static permissions or hardcoded rules, can scale better as it only requires more attributes. Similarly, it takes extra effort to make the model completely dynamic. For example, ABAC can easily handle user or application growth with cloud computing.

vi) **Use case suitability:** The specific scenarios and applications where the access control model is most effective are highlighted in this criterion. Organizations rapidly shift towards cloud computing and IoT services to improve scalability, flexibility, and

operational efficiency. This shift necessitates the implementation of fine-grained, context-aware access control policies. In response to these evolving security needs, attribute-based access control (ABAC) has become an increasingly preferred approach and is beneficial for dynamic and large-scale ecosystems where traditional role-based models struggle to meet evolving access requirements. Its suitability extends to healthcare, finance, and government sectors, where access decisions must be highly adaptive and context-driven.

1.4. Relationship of ABAC to Other Models

1.4.1 ABAC vs. RBAC - ABAC extends RBAC by allowing role-based policies while also incorporating additional attributes like user attributes (e.g., department, clearance level), environmental attributes (e.g., time of access, device type), and object attributes (e.g., file sensitivity).

1.4.2 ABAC vs. MAC - ABAC provides fine-grained control similar to MAC but is more flexible by allowing dynamic policies based on various attributes rather than predefined security labels.

1.4.3 ABAC vs. DAC - Unlike DAC, which relies on the discretion of resource owners, ABAC enforces policies centrally and can prevent privilege escalation.

1.4.4 Crawl-Walk-Run Approach for ABAC - As mentioned in Table 2, to transition to ABAC from simpler models:

Crawl (Basic)	**Walk (Intermediate)**	**Run (Advanced)**
Start with **RBAC** by assigning users to roles based on their job function.	Introduce **context-based conditions** in addition to roles (e.g., access only during working hours, device-based access).	Fully implement **ABAC** with dynamic, real-time policies considering multiple attributes beyond roles (e.g., user attributes, environmental conditions, object sensitivity).

Table 2 Crawl-Walk-Run approach for ABAC from other models

1.5. ABAC standard frameworks and the novel hybrid framework

1.5.1 eXtensible Access Control Markup Language (XACML)

ABAC is commonly implemented using XACML, an eXtensible Markup Language (XML)-based standard that defines access control policies. It is a standard defined by the "Organization for the Advancement of Structured Information Standards" (OASIS) [26]. The reference architecture shown in Figure 6 constitutes the main components of XACML 3.0 (third-generation model), including the access evaluation and enforcement process [27]. The component details are as follows:

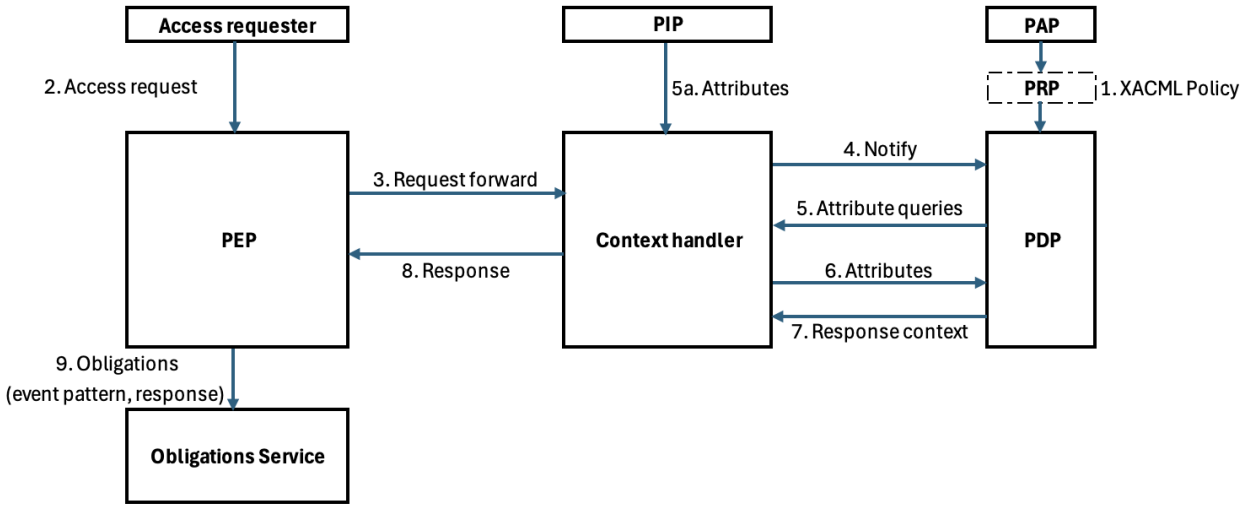


Figure 6 XACML reference architecture [27]

i. Policy Administration Point (PAP)

PAP is a repository where resource administrators or owners store policies. These policies are used by the Policy Decision Point (PDP) to evaluate access requests, ensuring that user requests are checked and matched against the stored policies.

ii. Policy Decision Point (PDP)

PDP is responsible for evaluating access requests by comparing them to the stored policies and determining whether to permit or deny the request.

iii. Policy Retrieval Point (PRP)

PRP is an optional component that stores and provides policies to the PDP, typically used in complex or distributed environments to decouple policy storage from decision-making; if absent, the PDP retrieves policies directly from a repository or local database attached to the PAP.

iv. Policy Enforcement Point (PEP)

PEP intercepts a user's resource access request and translates it into XACML format with the Context Handler. It forwards the request to the PDP for evaluation and enforces the decision (permit or deny) provided by the PDP.

v. Context Handler (CH)

CH acts as a bridge between the PEP and PDP. It translates PEP's native access requests into XACML-compliant authorization requests and converts PDP's XACML responses back into a format which PEP can enforce.

vi. Policy Information Point (PIP)

PIP supplies the necessary attribute values (subject, resource, environment, and action attributes) required during access evaluation. It responds to queries from the CH when the PDP needs external information that is not provided directly in the access request. The PIP enables dynamic and context-aware decision-making by integrating real-time attribute sources.

vii. Obligations Service

The Obligations Service handles operations specified in policies, rules, or policy sets. These operations, known as obligations, are triggered based on specific event patterns during access enforcement [28]. An obligation is defined as a pair (event pattern, response), where the event pattern specifies the conditions under which an obligation is activated, and the response determines the administrative actions that must be immediately executed. PEP is responsible for enforcing both the access decision and any associated obligations, ensuring that security policies, such as conflict of interest

restrictions, workflow progressions, or data leakage prevention measures, are dynamically and contextually applied following access request evaluations. For example, if a user successfully accesses a confidential document (event pattern), an obligation may trigger the immediate logging of this access into a secure audit system (response), ensuring traceability and compliance with organizational security policies.

1.5.2 Next Generation Access Control (NGAC)

Although ABAC significantly improves granularity and adaptability, implementing it through standalone XACML can present challenges. Most are related to complexity in policy creation and management, decision-making performance, and policy evaluation considering context and the dynamic nature in large-scale environments [10]. This has driven the need for alternative approaches, such as the Next Generation Access Control (NGAC), introduced by the National Institute of Standards and Technology (NIST) [28]. NGAC utilizes attribute relationships, which offer a framework to simplify policy management and improve access control. This approach allows faster context-based adjustments without making policy modifications. As shown in Figure 7, NGAC abstracts all low-level data types of different resources and treats them as objects. It simplifies the AC decisions through a logical and scalable structure [29] and is designed to work across multiple operating environments (OE).

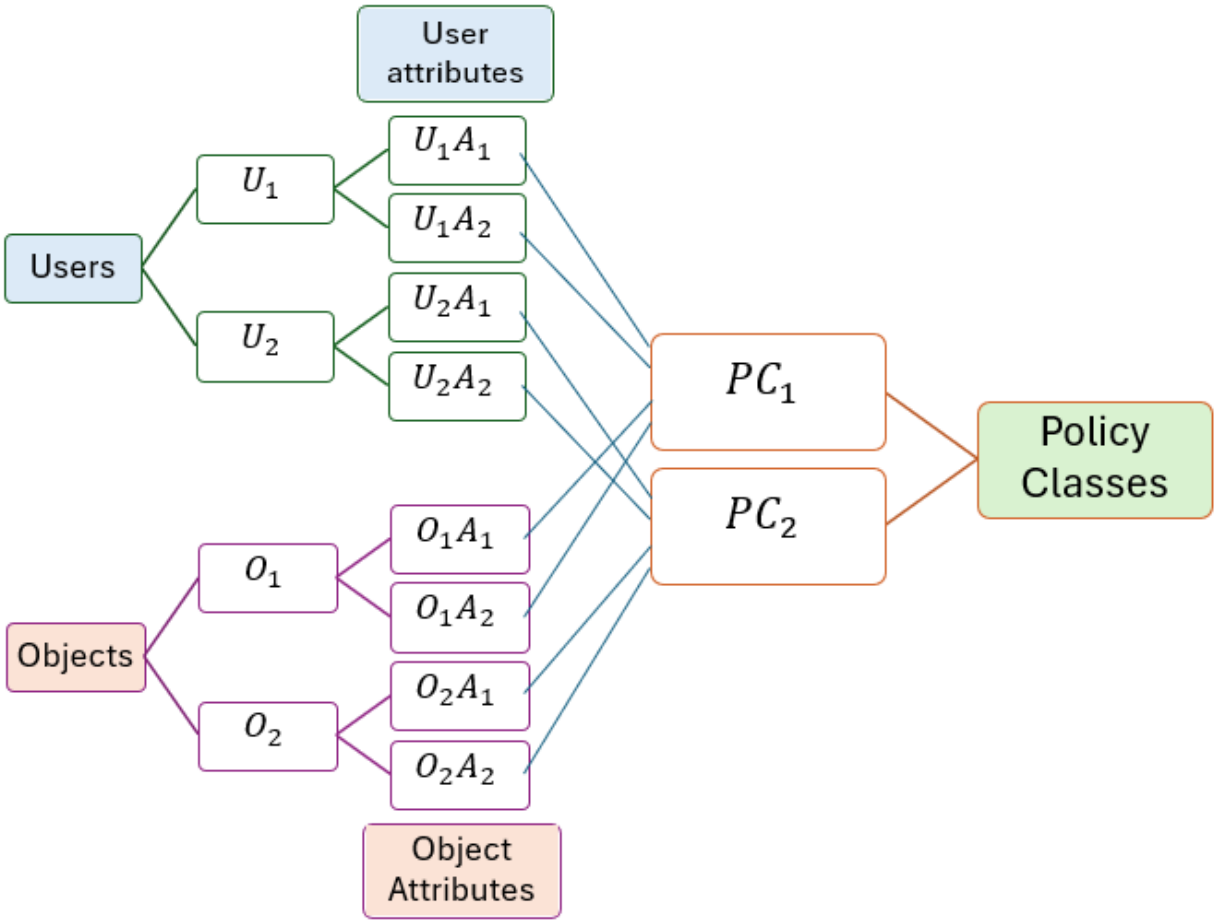


Figure 7 NGAC Graph Architecture [29]

The core components of the NGAC architecture are described below:

i. Basic elements - Users, Processes (NPE), Operations and Objects.

ii. Container

   a. User container - Group users by roles, affiliations, or security clearances.

   b. Object container - Group resources by attributes, such as security classifications or applications.

   c. Policy Class Containers - Organize related policy elements or services into distinct sets.

iii. Relationships

   a. Associations - Define permissions between user and object containers (e.g., read, write).

   b. Prohibitions - Specify restricted actions for certain users on specific objects.

iv. Dynamic policies - It adapts based on user roles, object attributes, and contextual factors like device type, time or location.

v. Obligations – Similar to the XACML model, these are the conditions which should be fulfilled before evaluation and the actions that should happen after the decision. These conditions enhance control by linking access decisions to specific actions.

Additionally, NGAC enforcement also supports complex functionalities such as usage control, history-based constraints and separation of duty (SoD). Usage control manages not only who can access a resource, but also how it is used once access is granted. It can enforce limits on how long a file can be viewed, how many times it can be downloaded, or which actions can be performed. For example, a doctor may access a patient record for only 30 minutes before re-authentication is required. History-based constraints apply rules based on a user's past actions. They ensure that previous activities influence future access rights. For instance, if a user has already approved a document, they might be restricted from approving another related document to prevent conflicts of interest. Separation of Duty (SoD) splits critical tasks between different users to prevent fraud or mistakes. For example, the employee who submits an expense claim should not be the same person who approves it. This ensures independent verification and protects sensitive processes.

However, even though NGAC is effective for straightforward policy enforcement, it presents some challenges when used alone. For instance, it does not provide extensive policy details and flexibility for XACML. Furthermore, its lower industry adoption and lack of standardization can result in fewer available tools, reduced community support, and limited implementation resources. Therefore, although NGAC streamlines the structure for enforcing access control, it does not match the level of granularity that XACML offers, making it less suitable for systems that demand intricate policy rules.

Despite XACML and NGAC's advancements, neither model fully addresses all the challenges of dynamic, scalable, and context-aware access control in complex environments such as cloud computing. Combining the flexibility of XACML with the structural advantages of NGAC can create a hybrid model that leverages the strengths of both systems. This hybrid approach aims to enhance policy expressiveness, simplify administrative management, and improve decision-making. Developing a hybrid ABAC framework represents a significant step forward in modern access control strategies, ensuring robust, context-aware security that meets the evolving demands of contemporary business and technology landscapes.

1.6. Motivation

Organizations increasingly rely on collaboration and data exchange to achieve their goals in today's interconnected world. This necessitates effective mechanisms for sharing data and seamlessly granting access in real-time to authorized users from geographically dispersed teams. The shift in data storage from traditional on-premises servers to cloud servers owned by a private vendor gave birth to cloud computing access management (CCAM). This encompasses a set of policies, procedures, and technologies designed to control access to cloud resources, including

data storage, applications, and services. Effective access management is crucial for several reasons. It includes authenticating users to verify their claimed identity [based on something they know, something they have, something they do, or something they are], authorizing user details to provide sensitive data access (or minimum level of access required) and promoting compliance with application privacy and security regulations.

Controlling the user authorization comes primarily with a traditional access control mechanism. RBAC is deeply integrated into current industry-level access management and is used extensively to manage access to resources, applications, and administrative roles. The system administrator creates multiple roles based on the required access level for a given position/job (or role) and then assigns those roles to registered users. The defined roles are often static and cover broader access control, making adapting to changing user needs or security threats difficult. Hence, more granular controls are needed to handle an increasingly complex digital landscape and improve user experience. As a result, ABAC was introduced to effectively implement access management and authorize user access based on the attributes or tags associated with a particular user profile, user actions, the hosted resources and environment, such as time of the day or location [30]. It is a superset of RBAC since the roles can be defined as attributes in the user profiles or application attributes. Although it provides more flexibility, granularity, and dynamic security control, implementing it could result in more complex policies or time-consuming processes. New models are being developed continuously to use ABAC with less complexity. However, it has been observed that access provision efficiency can be improved by combining multiple methods [31].

Access management should not be limited to static information like user details or fixed policies; instead, it should be adaptable, allowing decision patterns to change based on dynamic data. Although several methods and applications have been defined to provide such capabilities, they prioritize analyzing the most recent access patterns rather than historical data and do not provide any customization features per the application design or changes in security requirements [4], [32]. Due to this, the results could be less precise and include false-positive scenarios, such as flagging normal users, or false-negative scenarios, such as missing high-risk users. Additionally, ABAC models do not provide a mechanism to prioritize specific attributes or sets of attributes over others during policy evaluation. This limitation makes it difficult to determine whether a user holds essential attributes critical for access decisions, such as security clearance, role sensitivity, or known risk indicators. As a result, the model treats all attributes equally, which may lead to granting access even when key high-impact attributes are absent or potentially denying access merely because a less critical attribute is missing. A more granular model is needed that can be used to implement user-level policy and that can be customized or tuned over time to include historical data and application requirements. To implement this concept, a custom security model can be developed with tailored policies based on specific access requirements. This model can be integrated with an existing identity and access management tool to focus on the dynamic access evaluation while outsourcing the authentication processes and user profile management.

1.7. Proposed work

This research focuses on application-level access control in a cloud environment, specifically evaluating access decisions for simulated user profiles interacting with multiple applications hosted on Microsoft Azure. At this layer, access control is implemented using a hybrid framework that combines ABAC, which implicitly encompasses RBAC, with the policy logic of XACML and the graph-based enforcement of NGAC. Microsoft Entra ID is used as the identity provider to manage authentication, user attributes, and conditional access policies, while the hybrid model evaluates the authorization using additional dynamic and context-aware decision-making capabilities beyond standard techniques.

The overall research is carried out in four phases. In the first phase, a test infrastructure is created in Microsoft Entra ID, including multiple users, groups, applications and other Entra components to mimic the industry-level access management structure. The required privileges to access these resources are provided to the test environment, which is used to integrate the hybrid model. The complete setup is implemented programmatically to provide a more efficient way to manage the infrastructure in future.

In the second phase, ABAC controls are applied using a new hybrid model that includes XACML and NGAC models. XACML has been implemented to manage the access policies and static attributes. In contrast, NGAC operates at a broader level to manage dynamic and contextual conditions, offering improved performance in complex scenarios through efficient in-memory graph-based computations. The design distributes the access evaluation process by combining the complementary strengths of XACML and NGAC, leveraging XACML's fine-grained, policy-driven precision and NGAC's dynamic, context-aware enforcement, while

simultaneously reducing their individual limitations, such as XACML's rigidity and NGAC's over-permissiveness[34]. A development environment is created to test the system using provisioned resources and distinct access requests. Access decisions are evaluated not solely on attribute values but through an *integrated mechanism* involving policy checks, graph relationships, and runtime context analysis techniques developed as part of this research. This hybrid method enables the resolution of conflicts between models and enhances overall decision accuracy, precision, recall, as well as the F1-score (a combined metric that balances precision and recall). Additionally, the architecture includes a novel *risk assessment engine* that evaluates user behaviour and logs data to compute dynamic risk levels. These risk scores are incorporated into policy conditions and attribute checks to influence final access decisions. Considering the factors involved in the access evaluation and a new approach, the framework is named **Dynamic Trust Weighted-Attribute Based Access Control Hybrid Framework (DTW-ABAC).**

The third phase generates comprehensive simulated data, including sign-in and risk assessment logs. The simulated environment replicates realistic access scenarios by incorporating real-world examples, such as applications with domain-specific attributes (e.g., healthcare, education, technology), users with varying roles and privilege levels, and permissions aligned to different actions or operations. The details about the quality of workloads are discussed further in the methodology section. The risk logs are accomplished by programmatically simulating normal and suspicious user behaviour during sign-in processes and resource access using registered user profiles. As per the NIST publication [33], NGAC's key advantage is: "The system's ability to tightly restrict access without losing the all-important capability of scalability." The high volume of access requests will be initiated from various virtual locations to enhance the realism of the simulation. This is achieved using Virtual Private Networks (VPNs) and the Tor network

(via the Tor browser), thereby introducing dynamic and varied geolocation data into the logs. The simulation also includes dynamic role assignments, access from different devices and browsers, and sign-ins at times outside of regular working hours. These variations create a diverse dataset that captures a range of risk levels, from normal to high-risk behaviours. This process generates initial log data that is used to fine-tune and refine the developed model. The model iteratively improves over time based on comparisons between its results and the expected outcomes.

In the fourth phase, the model parameters are fine-tuned to enhance decision accuracy across test scenarios, strengthen the reliability of trust factor evaluations over time, and validate the system's effectiveness in detecting common attack patterns. This includes iterative testing using varying model constants, attribute sets, and policy parameters to assess their impact and identify optimal configurations. In addition to performance tuning, the model undergoes security evaluation through simulated attack scenarios, including unauthorized access attempts, privilege escalation, and injection of forged attributes. These tests help assess the robustness of the hybrid model against common access control threats and verify its ability to respond to malicious behaviours in real-time. The scope for future enhancements is clearly defined. Results are analyzed and compared with those of other access control models using statistical confusion matrix metrics (TPs, TNs, FPs, FNs) and Performance Metrics for Classification Models (Accuracy, Precision, Recall, and F1-Score). A conclusion regarding the hybrid model's effectiveness is drawn, and the thesis is documented with all relevant findings, analysis, and references.

1.8. Contributions

This research aims to verify the challenges in the current access solutions regarding granular access control and find solutions to overcome them. After considering the currently available security methods, adding a hybrid access control model with Entra ID and other solutions aims to achieve five goals, each answering a specific research question.

1. Hybrid Access Control Framework Combining XACML and NGAC

   Research Question:

   How can the hybrid integration of static and dynamic models improve access correctness in terms of Accuracy, Precision, Recall, and F1-score?

   Contribution:

   The research introduces a layered architecture that combines XACML's rule-based evaluation and NGAC's dynamic context logic into a coordinated decision pipeline. This hybrid design allows the decision mechanism to consider both predefined access rules and runtime context, such as environment, user state, and application behaviour, ensuring higher decision quality and adaptability in authorization decisions.

2. Trust-Based Risk Evaluation with Adaptive Risk Reclassification

   Research Question:

   How can user access history and trust metrics be used to update risk posture dynamically?

   Contribution:

   A dynamic trust evaluation model is developed to compute weighted access decisions based on attribute relevance and contextual integrity. Users are assigned evolving risk

classifications labels (low, medium, high) based on how consistently their access patterns align with organizational expectations. The risk labels are quantified with associated scores (e.g. Low = 1, Medium = 1.5 and High = 2) that inversely affect the historical confidence factor. These risk scores influence future decisions and support proactive risk management by continuously adapting access boundaries based on trust loss or improvement.

3. Scenario Diversification through FSM-Guided Policy Testing Framework

   Research Question:

   How can realistic and policy-relevant access scenarios be systematically generated for robust evaluation?

   Contribution:

   This work proposes a structured methodology for generating high-impact access scenarios using finite-state machine (FSM) logic to represent access paths and attribute transitions. The approach aims to produce diverse and realistic access attempts, ranging from compliant to borderline or adversarial, helping researchers or administrators validate policy correctness and system resilience under various operational conditions.

4. Attribute Governance and Weighted Policy Enforcement Strategy

   Research Question:

   How can attribute criticality and governance be embedded within access control frameworks?

   Contribution:

   A centralized attribute management strategy is proposed to differentiate between core and

supporting attributes, assign weights based on their security impact, and define mandatory conditions for access enforcement. This approach is intended to enhance compliance, enable fine-grained policy authoring, and ensure that sensitive resources are only accessible when high-confidence attribute matches are observed.

5. Explainable Decision Model with Traceable Evaluation Paths and Integration Support

   Research Question:

   How can access control models ensure transparency, auditability, and operational applicability?

   Contribution:

   The research outlines a multi-layered decision-making model that supports traceability of the whole evaluation process, from attribute collection to policy resolution and trust calculation. By structuring policy logic and runtime signals into clearly defined decision flows, the model supports policy explainability, eases troubleshooting, and can support integration across multiple systems while maintaining consistent governance and risk visibility.

Chapter 2

Related Work

This section presents a detailed review of the current advancements, limitations, and research gaps in access control mechanisms, focusing on integrating XACML and NGAC. The review draws upon diverse sources, including peer-reviewed journal articles, technical conference papers, book chapters, and official National Institute of Standards and Technology (NIST) publications. As mentioned in [34], their mission is to "promote U.S. innovation and industrial competitiveness by advancing measurement science, standards, and technology in ways that enhance economic security and improve our quality of life."[35] Special attention is given to modern and hybrid access control models, particularly those tailored for dynamic environments such as cloud platforms, IoT networks, and multi-domain enterprise systems. The selected literature provides theoretical and practical insights into policy expression, enforcement techniques, scalability, flexibility, and interoperability challenges. This review also examines the possibility of combining XACML's policy logic with NGAC's graph-based structure can overcome individual shortcomings, leading to more adaptive, fine-grained, and context-aware access control frameworks.

2.1. Current Access Control Hybrid Models

Access control is a key part of digital security. It helps decide who can access what information and under what conditions. However, as organizations handle more complex and larger amounts of data, traditional models like DAC, MAC, RBAC, and ABAC start showing limits, especially in systems that are constantly changing or spreading across different domains. The

comprehensive survey by M.U. Aftab et al. [18] explore both traditional and hybrid AC models, offering critical insights into their strengths, weaknesses, and suitability across domains such as IoT, cloud computing, and e-health, as summarized in Table 3. It also explains how traditional ABAC suffers from policy specification complexity and high computational overhead. While easier to manage in static environments, RBAC struggles with flexibility and scalability. Hybrid models are designed to bridge these gaps by taking the strengths from two or more AC mechanisms and performing efficiently compared to them.

Model Name	Functionality	Strength	Weakness
Temporal Role-Based Access Control (TRBAC)	Extends RBAC with temporal constraints (e.g., time-bound access).	Improves the timeliness of access decisions in dynamic environments.	Policy complexity increases; it lacks fine-grained attribute control.
Attributed Role-Based Access Control (ARBAC)	Combines ABAC's attribute evaluation with RBAC role structure.	Provides flexibility of ABAC with the structure of RBAC.	Limited support for role hierarchies and Segregation of Duties (SOD) constraints.
Fuzzy-Based Access Control (FBAC)	Applies fuzzy logic to attribute conditions for flexible access decisions.	Handles uncertain or imprecise access requirements effectively.	Limited policy enforcement strength; difficult to audit.
Emergency Role-Based Access Control (E-RBAC)	Adds emergency override (Break-the-Glass) features into RBAC.	Supports emergency access scenarios with audit trails.	May conflict with standard access policies; management overhead.
RBAC with Smart Contracts (RBAC-SC)	Implements RBAC using blockchain smart contracts for auditability.	Enhances transparency and tamper-proof logging via blockchain.	Scalability issues in large organizations, infrastructure dependency.
Trust-Aware Role Assignment System (TARAS)	Dynamically assigns roles based on user trust scores and behavioural patterns in IoT and	Enhances security in dynamic IoT environments by detecting malicious users,	Trust score calibration is subjective and may be difficult to standardize across

	wireless networks. Operates at a broader role-assignment level rather than specific task access.	improving integrity, availability, and robustness under high attack conditions.	heterogeneous IoT systems. Limited applicability outside wireless networks and cloud-based IoT systems.
Trust-Based Access Control (TBAC)	Makes fine-grained access decisions based on calculated trust values at the task or data access level, particularly in online social networks (OSN).	Highly effective in distributed, dynamic environments like OSNs with multi-role flexibility, enabling collaborative user-based access control.	Susceptible to trust value manipulation if recalibration is not frequent. Administrator oversight is weak or missing, creating gaps in moderating unethical content.

Table 3 Comparison of application-specific hybrid access control models

A similar hybridization concept for IoT applications extends the traditional XACML model to better-fit environments that require real-time, context-aware decisions [35]. The authors propose a framework called FB-ACAAC (Fog-Based Adaptive Context-Aware Access Control), which works in distributed fog environments (an architecture that extends cloud computing to the edge, bringing computing resources closer to the data source) with critical latency and responsiveness. The system uses contextual inputs like time, user behaviour, device type, and location to decide access permissions. The access control logic is deployed at the fog layer (closer to the devices), reducing delays and minimizing the load on central cloud servers. Their implementation uses standard XACML components and custom logic for dynamic context evaluation, showing practical feasibility using Raspberry Pi-based fog nodes. One important contribution of this paper is the adaptability of policy enforcement. Unlike traditional XACML, the FB-ACAAC model adapts policies based on runtime context without rewriting core rules. While these models

demonstrate innovative solutions to evolving access control challenges, they often sacrifice manageability, introduce complexity, or rely on specific infrastructures like blockchain or IoT.

With respect to enhancing the traditional model capabilities in the cloud environment, a model named Attribute-Based Access Control and Communication Control (ABAC-CC) was developed to improve security in Cloud-Enabled Internet of Things (CE-IoT) environments [17]. Traditional ABAC models mainly control who can access which resources by using attributes like the user's role, the resource type, and environmental conditions. ABAC-CC takes this further by also controlling how devices communicate with each other. They add a new layer called Attribute-Based Communication Control (ABCC), which applies rules to the messages exchanged between users, devices, and services. The model uses several attributes from the device, user, application, and message to make access and communication decisions, improving context awareness, privacy, and flexibility, especially in smart homes, smart healthcare, and transportation networks. Policies in ABAC-CC are designed in a modular way. This implies that access control and communication control can be handled separately, which helps make the system more adaptable to different types of IoT setups. However, the paper does not include real-world implementation or performance results. This leaves concerns about how well the model would scale, handle conflicts between rules, or perform in larger systems. Similar layer extension can also be seen in two hybrid models, i.e. HyBACRC (Hybrid Attribute-Based Access Control with Role-Centric approach) and HyBACAC (Hybrid Attribute-Based Access Control with Attribute-Centric approach), to improve access management in smart home IoT environments [36]. These models aim to combine RBAC's ease of use with ABAC's flexibility. In HyBACRC, roles are assigned based on relatively static user attributes. Then, dynamic attributes like time, temperature, or device state are used to fine-tune the decision. In contrast,

HyBACAC puts attributes first and uses roles only as supporting context, making it more suitable for changing environments. The paper presents clear, formal definitions, examples, and an AWS-based prototype. The use cases are relevant to real-world IoT setups, such as restricting access to a room or device based on context. However, a key limitation is redundancy; since ABAC already covers most RBAC functionality, mixing both can complicate policy design and lead to policy overlap. Secondly, while the HyBACRC model simplifies role management, it risks a "role explosion" when combined with many dynamic attributes, creating administrative challenges. Conversely, the HyBACAC model, though more attribute-focused, complicates ongoing policy maintenance and auditing due to its highly dynamic structure. Also, these models are tightly focused on smart home systems, so their general applicability to large enterprise or cloud environments is not fully explored. A broad and systematic survey of different hybrid AC models in modern computing environments can be found in [11]. The survey classifies and compares over 25 access control models based on various parameters like flexibility, scalability, granularity, use of attributes or roles, dynamic behaviour, trust handling, context awareness, encryption, workflow support, and more. The models range from traditional ones like DAC, MAC, and RBAC to modern hybrid and context-aware approaches like NGAC, TrustBAC (Trust-based AC), CBAC (Context-based AC), and Blockchain-based access control. Each model is analyzed for its strengths and limitations. For example, RBAC is noted for its simplicity and scalability but lacks fine-grained control. ABAC offers attribute-based flexibility and is widely adopted in cloud services. NGAC is a scalable model suitable for distributed systems, offering real-time policy enforcement in memory. TrustBAC and RiskBAC bring dynamic trust and risk evaluations into decisions, which is important for behaviour-based control. The author also highlights newer models like PBAC (Policy-Based Access Control) for provenance tracking,

SEAC (Situation- and Environment-Aware Access Control) for distributed database systems, and CBAC for ubiquitous and IoT environments. Importantly, the paper underscores that no single model fits all use cases, and hybrid solutions are increasingly necessary. A side-by-side comparison of major access control models with the DTW-ABAC Hybrid model, highlighting the presence (X), partial inclusion (P), absence of key features (-), or not mentioned (?), is presented in Table 4.

Metrics	**ABAC**	**RBAC**	**NGAC**	**ReBAC**	**Trust-BAC**	**HyBACRC**	**HyBACAC**	**DTW-ABAC Hybrid**
Identity storage	X	X	X	X	X	X	X	X
Fine-grained policies	X	P	X	X	P	X	X	X
Dynamic decision-making	P	-	X	P	X	X	X	X
Workflow control	P	X	X	P	?	P	X	X
Delegation of trust	-	-	X	X	?	P	P	X
History-keeping	-	-	X	-	?	P	P	X
Scalability	X	X	X	P	P	P	P	X
Encryption/ Tokenizatio	-	-	-	-	-	P	P	P
Attribute-based access	X	P	X	P	P	X	X	X
Role-based assignment	-	X	P	-	-	X	P	X

Risk factor evaluation	P	-	X	-	X	P	P	X
Distributed compatibility	P	P	X	X	P	P	P	X
Artificial Intelligence / Predictive Analysis	-	-	-	-	-	-	-	-
Time constraint	P	P	X	P	-	P	P	X
Location constraint	P	P	X	P	?	P	P	X

Table 4 Comparative feature analysis of access control models

2.2. ABAC Standalone Models

2.2.1. XACML (eXtensible Access Control Markup Language)

XACML is a widely accepted standard developed by OASIS (Organization for the Advancement of Structured Information Standards) for implementing ABAC. It provides a flexible, XML-based framework to define access control policies, evaluate requests, and make decisions [26]. As explained in 1.5.1, the XACML model separates the roles of policy definition, decision-making, and enforcement across distinct architectural components - the Policy Enforcement Point (PEP), Policy Decision Point (PDP), Policy Administration Point (PAP), Policy Information Point (PIP), Policy Retrieval Point (PRP), Context Handler (CH) and Obligations (Figure 6) [27], [29]. XACML policies are organized into rules, policies, and policy sets hierarchy. Each rule contains a condition and an effect (permit/deny). For example, a rule could state - "Permit access if the user's department attribute is 'HR' and the action is 'read'." Policies aggregate multiple rules and specify a combining algorithm (such as permit overrides or deny

overrides) to resolve conflicts among them. For instance, a policy could combine several rules to allow employees to read their own records but deny access to others' records, using a 'deny overrides' strategy to prioritize denials. A policy set groups multiple policies to manage larger collections of related access rules across departments or systems. An example of a policy set would be grouping HR, Finance, and IT policies under a single organizational access control framework. This makes XACML expressive for describing fine-grained access decisions based on attributes of users, resources, actions, and the environment. XACML is used across domains such as cloud systems, federated identity environments, government security systems, and financial data systems. For instance, many commercial identity providers like WSO2 and Oracle Identity Manager integrate XACML to manage authorization across complex enterprise setups [30]. Patra et al. propose an innovative approach to enhance performance in ABAC implementations based on the standard XACML architecture, specifically within the context of electronic healthcare records (EHR) [27]. As healthcare data is highly sensitive, secure and efficient access control mechanisms are essential. Their research identifies a common performance issue in traditional XACML implementations, where each denied request is repeatedly re-evaluated, causing inefficiencies. To address this, they introduce a novel Request Denial Cache (RDC) within the XACML reference architecture. This RDC retains the attributes of denied requests, preventing redundant policy evaluations for identical subsequent requests, thereby significantly reducing processing overhead and improving overall system performance. This approach is especially valuable in healthcare settings, where quick and secure access decisions are vital due to high concurrent requests. Their model also provides resilience against insider flooding attacks, a common vulnerability wherein a legitimate but malicious user overloads the server by repeatedly requesting denied access. Additionally, the solution

incorporates detailed audit logging of all access requests and their outcomes, which helps in retrospective security analyses and compliance verification. However, their research has some inherent limitations. Firstly, the Request Denial Cache requires manual maintenance. For instance, an administrator must manually remove the outdated entries from the cache whenever user privileges change. This manual intervention can be problematic in large-scale or dynamic environments, introducing the possibility of human error and delays in updating access privileges. Secondly, their proposed solution still adheres strictly to XACML, inherently limiting dynamic context-awareness. XACML policies, being predefined, lack the ability to adapt dynamically in real-time based on evolving contexts, such as rapid changes in risk or recent user behaviour patterns. This static nature could result in persistent false negatives, unintentionally denying legitimate requests due to outdated or overly rigid policy evaluations. Another potential limitation of their model is its scalability. Although RDC reduces redundant processing, continuously growing cache entries could lead to increased memory usage over time, potentially requiring complex cache eviction strategies and further administrative overhead.

The following points summarize the primary limitations associated with the XACML access control model:

1) Lack of Real-Time Context Awareness

   XACML evaluates requests against static policy sets and lacks built-in support for real-time or historical context (e.g., user's access history or current risk level). This limits its effectiveness in dynamic systems [26], [28].

2) Over-Restrictive by Design

   The strict policy evaluation and denial by default can lead to false negatives, where legitimate users are denied access due to missing attributes or rigid policy rules. This makes

it difficult to adapt to flexible or exception-based scenarios (e.g., emergency access in healthcare) [37].

3) Scalability Issues

   The use of XML and deep hierarchical policy nesting introduces significant performance overhead. Policy evaluation latency increases as policies grow complex or reused across domains, particularly in large-scale cloud or multi-tenant environments [26].

4) Policy Authoring Complexity

   Writing XACML policies requires technical expertise in XML and an understanding of policy-combining algorithms. Administrators must carefully validate and test policies for minor changes to avoid unintentional access violations [30]. It cannot fulfill the emerging demand for flexible security models.

5) Limited Policy Conflict Resolution

   While XACML supports combining algorithms, they are insufficient for resolving complex policy overlaps or inconsistencies in distributed systems. Manual conflict resolution is still required, adding to the maintenance effort.

2.2.2. NGAC (Next Generation Access Control)

Unlike XACML's rule-based design, NGAC adopts a graph-based model where users, objects, and attributes are represented as nodes and relationships (like assignments or associations) are represented as edges, as shown in Figure 7. For example, a user node could represent "Nurse_Alice", an object node could represent "PatientRecord_123", and an attribute node could represent "Role_Nurse".

Assignments link users and objects to attributes, while associations define what actions are allowed. For instance, an assignment edge could connect "Nurse_Alice" to "Role_Nurse", and an association edge could allow "Role_Nurse" to "read" "PatientRecord_123". As explained in 1.5.2, NGAC stores policies and access rights in memory as graph objects and evaluates decisions by traversing this graph. This structure supports dynamic and history-aware access decisions more efficiently than traditional rule evaluation. It can also enforce obligations and prohibitions, which makes it suitable for dynamic and context-rich environments such as collaborative systems, IoT, and streaming data platforms [38]. Like XACML, the NGAC architecture uses the policy administration point, access requestor, decision function (conceptually the same as PDP), and policy repository. Enforcement is done in memory, enabling high-speed evaluation, and it can support complex policies like separation of duty, history-based constraints, or usage control.

Recent research has shown that NGAC can be extended for multi-policy evaluation, risk-aware access, and trust integration [21], [38]. However, NGAC still lacks formal standardization and is primarily used in research or controlled prototypes. [40]A framework to build NGAC policies automatically from natural language was introduced in [39]. Using Natural Language Processing (NLP) tools like spaCy, extracts access control elements such as users, actions, and objects from written policy rules. These are then mapped into NGAC graph structures within a Neo4j database. The authors also propose methods to validate the generated policy graphs using NGAC correctness checks such as consistency, completeness, and minimality. The system allows both access permissions and obligations or prohibitions to be defined. This makes the NGAC structure more expressive and usable in real-world security policy enforcement. This paper reinforces the

importance of using NGAC's graph-based policy structure. However, NGAC also comes with some limitations, such as:

1. Lack of Standardization and Adoption

   Despite being recognized as an official standard, mainstream identity and access management solutions do not support NGAC, unlike XACML. This leads to interoperability gaps and complicates integration with existing enterprise systems [38].

2. Over-Permissiveness

   NGAC's flexibility can backfire when policy graphs are poorly designed or not tightly constrained. False positives may occur when users gain access due to transitive relationships or implicit associations in the graph structure [28], [37].

3. Complex Graph Management

   Policy administration in NGAC requires understanding and managing complex graph structures. Graph bloat can make performance tuning and debugging difficult as the number of users, objects, and policies increases [21].

4. Harder Migration from Legacy Systems

   Organizations with legacy XACML-based infrastructure may struggle to upgrade to NGAC due to a lack of tooling or migration support. This creates resistance to adoption even when NGAC may be more suitable in theory. Rather than requiring a complete overhaul, this situation highlights the need for a hybrid framework that can integrate XACML with NGAC, providing a gradual transition and easier adoption for security administrators.

5. Limited Tooling and Visibility

   Few development tools or GUIs are available for NGAC. Visualizing policies, detecting

misconfigurations, or performing audits remains challenging. This limits its usability in real-world operational settings.

2.3. Combining NGAC and XACML

XACML is rigid, less scalable, and lacks context awareness, while NGAC is flexible but immature and complex without proper design. Combining XACML's structured policies and NGAC's dynamic evaluations into a hybrid model addresses these issues by reducing false decisions, improving interoperability, and managing policy complexity. Inspired by existing NGAC approaches, the proposed hybrid framework utilizes relational tables, using Structured Query Language (SQL), to connect users and applications to their attributes and permissions, incorporating historical access data for real-time trust scoring. This design enables deeper context evaluation and adaptability suited to dynamic environments such as cloud and enterprise systems. While standalone models are often evaluated qualitatively, quantitative metrics such as True Positives (TP), False Positives (FP), True Negatives (TN), False Negatives (FN), and related performance measures (e.g., precision, recall and F1-score) offer a clearer understanding of decision accuracy. This becomes especially relevant when comparing static policy models like XACML with dynamic ones like NGAC. A hybrid model can optimize the trade-offs, reducing FP and FN rates across diverse scenarios.

2.4. Trust factor-based access control and common vulnerabilities

A growing body of research explores how fuzzy logic can compute trust in access control in cloud environments. Trust is a quantitative measure of a cloud service provider's and user's reliability, security, and behaviour over time. This trust score helps access control decisions or

service selection in dynamic environments. Kesarwani and Khilar propose a dual-layer fuzzy logic-based access control model to assess both users and Cloud Service Providers (CSPs) using behavioural and Quality of Service (QoS) parameters [40]. Their model for user trustworthiness factors in HTTP status code metrics, such as bad requests (HTTP 400), unauthorized access attempts (HTTP 401), forbidden access (HTTP 403), and not found errors (HTTP 404). These codes help indicate abnormal or suspicious user behaviour during access attempts. The frequency and severity of such requests result in a negative trust score. This score is then interpreted using weighted fuzzy inference rules, logical constructs that map input factors (e.g., severity levels) to qualitative trust levels (e.g., low, medium, high) through fuzzy logic reasoning. For CSP trust evaluation, they use performance and elasticity, quantified using attributes like workload, response time, availability, scalability, security, and usability. The model uses Mamdani fuzzy logic with Gaussian membership functions for fuzzification and triangular functions for defuzzification. The AR-ABAC (Attribute Rules-ABAC) model by Riad et al. introduces a flexible attribute-weighting mechanism where each attribute is assigned a numeric weight, and attribute sets are classified into power groups (G1 to G5) based on average weight [41]. However, the weights are not considered to take the weighted sum or confidence calculation, but rather the attribute group influence the user role and object sensitivity level assignment. G1 represents the low weight, so used to assign basic rules, whereas G5 has a high weight, used to assign sensitive roles. Also, a subjective call is used to assign static weight to attributes and with no provision to update them dynamically. An enhancement can be made to find the optimal weights based on rigorous experiments and observations. Then, a logic can be written to update them dynamically based on previous patterns after specific days or request count. While the AR-ABAC model provides a scalable and fine-grained access mechanism, the model also lacks

explicit mechanisms for marking attributes as "essential" or "mandatory". It relies on zero-weight exclusion. This static assignment of weights might limit adaptability to dynamic contexts or user behaviour trends. Another prominent approach is the Parameter-Based Trust Calculation (PBTC) model, which uses fuzzy inference systems to compute a final trust score from multiple input factors [42]. The system applies rule-based reasoning, with triangular membership functions that convert crisp values into fuzzy linguistic terms like "low" or "high" and back again via defuzzification. A fuzzy rule-based model that includes user behaviour tracking and resource specifications like response time was discussed in [43]. Their trust management system includes distinct modules for users and service entities. The model evaluates trust for both users and CSPs, using monitoring logs and behaviour analysis, with fuzzy logic applied to assess elasticity and performance. Table 5 shows the key factors often used to calculate user trust across these models.

Factors	Definition	Importance	Example
Security	Protection against unauthorized access, data breaches, and malicious actions.	Indicates how responsibly users behave during access attempts.	A user consistently accessing only permitted data shows high security behaviour.
Privacy Compliance	Adherence to data usage, storage, and sharing policies.	Shows respect for handling sensitive data, reducing privacy risk.	A user never tries to export or misuse personal information fields.
Performance	Consistency in response time, query efficiency, and workload handling during access.	Reflects efficient and non-disruptive user interactions with the system.	A user whose queries execute within normal processing time is trusted more.
Reliability	Frequency of successful versus failed or abnormal access attempts.	Higher reliability builds trust over time by showing stable behaviour.	A user rarely causing access errors (e.g., no 400 or 401 codes) is more reliable.

Dynamicity	Ability to maintain trustworthy behaviour even when roles, environments, or conditions change.	Evaluates if the user remains consistent under varying contexts.	A user shifting to a new department still follows policies properly.
Data Integrity Compliance	Ensuring no unauthorized modification or tampering with data.	Protects system integrity by maintaining correct data usage.	A user who edits only authorized fields and does not corrupt records.
Behavioral Monitoring (HTTP Errors)	Tracking frequency of bad requests (400), unauthorized (401), forbidden (403), and not found (404) errors.	Helps detect malicious or suspicious users based on request patterns.	A user repeatedly causing 403 forbidden errors may have low trust.

Table 5 User trust factors details

While several models also evaluate trust at the cloud service provider level (e.g., based on SLA or infrastructure reliability), this study focuses exclusively on user trust derived from behavioural patterns, access logs, and rule-based analysis.

Regarding common attacks, ABAC faces unique vulnerabilities, particularly attacks involving attribute manipulation or policy exploitation. Rubio-Medrano et al. introduce a specific type of vulnerability in ABAC systems: attribute-forgery attacks [44]. In such attacks, malicious actors compromise the sources responsible for generating or managing attributes, intentionally altering attribute values to bypass ABAC policies. These attribute-forgery vulnerabilities can allow unauthorized access to sensitive resources if the ABAC model lacks effective mechanisms to verify attribute integrity or source reliability. The authors propose a risk assessment tool called RiskPol, which dynamically assigns trust scores to attribute sources, thus mitigating forgery risks by proactively identifying and addressing vulnerability in attribute generation and delivery processes. Further, Morisset et al. [45] identify vulnerabilities for evaluating missing attributes in ABAC systems, highlighting attribute-hiding attacks. In such scenarios, users or attackers

deliberately conceal attribute values that lead to unfavourable authorization decisions. Standard ABAC systems might incorrectly interpret incomplete attribute data, potentially granting unwanted access. So, an extended evaluation method is proposed, which examines all possible query extensions. This approach counters attribute-hiding by thoroughly exploring attribute presence or absence, thus reducing the chance of exploitation through incomplete or misleading attribute information. However, a major drawback of their method is the computational complexity and the overhead introduced by examining extensive attribute query spaces. In a broader analysis, Policy Gap vulnerabilities also emerge in complex ABAC scenarios. These occur when authorization policies fail to account adequately for domain-specific constraints or environmental contexts, making them vulnerable to exploitation. For instance, overly permissive or inadequately constrained ABAC rules can lead to unintended policy conflicts or implicit attribute dependencies, potentially enabling indirect unauthorized access.

Insider threats can also pose vulnerabilities in ABAC, as described in [46]. Insiders, leveraging knowledge of internal policies and attribute assignments, manipulate or forge attributes, create unauthorized attribute-to-entity assignments, or deliberately alter policy rules. The proposed framework dynamically mutates ABAC policies by identifying and substituting correlated attributes from new access requests, making insider attacks harder by ensuring attackers cannot predict the altered policy logic [46]. This Moving Target Defense (MTD) strategy leverages attribute variability to continuously evolve the policy set. The authors combine ABAC with MTD and deception techniques, such as “honey attributes,” to mislead malicious insiders and increase the attacker's operational costs. Their method effectively addresses insider threats but may inadvertently complicate legitimate access management due to the constant modifications of access policies. This dynamic nature, while powerful, introduces administrative overhead and

could inadvertently affect legitimate access workflows, complicating auditing and compliance tracking in regulated industries.

Similarly, Abduhari et al. present a comparative analysis of vulnerabilities and gaps in three access control mechanisms: RBAC, Multi-Factor Authentication (MFA), and Strong Passwords [47]. Although strong passwords are widely used, current best practices recommend moving towards passphrases and passwordless methods due to risks like reuse and brute-force attacks. Multilingual passphrases offer improved security through increased complexity [48], while modern systems adopt passwordless authentication, such as biometrics and passkeys, for enhanced resilience [49]. It uses both literature support and simulation through the Access Control Simulation Environment (ACSE). The study evaluates how each mechanism performs against common attacks such as phishing, brute force, and privilege escalation, as detailed in Table 6. Their findings show that MFA offers the strongest defence, particularly against brute force and phishing, by requiring multiple verification steps. RBAC performs moderately, especially in internal role misuse, but is less effective against external threats and privilege escalation when roles are overly broad. Strong passwords are the least reliable, as they can be compromised through reuse, weak patterns, or dictionary-based brute-force attacks. Although the paper does not directly cover ABAC (Attribute-Based Access Control), it notes the need for more flexible, adaptive models that consider context and real-time factors, which are the core strengths of ABAC. This encourages innovative solutions such as attribute trust, which can be evaluated using assigned weights and “isEssential” flags to validate critical attributes. It can also reduce policy overload by separating logic, i.e., XACML handles fixed rules, and NGAC handles dynamic trust-based evaluation. Brute force attacks are managed via historical context tracking and optimal model constants.

Access Control Mechanism	Key Strength	Typical Vulnerabilities / Attacks
RBAC	Role-based structure	Role misuse, Privilege escalation, Static roles, No context-awareness.
MFA	Multi-layer verification	SIM swapping for SMS-based authentication, MITM (man-in-the-middle) attacks, and device compromise.
Strong Passwords	Basic identity check	Brute-force, Reuse, Dictionary attack.
ABAC [44], [45], [46]	Contextual, fine-grained control	Attribute forgery and hiding, Policy or attributes overload, Evaluation delay, Privilege escalation by insider threats.

Table 6 Vulnerabilities in access control mechanisms [47]

2.5. Industry Applications based on ABAC access control

As identity management becomes increasingly central to secure access in distributed systems, cloud-based platforms like Microsoft Entra ID (formerly Azure Active Directory) have gained importance for integrating attribute-based and context-aware access control mechanisms. Microsoft Entra ID supports Conditional Access (CA) policies, which apply real-time contextual parameters, such as user identity, device compliance, location, session risk, and sign-in behaviour, to determine access decisions [4], [32]. These features enable organizations to enforce adaptive policies that go beyond static role assignments, aligning access control with dynamic security postures. Microsoft's platform evaluates “user risk” and “sign-in risk” using signals aggregated from numerous sources, including atypical sign-in patterns, anonymous IP addresses, malware-linked infrastructure, and leaked credentials. Based on these factors, risk levels are categorized as low, medium, or high. A low risk signifies routine activity, while medium and high risks indicate potential anomalies or compromise. Although this approach enhances real-time threat mitigation, the underlying logic for risk scoring remains opaque and non-

customizable, raising concerns regarding auditability and explainability of access decisions. This is partly due to proprietary considerations, i.e. designed to protect Microsoft's competitive edge and, potentially, to prevent reverse engineering or misuse. However, this lack of transparency hinders auditability and explainability. As shown in Figure 8, Entra ID's strengths are highlighted by its position as a leader in the Gartner Magic Quadrant for Access Management, maintaining this status for eight consecutive years, as of 2024 [50]. Its widespread adoption is recognized for strong identity governance features, including role-based access control (RBAC), privileged identity management (PIM), and application-level access control via Access Packages [4]. These features simplify identity lifecycle management and reduce manual administrative overhead, particularly in hybrid and multi-cloud environments. The platform also facilitates integration across a range of Microsoft services (e.g., Microsoft 365, Azure App Service, Logic Apps) and third-party applications, offering single sign-on and centralized policy enforcement. Furthermore, attribute-based conditions are leveraged to dynamically add or remove users from security groups, supporting granular access control configurations.

Figure 8 Magic Quadrant for Access Management [50]

Despite recent advancements, Microsoft Entra ID continues to exhibit several limitations. While Conditional Access policies and Continuous Access Evaluation (CAE) now offer improved session-level enforcement, they are still tied to predefined triggers and vendor-controlled logic. Although dynamic role assignment via user or device attributes has been introduced, Entra ID lacks native support for delegation, customizable trust scoring, or extensible risk evaluation. These limitations reduce its effectiveness in environments requiring continuous, context-sensitive, and domain-aware access decisions, particularly when factoring in dynamic changes such as temporary location shifts, behavioural anomalies, or evolving organizational policies.

In identity governance scenarios, third-party tools like DynamicSync by FirstAttribute AG help bridge hybrid environments by synchronizing on-premises Active Directory group memberships with Entra ID based on real-time directory attributes [32]. This allows dynamic group updates without manual intervention and simplifies administrative overhead in enterprises where on-premises AD remains the primary identity source. However, these tools manage directory state rather than enforce runtime access decisions. Group membership is usually evaluated at token issuance, not continuously during a session. In environments requiring highly dynamic or context-sensitive decisions, such solutions may lack the necessary granularity and responsiveness in environments requiring highly dynamic or context-sensitive decisions. Given these observations, Platforms like Microsoft Entra ID and DynamicSync serve primarily as identity orchestration and policy enforcement layers rather than full-fledged access control engines. While they offer robust tools for identity lifecycle management, such as group automation, risk-based signals, and compliance checks, they operate within fixed policy evaluation frameworks. Furthermore, Entra's Conditional Access policies already support resource targeting such as cloud apps, actions, and authentication contexts. However, it is underutilized in dynamic risk alignment because the CA policy, while it can target dynamic groups based on Entra attributes, remains limited in flexibility and focused purely on supported user attributes. It also lacks either fuzzy logic or trust weighting. Customization of internal scoring models is limited, making it difficult to align risk-based decisions with domain-specific contexts. There is also a lack of support for evaluating partial policy matches, assigning attribute weights, or enforcing essential attribute conditions. Furthermore, decision transparency is minimal: access logs indicate outcomes but do not explain which attributes influenced the result

or how risk levels were computed. This constrains auditing, debugging, and fine-tuning access control, particularly in sensitive domains like healthcare, finance, or academic research.

The Open Policy Agent (OPA) introduces a more flexible “policy-as-code” framework through its Rego language, enabling cloud-native, microservice-driven environments to enforce custom, fine-grained access control [51]. OPA allows developers to inject external data sources for dynamic evaluation, making it well-suited for stateless and decentralized applications. All attributes are treated equally unless additional logic is hand-coded. Furthermore, OPA lacks built-in visualization or simulation capabilities, which limits its ability to model and tune access strategies over time.

Enterprise tools like ForgeRock Access Management and the Auth0 Rules Engine provide more extensive policy scripting and dynamic context integration [52], [53]. They support adaptive authentication, device fingerprinting, and risk-based flows. However, these platforms are often infrastructure-heavy, require complex setups, and rely on a black-box approach to dynamic signal evaluation. That is, while administrators can define rules and scripts, the internal logic used to interpret signals (such as how risk levels are computed or how multiple conditions are weighted) is not exposed or explainable. These systems typically return only the final access decision without revealing how each input contributed, making it difficult to debug, audit, or optimize decisions.

In contrast, white-box models offer transparent evaluation pipelines, exposing intermediate computations, scoring breakdowns, and attribute-level contributions, which are essential for policy debugging and iterative improvements. While ForgeRock and Auth0 offer high scripting flexibility, they do not offer multi-model policy structures (e.g., ABAC + graph-based NGAC

logic) nor provide the internal scoring analytics necessary for system transparency and iterative improvements.

Amazon Web Services (AWS) offers Identity and Access Management (IAM), which supports ABAC via tag-based access control and allows organizations to restrict access based on resource or user tags [54]. However, its policy structure is static and declarative and lacks prioritization of attributes or integration with real-time behavioural signals. IAM policies cannot distinguish between critical and optional attributes and don’t offer mechanisms to incorporate permit/deny history, contextual thresholds, or trust-based adaptation over time. Like other systems, AWS ABAC lacks any notion of attribute scoring, decision explainability, or hybrid policy integration.

Tall and Zou proposed an innovative ABAC framework to manage the security of big data systems, specifically designed for environments like Hadoop and Spark and often used in AWS [55]. Their main goal was to handle data from diverse sources and have multiple sensitivity levels, particularly important in healthcare and social media domains, where privacy and security are critical. They argued that existing cloud platforms typically adopt a "castle wall" security approach. It grants complete access once users enter the perimeter without enforcing strict data-level security measures. To address this vulnerability, Tall and Zou's ABAC framework integrates detailed metadata (attributes) that describe both users and datasets, such as sensitivity levels, origins, and processing history. They demonstrated the feasibility of their framework using open-source tools (Apache Ranger and Apache Atlas) on AWS, providing a practical approach to manage fine-grained security policies dynamically across large datasets. One of their primary contributions is the integration of metadata-driven security policies, where metadata tracks the history and sensitivity of data as it undergoes transformations and anonymization

procedures. This approach ensures security rules remain relevant, even when data characteristics change over time, enabling dynamic decision-making. Furthermore, they highlighted several security risks inherent in big data environments, such as credential hijacking, job submission attacks, and unauthorized access due to misconfigured permissions. The proposed framework addresses these challenges in several ways. For example, unauthorized access is mitigated through the continuous synchronization of attributes between data repositories and the central policy management service, ensuring policies are consistently enforced even during data transformations or analyses. Despite its strengths, the framework also has some limitations. While the model effectively integrates metadata-based decision-making, it heavily relies on correctly defined and accurately maintained attributes, which can be complex and error-prone. The authors acknowledged that poorly defined or overly complex attribute models could lead to inaccuracies in security policy enforcement and cause either overly restrictive access (false negatives) or unintended access permissions (false positives).

Data Access Governance (DAG) tools are useful for spotting overly shared data and helping clean up access permissions. However, according to Atlan, they have limitations with ABAC systems, especially in dynamic, cloud-based setups [56]. While these tools aim to improve access decisions by using metadata and rules, they often can't keep up with the real-time and flexible needs of ABAC. Many DAG tools still rely on fixed role-based models, ignore important context like data history and sensitivity, and don’t handle audit logs well. They also struggle to connect smoothly with other governance tools like data catalogues and compliance systems. So, even though DAG tools are helpful for basic governance tasks, they fall short when it comes to supporting the advanced, context-aware decisions that modern ABAC systems require.

Moreover, the practical implementation within ecosystems like Hadoop often requires extensive manual configuration and continuous synchronization, which could increase administrative overhead and vulnerability to human errors. Another weakness is the operational complexity, where each dataset and process requires continuous metadata management. Without robust automation, maintaining accurate and relevant metadata becomes difficult, especially in large-scale environments, potentially leading to delays or inaccuracies in policy evaluation and enforcement. Additionally, the authors pointed out that the existing security standards, such as XACML, although widely cited, are not directly implemented within Hadoop-based tools, which may limit interoperability and standardization across platforms.

2.6. Summary

Traditional access control models face limitations in dynamic environments such as cloud computing, IoT, and large-scale enterprise systems. They often lack scalability, adaptability, and contextual awareness. Hybrid access control models have emerged to overcome these shortcomings by integrating the strengths of multiple approaches, offering fine-grained, adaptive, and flexible access control with context-aware mechanisms. However, hybrid models can introduce redundancy and complexity, or be too domain-specific. Enhancements like Request Denial Cache (RDC) can reduce evaluation time but require manual updates and still lack dynamic context awareness.

XACML's primary issues include over-restrictiveness, policy authoring complexity, scalability challenges, and insufficient conflict resolution mechanisms in distributed systems. These constraints limit its usability in dynamic domains like healthcare, where rapid, real-time access to decisions is crucial. For instance, a doctor may be denied urgent access to a patient's record

because the policy wasn't updated for emergency roles, highlighting XACML's static nature. NGAC addresses these gaps using a graph-based structure by adding history-aware decisions and can enforce constraints like obligations or usage control. For example, NGAC can ensure that once a nurse has approved a medication, they cannot also dispense it, enforcing separation of duty through graph associations. NGAC is more adaptable than XACML but has its drawbacks, mainly a lack of industry adoption, limited tooling, and complexity in graph administration. Poorly designed graphs can result in over-permissiveness, and migration from legacy systems remains difficult. For example, if a user is mistakenly linked to a high-privilege attribute in the graph, they may gain unintended access. In another case, maintaining thousands of user-object relationships manually can lead to policy inconsistencies or gaps. Trust-based models use fuzzy logic and behavioural data to compute trust scores for users and cloud providers to enhance decision quality. These scores guide access control in uncertain or rapidly changing environments. While effectively refining access decisions, they often face subjective rule settings, scalability concerns, and data dependency issues.

ABAC systems face evolving threats. Attribute forgery and hiding can manipulate access outcomes by altering or omitting attribute values. Tools like RiskPol and extended query evaluation methods help detect such attacks, but often increase computational costs. Insider threats are another concern; users may exploit knowledge of policies or manipulate attribute assignments. Solutions like honey attributes and deception-based defence techniques can mitigate these risks, but add management complexity. Industry platforms incorporate ABAC-like logic, often layering risk signals or tag-based policies over static evaluation engines. These systems lack transparency, real-time adaptability, and support for weighted or essential attribute

logic. Similarly, big data environments offer fine-grained control but depend heavily on consistent metadata and manual configuration, limiting scalability.

The literature clearly supports hybrid approaches. A layered model combining XACML for static policies and NGAC for dynamic evaluation, enhanced by trust scoring, offers the adaptability and precision needed in modern systems. Such integration reduces false decisions, improves scalability, and aligns access control with real-world behaviour, making it suitable for domains like healthcare, cloud platforms, and enterprise security.

Chapter 3

Methodology

This chapter presents the methodological foundation for designing and evaluating a hybrid attribute-based access control framework intended for enterprise environments. The system is designed specifically for internal users, such as employees, faculty, staff, and system administrators, who access applications and services within a controlled organizational infrastructure. These users are not general members of the public, but authorized personnel operating across various internal systems, including cloud services, institutional databases, administrative tools, and secure file-sharing platforms. The nature of their responsibilities often demands fine-grained, dynamic, and context-aware access decisions that go beyond traditional static models like RBAC or simple rule-based ABAC.

To meet these needs, the proposed framework integrates and extends concepts from existing models such as XACML and NGAC, while introducing several key innovations that address the limitations of prior approaches. The central logic is the introduction of Attribute-Rule Weighting, where each subject or object attribute is not only defined by its presence or absence (match) but is also assigned a weight representing its criticality or influence on access decisions. These weights help determine how strongly a given attribute influences the trustworthiness or sensitivity classification of a request. While models like AR-ABAC propose grouping attributes by average weights to map them to roles or sensitivity levels [41], the hybrid framework advances this idea by directly integrating these weights into runtime decision logic. Instead of merely assigning a role based on attribute grouping, the system computes a trust factor that reflects both attribute match constraint and behavioural history. Another important part is the use

of a risk-adjusted access score, which combines historical access outcomes with contextual risk levels. It penalizes users differently based on their recent access behaviour and associated risk classification. Here, permitted and denied access attempts are tracked over time, and denials are weighted more heavily for high-risk users using a risk multiplier. This ensures that two users with the same historical access patterns can still be treated differently based on their risk posture, so adding a dynamic layer of accountability and precaution to the access control process. This research introduces original formulas (from Equations (1)-(9)) derived at various stages of the evaluation process; each developed specifically to support and justify the decision-making logic proposed in this work. These step-by-step calculations guide the access evaluation and ensure each phase can be transparently validated. Further, the model's effectiveness is evaluated using four performance metrics: accuracy, precision, recall, and F1-score, which collectively assess decision correctness and reliability. Their detailed calculation and application are presented in Chapter 4 (section 4.4.2).

The framework adopts a layered architecture in which static attributes (such as department, job role, or assigned project) are handled using conventional XACML policy rules, while dynamic context (such as device, time, or geolocation) is evaluated using graph-based relationships from the NGAC design. These relationships are stored and managed within a relational database structure that represents graph edges and hierarchies, allowing us to dynamically compute permissions based on indirect or inherited associations. For example, an attribute such as "assigned to project X" may grant access not only to Project X resources but also to related datasets, if permitted by contextual policy rules or relationship mappings. This hybrid structure enables both direct and inferred authorizations to be processed efficiently and transparently.

To ensure that the model is not only theoretically sound but also practical for deployment and testing, a comprehensive framework for scenario generation and categorization was developed. Access scenarios are generated using structured techniques rather than simple cross-product combinations. The process begins with policy-driven filtering to ensure only users with plausible attribute-policy matches are considered. From there, FSM (Finite State Machine) based path logic simulates transitions from attribute assignment to trust evaluation to permission requests, forming realistic access flows. Negative cases are programmatically injected by mutating key attributes in otherwise valid scenarios, mimicking adversarial behaviour. Additionally, trust scores are varied across users to simulate contextual changes and behavioural history, allowing dynamic evaluation through a risk-weighted trust formula. Scenarios are categorized into valid access, attribute violations, contextual drift, adversarial mutations, ambiguous overlaps, and behavioural anomalies. These diverse, high-impact cases are then used to test the model's correctness and consistency. In particular, the trust-weighted access factor serves as a continuous variable that reflects the system's confidence in a given access request. Decisions are logged alongside this score for auditing purposes and can be visualized in dashboards to help administrators understand why certain access was granted or denied. Over time, this enables feedback loops and trust recalibration based on longitudinal patterns, further enhancing the adaptiveness and intelligence of the access control system.

Unlike static rule-based models, this framework supports fine-grained differentiation even among users who share identical roles or attributes. Since the evaluation logic incorporates a combination of attribute weights, behavioural data, and contextual risks, access decisions adapt to changing internal operations. For example, a staff member working remotely under a known low-risk profile may be granted access to certain sensitive files, while another staff member with

similar attributes but elevated risk and recent denial patterns may be restricted or flagged for secondary review.

Figure 9 illustrates the core components of the complete proposed model described in the following sections. It integrates with Microsoft Entra ID as the authentication and identity provider and third-party tools for policy administration and environment hosting.

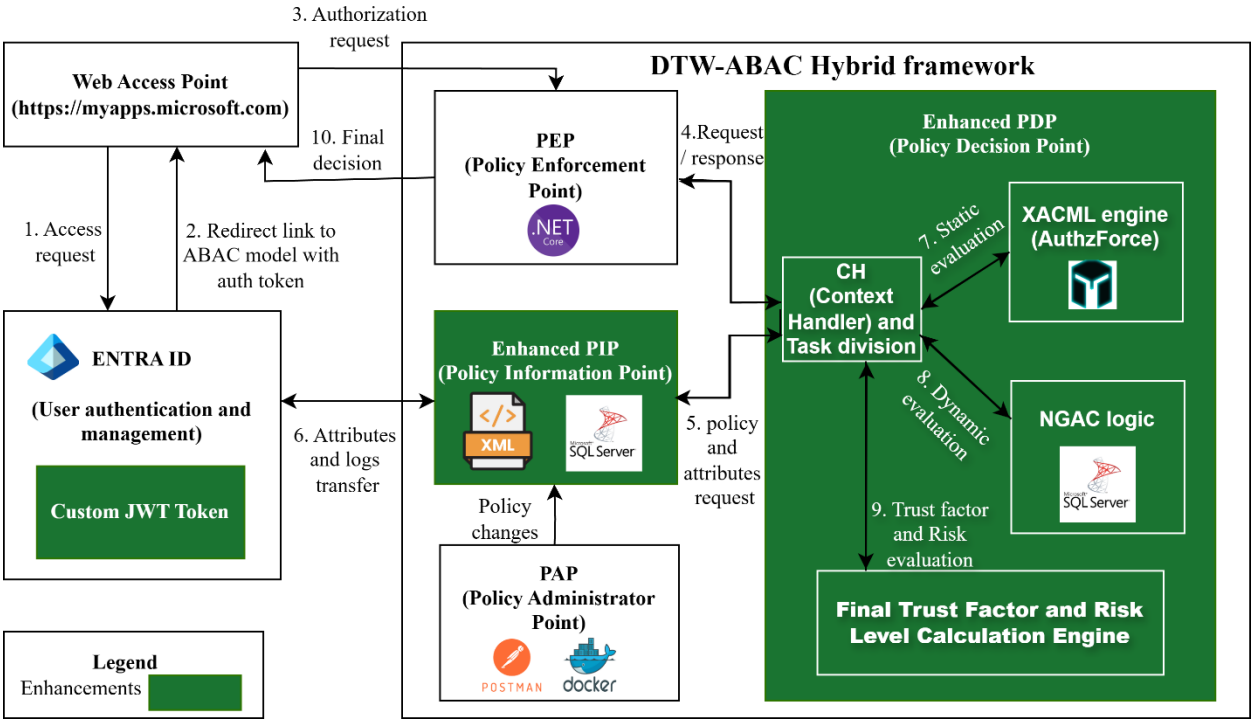


Figure 9 Dynamic trust weighted-attribute based access control architecture

Based on the architecture diagram in Figure 9, here is a concise summary of Steps 1 to 10 with their corresponding components:

1. **Access request (Web access point → Entra ID)** – User initiates request to access a protected application.
2. **Redirect link to ABAC model with auth token (Entra ID → Web app)** – Entra ID authenticates and returns a token to the application (OAuth 2.0 protocol).

3. **Authorization request (Web access point → DTW-ABAC)** – Application forwards the request to the ABAC engine for authorization.
4. **Request/response (PEP ↔ PDP)** – Policy Enforcement Point (PEP) and Policy Decision Point (PDP) exchange the access request and final decision.
5. **Policy and attribute requests (PDP ↔ PIP)** – PDP queries the PIP for required policies and attributes specific to the user and application IDs (details retrieved from Entra ID token).
6. **Attribute and log transfer (Entra ID ↔ PIP)** – PIP retrieves user attributes and logs from Entra ID for evaluation using assigned permissions to the DTW-ABAC application.
7. **Static evaluation (PDP: CH ↔ XACML)** – Context Handler evaluates static rules using the XACML policy engine.
8. **Dynamic evaluation (PDP: CH ↔ NGAC)** – Context Handler evaluates real-time, context-aware policies via NGAC.
9. **Trust factor and risk evaluation (PDP: CH ↔ Trust/risk engine)** – The Trust engine computes a risk score to refine the access decision based on multiple formulas.
10. **Final decision (PEP → Web access point → redirect to app or deny)** – Based on evaluation, the user is granted and redirected to the application or denied access.

The next part deep dives into the proposed research framework, including the hybrid model architecture, software tools, research dataset, and key challenges encountered.

3.1. Entra ID custom configuration for authentication and attributes source

Dynamic Trust Weighted-Attribute Based Access Control (DTW-ABAC) is built along with the foundation provided by Microsoft Entra ID, which offers advanced user identity and access management features, eliminating the need to develop a new registration and authentication

process. Although it provides data control for Azure resources and application control features, this research will be limited to securing access for registered applications developed by the organization, any Microsoft 365 applications (can be added as enterprise applications) or third-party enterprises. The required licenses (P2 or formerly known as AAD P2 premium license) were purchased to use the admin portal and governance features. P2 was necessary because the hybrid model relies on dynamic risk-based decisions, access reviews, and identity protection signals, all of which are not available in the P1 license [4]. Features like user risk evaluation, sign-in risk, and conditional access behaviour based on behaviour or trust are critical for the hybrid model's real-time enforcement logic, and these are only included in Entra ID P2. P1 supports basic conditional access but lacks the advanced context and automation required. P2 license is user-specific, which means an organization needs to purchase the same number of licenses as the number of users and administrators. It creates a new user account and tenant domain for an active Azure account during registration. The new account credentials were used to perform access control operations on the admin portal and integrate them into the programming code to access the resources from an executable environment. Although this research utilizes Entra ID, it can be integrated with any platform that offers authentication and user registry capabilities.

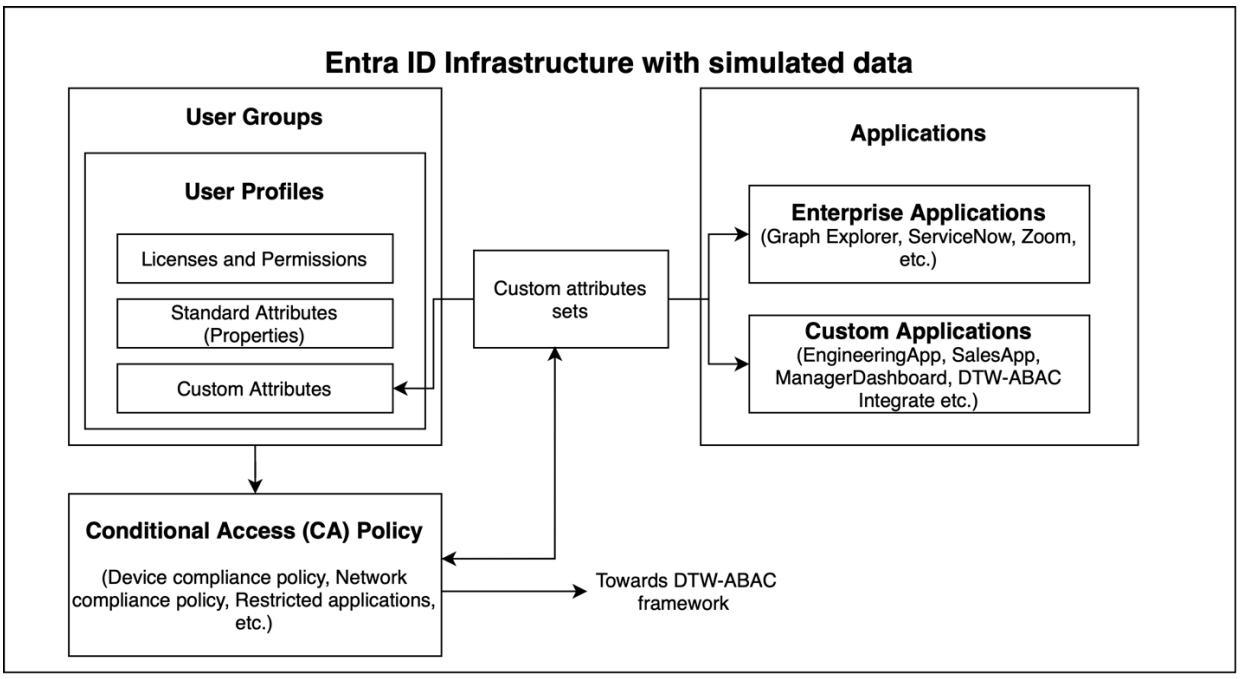


Figure 10 Entra ID test infrastructure with simulated data

Figure 10 shows the detailed infrastructure in Entra ID. Multiple users were created in the user management section, and simulated profile data and licenses were assigned. This follows multiple user group creation in the group management section based on conditional expressions, also called dynamic membership rules. Each conditional expression consists of multiple user attributes and defined values for evaluation. These expressions are then used to build dynamic Microsoft 365 groups, which are further organized using the "memberOf" attribute to form nested group hierarchies that mimic real-world organizational structures. However, this approach is constrained by key limitations: only up to 500 dynamic groups can use the "memberOf" attribute, each group can include up to 50 member groups, and nesting is not recursive, limited to a hierarchy depth of two levels. Additionally, indirect members are not resolved, and backups or advanced management require third-party solutions to flatten or replicate the nesting structure reliably. Similarly, enterprise and custom-built applications were registered in the application management section with properties, such as homepage URL, visible to user flag, enabled for

sign-in flag, etc. Those applications are considered the target resources for which the overall access control environment is created. In the same section, an application named 'DTW-ABAC integrate' was registered to connect with the hybrid framework, with associated permissions as described in Table 7 to access the attributes and provisioned resources using the Microsoft Graph API [4]. Usually, Entra ID allows you to set the permissions as either delegated access or the application context. The delegated access (user context) is where the application acts on behalf of a signed-in user, and it is used in web apps or mobile apps where a user is present and signed in. The application access (app context) is where the app acts as itself, without a user and is used in the background services, automation scripts or APIs that run without user interaction. Hence, in this architecture, the permissions are assigned at the application access level.

Scope	**Permission name**	**Used For**
Read user info	User.Read.All	Pulling identity/profile attributes
Read custom attributes	Directory.Read.All	Accessing organization-wide directory data such as users, groups, roles and custom security attributes
Read app metadata	Application.Read.All	App-based policy or attribute links
Read app role assignments	AppRoleAssignment.Read.All	User-to-app mappings
Read audit logs	AuditLog.Read.All	NGAC context from logs
Read the sign-in context	SignInActivity.Read.All	Trust factor/risk condition evaluation

Table 7 Microsoft Graph permissions for the "DTW-ABAC Integrate" application

The integration with Entra ID is strictly read-only, i.e., no write permissions are granted to avoid introducing additional risk to the identity source. The dynamic state (e.g., session risk, behavioural drift, NGAC graph assignments, trust scores) is maintained entirely outside of Entra,

within the custom application database designed for the DTW-ABAC model. Static attributes such as roles, group memberships, and custom security attributes are fetched from Entra and cached if needed, but not modified. The external database is fully encrypted at rest using Transparent Data Encryption (TDE), which encrypts the entire database, including backups and transactional logs. The framework relies on a rich and diverse set of attribute data to evaluate the access request with high context and granularity. The assigned properties for users and applications are considered standard attributes during the access evaluation. They also support assigning the custom attributes created in the attribute sets. Table 8 shows the user (subject), application (object) and Environmental attributes with sample values imported from Entra ID using Graph API access. This model uses custom security attributes in Microsoft Entra ID to support fine-grained access control and trust evaluation. These attributes are not part of the default user schema and are managed under the Custom Security Attributes feature in Entra ID, allowing tenant-specific definitions. They follow the camel case style and are used to represent application-specific roles (e.g., devRole, projectAccessLevel), contextual metadata relevant to decision logic (e.g., isRemoteEnabled, regionAffinity) and support trust factor calculations based on historical or environmental conditions (e.g., lastPolicyViolation, loginConsistencyScore). The Attribute type includes string, integer, date, floating number, or Boolean values.

Entity	Attribute type	Attribute name	Sample values
User (Subject)	**Standard**	Object ID	9abc098e-4546-4d92-95fd-567fcd51d9f9
		User principal name (Email)	user1@birg.onmicrosoft.com
		User type (Member, Guest)	Member
		Job title	Associate Engineer
		Company name	UNBC
		Department	Computer Science
		Employee type (Part-time, Full-time, Temp-contract)	Full-time

		Office location	COPG
		Manager	{"jobTitle": "Engineer","mail": "kulkarnis009@gmail.com","officeLocation": "Prince George"}
		City	Prince George
		State or Province	BC
		Country	Canada
		Usage location	CA
	Custom	devExperience	3
		isRemoteEnabled	TRUE
		appWriteStatus	Approved
		projectName	DTW-ABAC
		projectClearance	Confidential
		subRole	PDP-designer
Application (Object)	**Standard**	Application ID	b636f32f-c36b-4756-8251-6747539a0688
		appRoles	{"allowedMemberTypes": ["User"],"description": "User","displayName": "User","id": "18d14569-c3bd-439b-9a66-3a2aee01d14f","isEnabled": true,"origin": "Application", "value": "Survey.Create"}
		web/redirecturi	"uri": "https://localhost:9001/api/access/authorize",
		publisherDomain	birgsk.onmicrosoft.com
	Custom	appCofidentialLevel (High, Medium, Low, NA)	High
		isRemoteEnabled	TRUE
		accessDepartment	Computer Science
		appEnvironment (Prod, Test, QA, Dev)	Prod
		securityClearance	Confidential
Environmental (fetched from Token, Sign-in logs)		ipAddress (User)	142.207.116.128
		createdDateTime	Fri Jan 3 2025 13:10:08 GMT-0700
		deviceDetail.isCompliant	TRUE
		deviceDetail.isManaged	TRUE
		deviceDetail.deviceType	Mac
		deviceDetail.browser	Chrome

Table 8 Subject, object and environmental attributes with sample values

During the authentication process, conditional policies are configured to be applied before routing it towards the hybrid model. These policies enforce an enterprise-level wide access rule,

such as network-related and device-related restrictions, and grant or deny the request with additional steps like MFA or password change requirements. Once passed through these steps, the user can see all the assigned applications in the MyApps portal (Figure 11). At this step, the authentication process is completed.

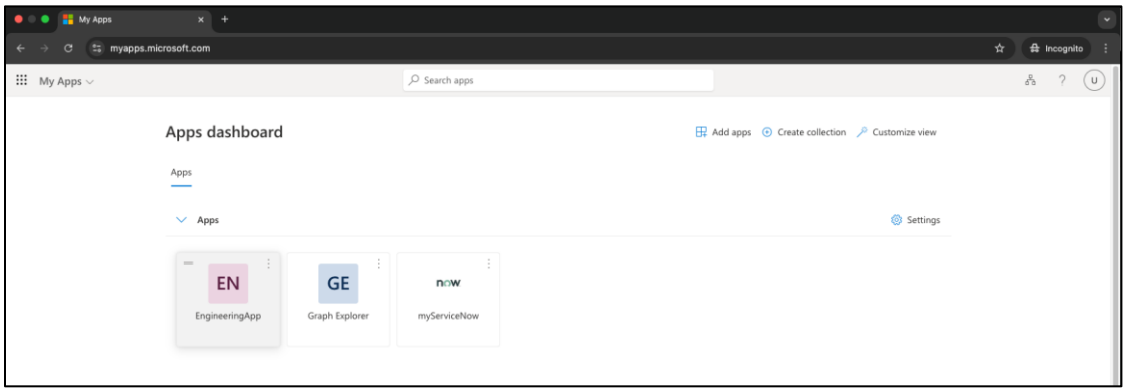


Figure 11 Entra ID apps dashboard

Upon selecting a particular application to access, the JWT (JavaScript Object Notation Web Token) is issued by Microsoft Entra ID, which carries essential information (claims) about the user and the authentication context. The process is similar, even if the user directly opens the application (e.g. myServiceNow) instead of using the MyApps portal; the redirect link to DTW-ABAC is added to the application level. The dashboard is just a medium to see all the available applications (based on the Entra ID CA policies and assignments). The token structure consists of three parts: a header with RSA256 (Rivest-Shamir-Adleman algorithm using SHA-256, a 256-bit hash function) as a signing algorithm, a payload containing claims (such as user identity and roles), and a digital signature that ensures the integrity and authenticity of the token. The token ensures both identity validation and secure API access by parsing and validating on the server side before granting or denying access to protected resources. Table 9 shows the decoded structure of a custom JWT issued during the configured authentication setup of this research, reflecting key claims and signature metadata. Its critical properties like *aud*, *iss*, and *exp* are

validated to ensure the token is intended for the correct API, issued by a trusted authority, and is within the valid time window. The preferred_username and *oid* claims map the user's identity to the internal application user database for authorization. The token is passed to the redirect link set to the selected application. In this setup, the redirect link is set to *http://loclhost:9001/api/access,* which is basically routing to the DTW-ABAC model running on the localhost (port 9001). In the production environment, it can route to the model hosted in the cloud environment.

Claims	Key	Values	Explanation
JWT Header (Decoded)	*typ*	JWT	Type of token, indicating it is a JSON Web Token.
	alg	RS256	Signing algorithm used – RSA with SHA-256.
	kid	CNv0OI3RwqlHFEVnaoMAshCH2XE	Key ID used to select the correct public key for signature validation.
JWT Payload (Decoded Claims)	*aud*	b0383a20-1483-4bfb-b67f-5dffd4e578b3	Audience – identifies the application for which the token is issued.
	iss	https://login.microsoftonline.com/e3d59969-60f3-4913-adf4-5c1983159829/v2.0	Issuer – confirms Microsoft Entra ID issued the token.
	iat	Fri Apr 25 2025 13:10:08 GMT-0700	Issued At – timestamp when the token was generated.
	nbf	Fri Apr 25 2025 13:10:08 GMT-0700	Not Before – token is valid from this timestamp onwards.
	exp	Fri Apr 25 2025 14:15:08 GMT-0700	Expiration – token becomes invalid after this time.
	name	User1	Display name of the authenticated user.
	preferred username	user1@birgsk.onmicrosoft.com	Primary login identifier for the user.
	oid	9abc098e-4546-4d92-95fd-567fcd51d9f9	Object ID – uniquely identifies the user in Entra tenant.
	sub	j9dw5qOlf3uwLEN4lI8hoIX8SmK8YZGNjUPFBSHOC_k	Subject – unique principal for which the token was issued.
	tid	e3d59969-60f3-4913-adf4-5c1983159829	Tenant ID – identifies the Entra tenant (organization).
	ver	2	Token version – confirms this is version 2.0 format.
	kty	RSA	Key type – RSA encryption used.

Public Key (Used to verify signature)	*n*	hz6fUSCSAuiyQz6L1nQj4za8kItevJzxhVbecMigTIl9pXZSHZa3gzMgtapnb1q96CG5qvR78dH6ZvTKL8MzN4VfGgZhvLEv5LJKeo0tGgBIS65wxIiJYj9ExEDqFkw9RdhW1nN8IN9eO76PbC-fdEPtDekA2BaITY2DARISKN4Ke0RLBEWNrKeEjjOzrygS2e3Q9NVzE51ZGGQAGHau7atHy8M_qA1nnd2dMUgUMnEYIMzDBTSKz17G6itJOdanGvG3wXvdpndKffnDppaPkyWbnybdMI4IP7q6WsCqnt3Gtg-baG6GDqZQQEBp9C9gLAFv4ORTRlpD3w0gCMh7xw	Public modulus – a large base number used in RSA encryption; part of the public key, combined with *e* to verify signatures.
	e	AQAB	Public exponent – typically a small, fixed number (e.g., 65537). AQAB is the Base64 encoding of 65537, commonly used in RSA.

Table 9 Decoded structure of a custom JWT configured in Microsoft Entra ID with key claims and signature metadata

3.2. Dynamic Trust Weighted Attribute-Based Access Control hybrid framework (DTW-ABAC)

The proposed framework aims to combine static attribute checks (XACML-based) and dynamic context-aware logic (NGAC-based) with historical access patterns, along with introducing attribute weights and trust scoring mechanisms. The design follows a RESTful design, which enables the scaling capability, higher security, easy integration and maintainability [57]. This hybrid approach enables more adaptive and trustworthy access decisions, especially suited for sensitive domains such as healthcare and education. The framework is composed of multiple distinct modules designed to perform specific operations and is built using different programming frameworks and third-party tools. The modular architecture includes PAP (Policy Administration Point), PIP (Policy Information Point), PDP (Policy Decision Point), CH (Context Handler), final trust factor, risk level calculation engine and PEP (Policy Enforcement Point) as shown in Figure 9. The methodology introduces formulas developed in this research

(from Equations (1) - (9)) to systematically support and justify the proposed decision-making logic. These calculations guide access evaluation while ensuring each stage remains transparent and verifiable.

3.2.1 Policy Administration Point (PAP)

PAP is responsible for managing access control policies that form the foundation of static attribute evaluation in the DTW-ABAC framework. These policies follow the XACML standard and are consumed by the XACML engine to validate attribute-based access rules. In this implementation, policy creation and updates are carried out using the Postman tool, which serves as an external administrative interface. Postman is used to manually design and send HTTP requests that define and manage policy domains and their associated rules. These requests are sent to the PIP, specifically, the AuthzForce XACML engine, which is containerized and hosted using Docker. The process typically involves two steps:

1. Domain Creation:

   An HTTP POST request is made to the AuthzForce endpoint to create a new policy domain. This domain acts as a container for one or more policy sets. The request includes:

   - Request URL and Headers: This request initiates domain creation (Post) by calling the domains endpoint running in a Docker container (localhost, port 8080) with the required XML headers.

```
Post -> http://localhost:8080/authzforce-ce/domains
Headers -> Accept: application/xml Content-Type: application/xml
```

- Body: The XML body includes the domain ID (DTW_ABAC_Domain) via the `externalId` property and complies with the expected schema.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<!-- Creates a new policy domain in AuthzForce with the ID 'DTW_ABAC_Domain' -->
<domainProperties xmlns="http://authzforce.github.io/rest-api-model/xmlns/authz/5"
externalId="DTW_ABAC_Domain"/>
```

- Response: It returns a unique internal domain ID (href), i.e. `hRDKOyleEfCe1AJCrBEAAw`, which is used to upload policies to the domain.

```
<?xml version='1.0' encoding='UTF-8'?>
<!-- Link to the created policy domain in AuthzForce -->
<ns4:link xmlns:ns6="http://authzforce.github.io/pap-dao-flat-file/xmlns/properties/3.6"
xmlns:ns5="http://authzforce.github.io/rest-api-model/xmlns/authz/5"
xmlns:ns4="http://www.w3.org/2005/Atom" xmlns:ns3="urn:oasis:names:tc:xacml:3.0:core:schema:wd-
17" xmlns:ns2="http://authzforce.github.io/core/xmlns/pdp/8" rel="item"
href="hRDKOyleEfCe1AJCrBEAAw" title="hRDKOyleEfCe1AJCrBEAAw"/>
```

2. Policy Upload:

   Once the domain is created, another HTTP PUT or POST request is sent to attach a policy document with the domain ID, e.g. `hRDKOyleEfCe1AJCrBEAAw`.

```
Post -> http://localhost:8080/authzforce-ce/domains/hRDKOyleEfCe1AJCrBEAAw/pap/policies
Headers -> Content-Type: application/xml
```

- The body of the request is an XACML-compliant XML structure containing Rule definitions, Target conditions (based on subject, resource, action, and environment attributes) and Effect (Permit or Deny).

```
<?xml version="1.0" encoding="UTF-8"?>

<!-- PolicySet: top-level container defining access control scope -->
<PolicySet xmlns="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17"
        PolicySetId="root"
        Version="1.0"
        PolicyCombiningAlgId="urn:oasis:names:tc:xacml:3.0:policy-combining-algorithm:deny-unless-
permit">

  <!-- Description of the policy set -->
  <Description>PolicySet to manage access to EngineeringApp using static attributes</Description>

  <!-- Target of the PolicySet is empty = applies to all requests -->
```

```
 8.    <Target/>
 9.
       <!-- Begin individual policy -->
10.    <Policy PolicyId="EngineeringAppAccessPolicy"
11.          Version="1.0"
12.          RuleCombiningAlgId="urn:oasis:names:tc:xacml:3.0:rule-combining-algorithm:deny-unless-
permit">
13.
         <!-- Policy applies only when the resource matches 'EngineeringApp' -->
14.       <Target>
15.          <AnyOf>
16.             <AllOf>
17.                <Match MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
18.                   <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">EngineeringApp</AttributeValue>
19.                   <AttributeDesignator Category="urn:oasis:names:tc:xacml:3.0:resource-
category:resource"
20.                                        AttributeId="urn:oasis:names:tc:xacml:1.0:resource:id"
21.                                        DataType="http://www.w3.org/2001/XMLSchema#string"
22.                                        MustBePresent="true"/>
23.                </Match>
24.             </AllOf>
25.          </AnyOf>
26.       </Target>
27.
         <!-- Rule to permit access if action is 'access' and role is 'Engineer' -->
28.       <Rule RuleId="PermitIfEngineerRoleAndAccessAction" Effect="Permit">

            <!-- Target: applies only to action = access -->
29.          <Target>
30.             <AnyOf>
31.                <AllOf>
32.                   <Match MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
33.                      <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">access</AttributeValue>
34.                      <AttributeDesignator Category="urn:oasis:names:tc:xacml:3.0:action-
category:action"
35.                                           AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id"
36.                                           DataType="http://www.w3.org/2001/XMLSchema#string"
37.                                           MustBePresent="true"/>
38.                   </Match>
39.                </AllOf>
40.             </AnyOf>
41.          </Target>
42.
            <!-- Condition: subject-role must be 'Engineer' -->
43.          <Condition>
44.             <Apply FunctionId="urn:oasis:names:tc:xacml:3.0:function:any-of">
45.                <Function FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-equal"/>
46.                <AttributeValue
DataType="http://www.w3.org/2001/XMLSchema#string">Engineer</AttributeValue>
47.                <AttributeDesignator Category="urn:oasis:names:tc:xacml:1.0:subject-category:access-
subject"
48.                                     AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-role"
49.                                     DataType="http://www.w3.org/2001/XMLSchema#string"
50.                                     MustBePresent="true"/>
```

```
            </Apply>
        </Condition>
    </Rule>

</Policy>
</PolicySet>

```

The response confirms that the policy with ID `root` version 1.0 was successfully created and registered within the specified domain.

```
<?xml version='1.0' encoding='UTF-8'?>
<!-- Link to the uploaded XACML policy with ID 'root' and version 1.0 in AuthzForce -->
<ns4:link xmlns:ns6="http://authzforce.github.io/pap-dao-flat-file/xmlns/properties/3.6" xmlns:ns5="http://authzforce.github.io/rest-api-model/xmlns/authz/5" xmlns:ns4="http://www.w3.org/2005/Atom" xmlns:ns3="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17" xmlns:ns2="http://authzforce.github.io/core/xmlns/pdp/8" rel="item" href="root/1.0" title="Policy 'root' v1.0"/>
```

The XML-based policies are then stored within the corresponding domain and become active for any subsequent access request evaluations made through the framework.

Additionally, the attributes are categorized as either high-level (static) or low-level (dynamic) based on their modification frequency and scope of impact by the administrator. The goal is to avoid duplicate attribute evaluations and take advantage of XACML (restrictiveness) and NGAC (flexibility and permissiveness) frameworks, as discussed in the section 2.2. It plays a critical role in determining the efficiency and flexibility of access decisions. High-level attributes are those that remain relatively static over time and are essential for strategic access control policies. Since modifying XACML policies requires XML restructuring, domain redeployment, and potential policy evaluation disruptions, these attributes are selected with caution. Examples include a user's job title, department, or an application's Application ID and AppCriticality. Low-level attributes represent contextual, behavioural, or session-specific information. These attributes are more volatile and are updated frequently based on user behaviour, device posture, access location, or audit-derived risk states. Managed by NGAC's graph layer, their inclusion or

exclusion from access logic can be changed by simply updating the graph's edge relationships or attribute nodes. Examples include riskLevel, devExperience, webRedirectUri, or ComplianceCheck.

While PAP does not handle dynamic or real-time environmental data, it plays a critical role in maintaining the baseline access logic based on static attribute verification. Once the policies are authored and uploaded by PAP, they become accessible to the Policy Information Point (PIP) for repeated access evaluations.

3.2.2 Enhanced Policy Information Point (PIP)

The Enhanced PIP coordinates attribute resolution and policy retrieval tasks in the DTW-ABAC framework, working across both XACML- and NGAC-based logic layers. Figure 12 shows the components and connections of an entire operation of the modified PIP block. Although these components are stored separately for security and efficient management, a centralized service called the Policy Retrieval Point (PRP) is referenced from traditional XACML to facilitate repeated access during policy evaluation. The designated storage mechanisms include the AuthzForce server for XACML policies and a Microsoft SQL Server for attribute data and NGAC structures. Specifically, attributes are stored in tabular format, while NGAC graphs are captured using a unified node table and a relation table representing edges.

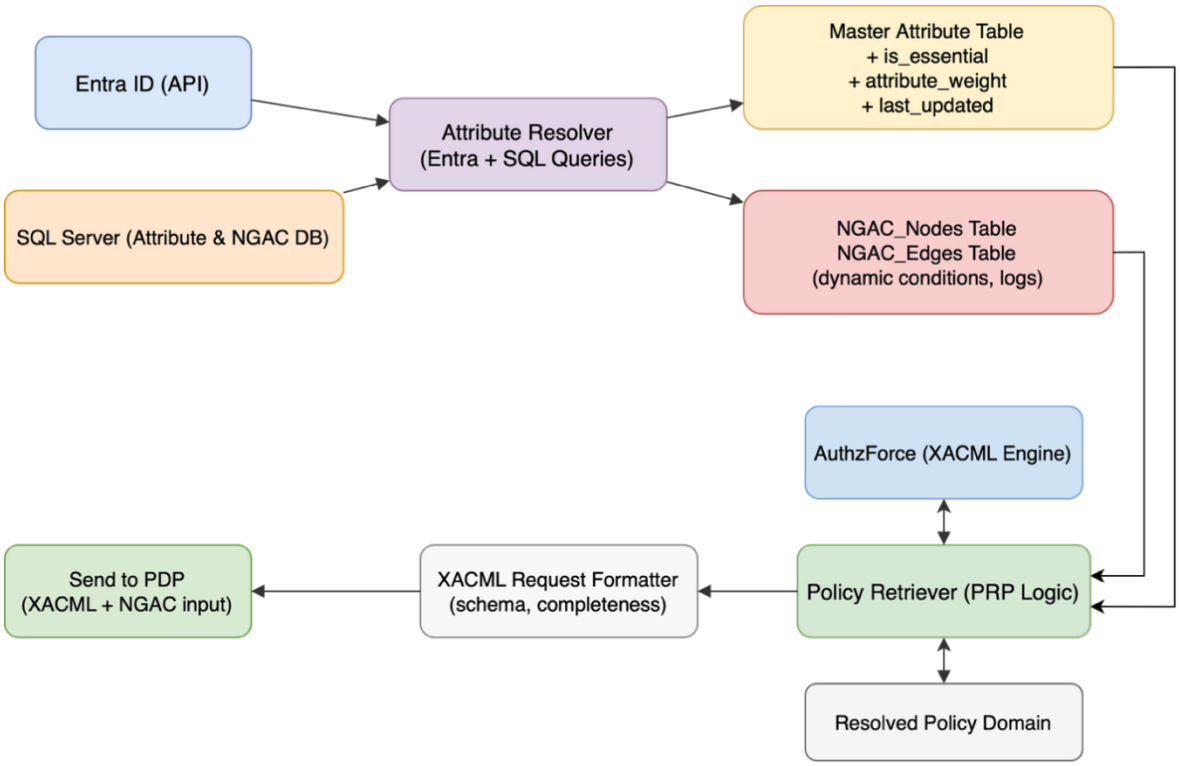


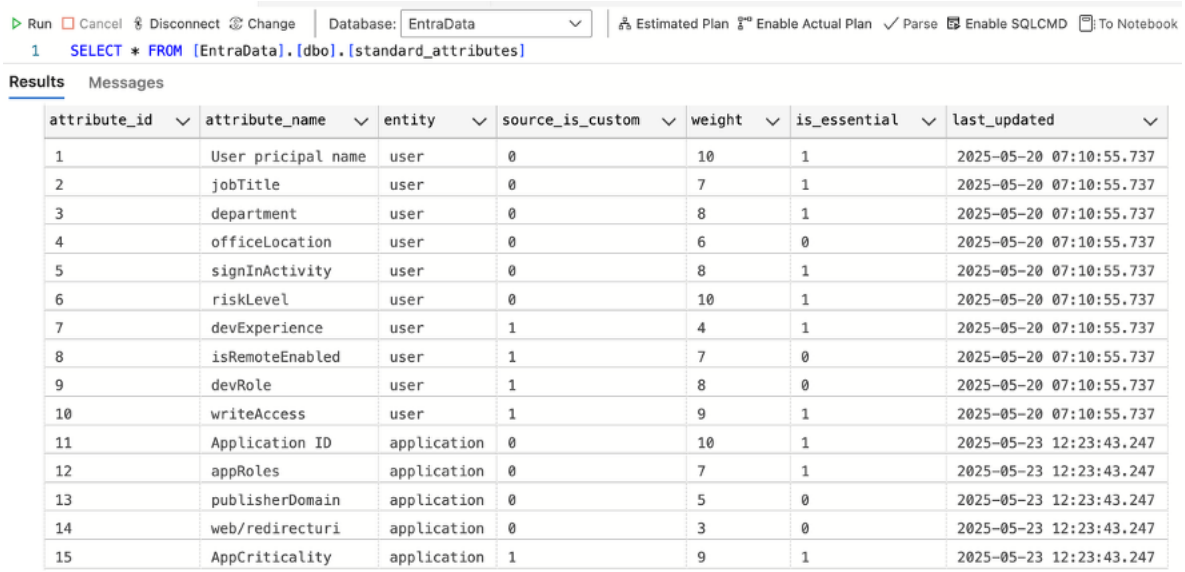
Figure 12 PIP components and connections

The attribute information is primarily sourced from Microsoft Entra ID. To enable access to this data, the PRP service is developed using the .NET Core (an open source and cross platform development framework) and configured with the necessary credentials, including the instance URL, Tenant ID, Client ID, Client Secret, and Graph API URL, credentials belonging to the DTW-ABAC integrated application (as detailed in section 3.1). A master attribute list is maintained in the SQL database to store all unique attribute names encountered during access evaluations. The table structure of user and application attributes, along with other parameters, is shown in Figure 13. This list is dynamically updated when new attributes appear in user or application profiles. In addition to the attribute name, the source classification (standard or custom) is maintained with a constraint that the custom attributes should be unique and different from standard attributes. Each attribute record also includes a ‘last_updated’ timestamp to

support future administration in terms of audit and compliance. To enhance access evaluation, two critical fields are appended to each entry:

1. attribute_weight: An integer between 1 and 10 indicating the security criticality of the attribute, where 10 represents the highest criticality and 1 represents the lowest.
2. is_essential: a boolean flag indicating whether the attribute must be present and match for granting access. The request is denied immediately without further evaluation if an essential attribute is missing or mismatched.

The exact values in these properties depend on the administrator and access scenario. Multiple experiments were performed to determine the critical attributes and weightage for each attribute used in the simulated environment.



attribute_id	attribute_name	entity	source_is_custom	weight	is_essential	last_updated
1	User pricipal name	user	0	10	1	2025-05-20 07:10:55.737
2	jobTitle	user	0	7	1	2025-05-20 07:10:55.737
3	department	user	0	8	1	2025-05-20 07:10:55.737
4	officeLocation	user	0	6	0	2025-05-20 07:10:55.737
5	signInActivity	user	0	8	1	2025-05-20 07:10:55.737
6	riskLevel	user	0	10	1	2025-05-20 07:10:55.737
7	devExperience	user	1	4	1	2025-05-20 07:10:55.737
8	isRemoteEnabled	user	1	7	0	2025-05-20 07:10:55.737
9	devRole	user	1	8	0	2025-05-20 07:10:55.737
10	writeAccess	user	1	9	1	2025-05-20 07:10:55.737
11	Application ID	application	0	10	1	2025-05-23 12:23:43.247
12	appRoles	application	0	7	1	2025-05-23 12:23:43.247
13	publisherDomain	application	0	5	0	2025-05-23 12:23:43.247
14	web/redirecturi	application	0	3	0	2025-05-23 12:23:43.247
15	AppCriticality	application	1	9	1	2025-05-23 12:23:43.247

Figure 13 Standard attributes with properties

The XACML access control policies are stored in the AuthzForce component hosted in a containerized environment. When an access request is triggered, the PIP retrieves the applicable policy domain and forwards the received attributes to the PDP for evaluation. The PIP does not

perform any evaluation logic itself but plays a critical role in orchestrating the data flow to and from the XACML engine. It ensures that:

1. The correct policy version is referenced using domain IDs.
2. Attribute values are correctly formatted and complete.
3. The request context complies with XACML schema expectations.

In addition to static attribute evaluation, the PIP supports dynamic context-aware logic through the NGAC layer. Figure 14 shows the NGAC graph topology, which is stored in SQL using two relational tables:

1. NGAC_Nodes: This table represents all atomic elements of the policy graph, including Entities (e.g., users e.g., Alice, applications, e.g., Health_App), Attributes - standard (e.g., singinLocation, webRedirectUri) and custom (e.g., devExperience, AppCriticality), Attribute values (e.g., Low, 3, EmergencyOverride) extracted from Entra ID or derived from logs, Permissions (e.g., Read patient data) and Policy logic nodes, such as compliance checks and override triggers

2. NGAC_Edges: This table captures directed relationships between nodes. While inspired by classic NGAC components, such as assignments and delegations, this implementation also extends to include real-time associations inferred from the underlying relational data. These represent dynamic links between users and attributes, attribute-based permission conditions, and behavioural constraints from user access logs.

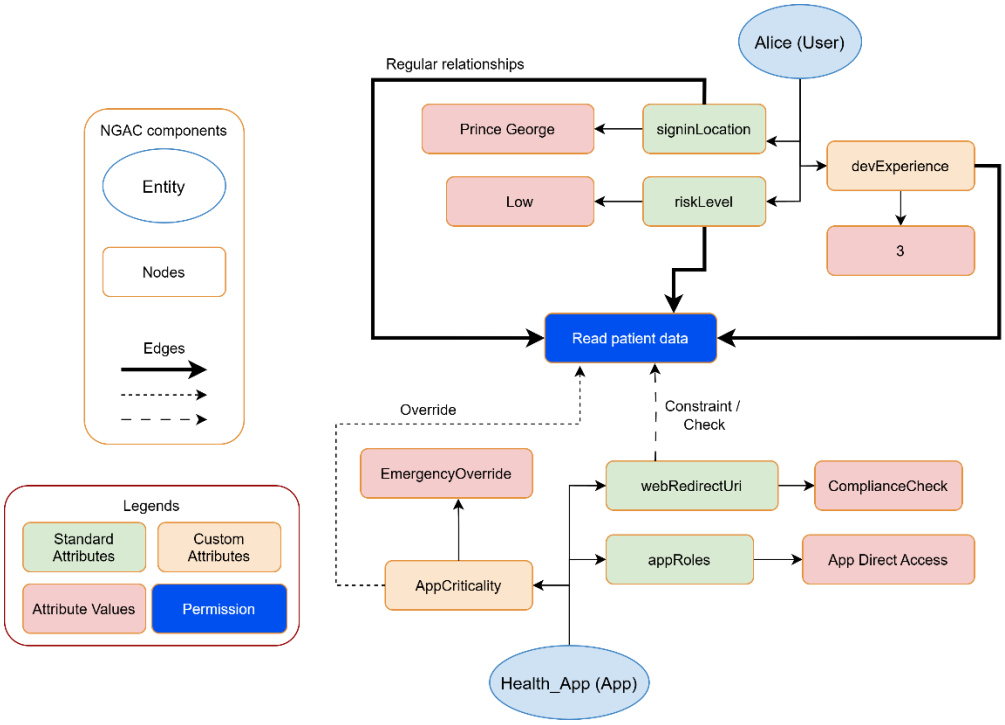


Figure 14 NGAC graph topology diagram

Upon receiving an access request, the PIP prepares a scoped subset of this graph, filtered by user ID, application ID, and permission type, ensuring that only relevant and recent attributes and edges are sent to the PDP. It also retrieves supporting metadata, such as weight values, deny thresholds, and historical access logs from the last 30 days. Although PIP does not perform any traversal or access path resolution, it ensures that PDP has all the necessary input to conduct a fully contextual and behaviour-informed NGAC evaluation.

3.2.3 Policy Decision Point (PDP) and Trust Factor (TF) formula Derivation

The Policy Decision Point (PDP) is the central intelligence unit of the DTW-ABAC architecture. It processes the structured access request forwarded by the Policy Enforcement Point (PEP). During this process, it requests the applicable policies and attributes through the Policy Information Point (PIP), and evaluates them using both static (XACML) and dynamic (NGAC)

engines coordinated by the Context Handler (CH). The PDP produces a final trust-weighted access decision, along with traceable metrics and logs for future evaluation.

a) Context Handler (CH) and Task Division

Context Handler is referenced from the traditional XACML framework; however, it is modified to act as the orchestrator between XACML and NGAC engines within the PDP. Based on the incoming request parameters such as user ID, application ID and permissions set, it identifies the application XACML policy sets and coordinates with PIP to extract attributes relevant to the decision (e.g., role, clearance, MFA device, Risk level). It delegates the request to the XACML engine with an appropriate list of domain ID and policy set ID (as described in the section 3.2.1, with the required attribute name-value pairs, and NGAC engines with previous access history simultaneously. Once the evaluation is completed, CH combines the evaluation results using a normalized trust factor. Additionally, override logic is applied if a delegation or critical threshold is received by either model.

b) Enhanced XACML evaluation

In the traditional XACML system, the evaluation result is typically a binary result, either "Permit" or "Deny", based on whether the access request fulfills the rules within the policy. However, the enhanced version supports a quantitative trust factor (TF) evaluation and attribute-weighted scoring, which enables a much more expressive and nuanced decision-making process. This modified engine still follows the XACML semantics, i.e. target, rule, effect and combining algorithm, but extends the output model to consider the matched attributes count with weight, and decision explanation for transparency. Upon receiving the request from the CH, the XACML

engine starts evaluating one or more relevant policy sets received from PIP. Each policy set may contain one or more relevant policies, and each policy contains a set of rules that specify match conditions for subject, object, action, and environment attributes. The evaluation flow is detailed in Figure 15.

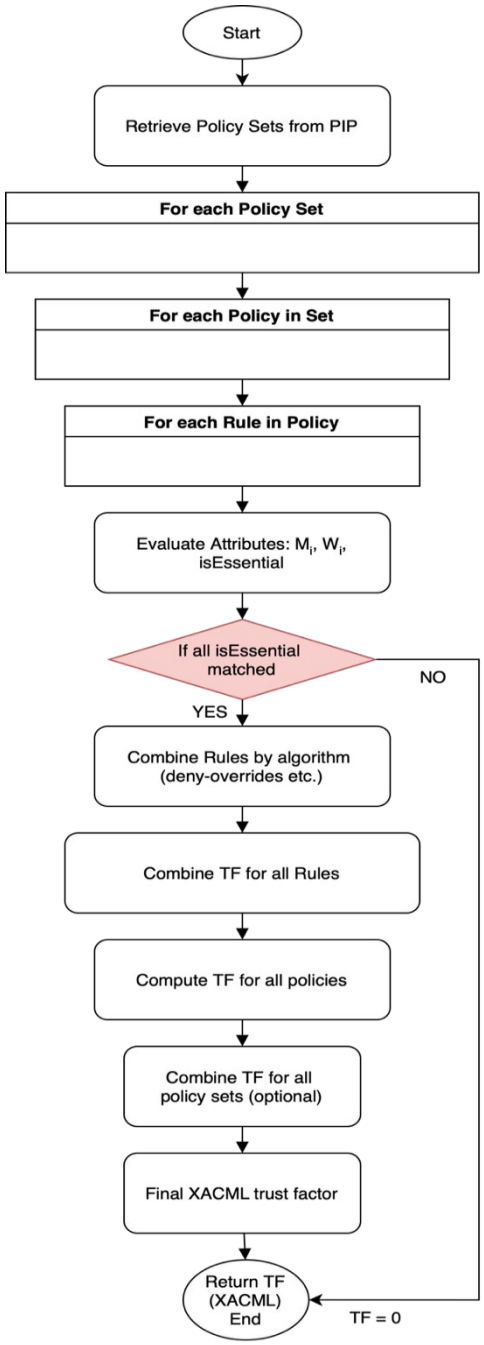


Figure 15 The XACML evaluation flow

The evaluation begins at the lowest level, i.e. per rule inside a policy. For each rule that applies (i.e., matches the target), the engine iterates through its attribute conditions. Each attribute (high-level only) is associated with:

- A match indicator $M_i \in \{0,1\}$ where 1 indicates that the attribute in the request matched the condition in the rule, and 0 otherwise.
- An attribute weight W_i, assigned by the administrator in the master list, reflecting the criticality of that attribute in enforcing secure access control.
- A boolean flag isEssential, which, if set to true and unmatched, triggers an immediate denial of the rule, policy, and policy set.

c) XACML Trust Factor Calculation

The rule-level contribution, i.e. TF_{rule} is computed in Equation (1), where i represents each attribute from 1 to N.

$$TF_{rule} = \frac{\sum_{i=1}^{N} W_i \times M_i}{\sum_{i=1}^{N} W_i} \quad (1)$$

If any essential attribute fails to match (i.e., $M_i = 0$ for any isEssential = true), the rule is invalidated (TF score is set to 0) regardless of its previously calculated TF score. If no essential failure is detected, the TF score of the rule is recorded. Rules within a policy are then combined using the configured rule combining algorithm, such as deny-overrides, first-applicable, or weighted-match. Once all applicable rules in a policy are evaluated, their combined score contributes to the policy-level trust as computed in Equation (2) where j represents each rule from 1 to R and an optional parameter, i.e. $RuleWeight_j$ represented as a weight associated with

each rule if the administrator wants to prioritize the rules (e.g., "department must match" > "location proximity"). By default, the $RuleWeight$ is set to 1 in case no rule prioritization is needed for the specific scenario.

$$TF_{policy} = \sum_{j=1}^{R} TF_{rule_j} \times RuleWeight_j \quad (2)$$

Further, across multiple policies within a policy set, a similar aggregation is performed in Equation (3). Each policy contributes a trust factor to the policy set's final score, where k represents a specific policy ranging from 1 to the total policies P. However, this calculation is ignored in case the applicable policy count is 1.

$$TF_{policySet} = \frac{1}{P} \times \sum_{k=1}^{P} TF_{policy_k} \quad (3)$$

Combining algorithms for policies also applies here, e.g., deny-overrides would force a zero score if any sub-policy issues a deny or fails an essential match constraint. The process is repeated for all policy sets returned by the PIP for the given request context. These sets can be domain-specific (e.g., one for HR, one for Finance) or application-specific. The final XACML trust factor is then computed by aggregating the TFs of each evaluated policy set, as mentioned in Equation (4), where l represents each policy set, ranging from 1 to the total number of policy sets S applied to the current context.

$$TF_{XACML} = \frac{1}{S} \times \sum_{l=1}^{S} TF_{policySet_l} \quad (4)$$

d) Enhanced NGAC evaluation

While the XACML engine evaluates the access request based on static policy rules, the NGAC logic within PDP simultaneously evaluates access using contextual attributes and relationship mappings provided by PIP. As shown in Figure 16, it includes relevant low-level attributes and graph data, such as session state, recent activity history, and attribute-to-permission mappings.

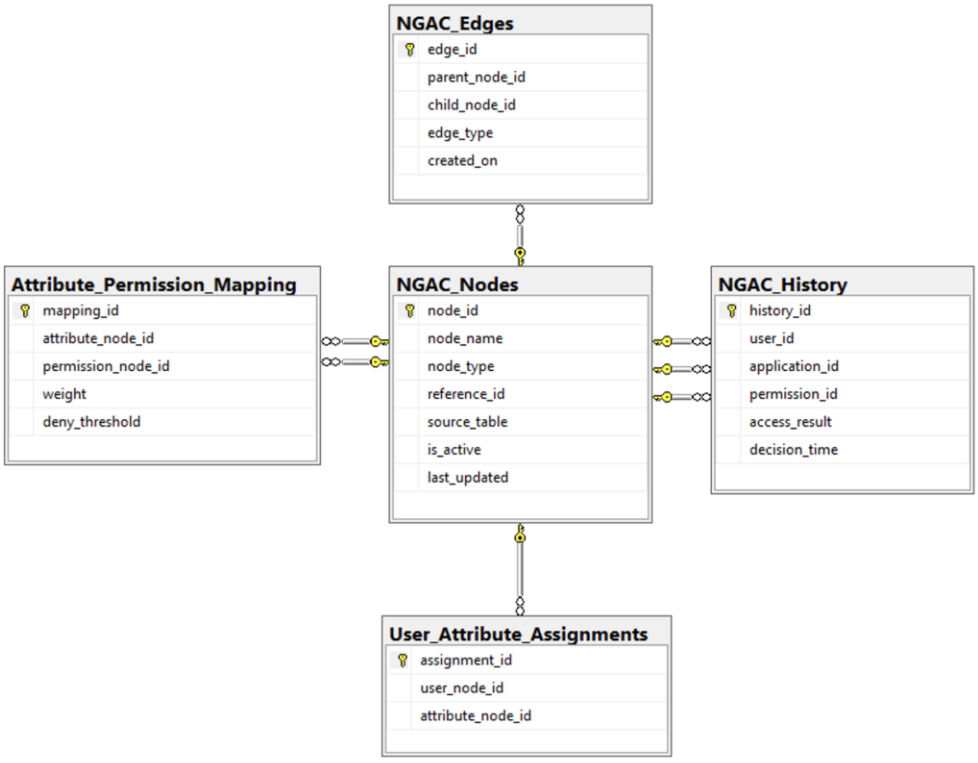


Figure 16 NGAC graph structure

This modified NGAC evaluation does not consider high-level declarative attributes or roles specified earlier in Table 8. Instead, it operates on concrete, low-level attributes such as login frequency (rate of successful login per specific day, like a month), device compliance, risk level, and MFA usage. These attributes are modelled in a SQL-based graph structure, where users, attributes, and permissions are represented as nodes, and their associations as directed edges. Unlike traditional NGAC, which emphasizes static graph traversal, this enhanced model

integrates real-time behavioural signals and trust-based scoring, allowing access decisions to reflect both current conditions and recent user behaviour. For each attribute relevant to the permission being requested, compute:

1. Match indicator $M_i \in \{0,1\}$, where 1 indicates that the user satisfies the attribute condition (e.g., MFA enabled, login frequency threshold met).
2. Assigned weight W_i, indicating the criticality of this attribute in dynamic risk evaluation.

e) NGAC Trust Factor Calculation

The base NGAC trust factor is then computed in Equation (5). This is conceptually identical to XACML's weighted match score, except that it operates over contextual and behavioural attributes, not declarative identity claims.

$$TF_{NGAC_{base}} = \left(\frac{\sum_{i=1}^{N} W_i \times M_i}{\sum_{i=1}^{N} W_i} \right) \quad (5)$$

Further, each user–application–permission tuple is associated with a deny threshold (T_d), which defines the maximum number of allowed denials for the same action within a recent time window. This mechanism is designed to penalize repeated access failures for a specific operation on a given application, regardless of improvements in attribute matching. Let:

1. $D_{\Delta T_H}$: denote the number of access denials in the last ΔT_H (as known as Historical period) days for a specific user–application–permission combination. Further work on finding the specific value of ΔT_H is presented in Equation (8).
2. T_d : represents the threshold for allowed denials for that same combination.

Then, if the user has reached or exceeded the denial threshold within the ΔT_H period (assuming ΔT_H = 30) day window, access is immediately rejected, and the NGAC trust factor is overridden; that is,

$$D_{30} \geq T_d \Rightarrow TF_{NGAC} = 0$$

This logic ensures that even if the user's current contextual attributes are favourable, repeated recent access failures for the same application-permission pair result in conservative denial, reinforcing risk-sensitive behaviour enforcement. In such a situation, the current setup stores an entry in the alert table with the obligation logic, but in a production environment, an email alert to the administrator can be configured.

To capture a user's historical behaviour, the enhanced NGAC engine incorporates a historical confidence (H) as shown in Equation (6), which is used to adjust the base trust factor according to previous success/failure counts within the applicable historical window (ΔT_H). The key component in the formula is the deny factor, i.e. D_{factor}, and the reason it has been multiplied by the deny count is to introduce a penalty weight for denied requests based on the user's risk level. This ensures each denied request has more damage to trust for high-risk users. So, it encourages high-risk users to improve both, the quantity and quality of their behaviour. Currently, the last 30 days of interaction logs are considered to ensure that the evaluation remains sensitive to current user behaviour and to prevent outdated historical data from skewing trust results. These numbers can be recalculated using the Equation (8) and modified easily, but require careful observation with respect to user traffic.

$$\mathrm{H} = \frac{A - (D \times D_{factor})}{T} \tag{6}$$

where,

- A : Total number of permitted requests per user per application
- D : Deny request count application
- D_{factor} : Deny factor based on previous risk level. i.e., for Low=1.0, Medium=1.5, High=2.0
- T : Total requests received per user per application

For example, let's consider three scenarios as below.

1. User with **Medium** Risk ($D_{factor} = 1.5, A = 30, D = 10, T = 40$)

$$\text{H} = \frac{30 - (10 \times 1.5)}{40} = \mathbf{0.375}$$

2. User with **High** Risk ($D_{factor} = 2, A = 30, D = 10, T = 40$)

$$\text{H} = \frac{30 - (10 \times 2)}{40} = \mathbf{0.25}$$

3. User with **High** Risk with a better permit ratio ($D_{factor} = 2, A = 35, D = 10, T = 45$)

$$\text{H} = \frac{35 - (10 \times 2)}{45} = \mathbf{0.33}$$

Based on the above calculation, it can be stated that two users with the same historical permit ratio will have different H factors due to different risk levels. Medium-risk users will have more H factor than a high-risk user with the same parameters, as well as another high-risk user with slightly better numbers. After calculating H, the final NGAC trust factor is:

$$TF_{NGAC} = TF_{NGAC_{base}} \times \text{H} \quad (7)$$

As derived in the Equation (7), the model continuously re-evaluates access outcomes and incorporates them into future calculations, which results in improved decisions over time. Users with consistent successful behaviour (e.g., repeated successful logins with compliant devices and MFA) will gradually build higher confidence scores through the H factor. On the contrary, users who accumulate frequent denials will see their access sharply penalized, either through a lower H value or full override via the denial threshold.

As part of this research, a custom formula was developed to calculate the historical period (in days), i.e. ΔT_H, as shown in Equation (8). This formula considers multiple factors that may affect security requirements. It helps determine how many past days should be considered relevant when evaluating historical access behaviour or trust levels.

$$\Delta T_H = \min(T_{max}, \max(T_{min}, W_1 \times E + W_2 \times R + W_3 \times U + W_4 \times A + W_5 \times C + W_6 \times F)) \quad (8)$$

where,

- $W_1, \ldots W_6$ are tunable admin weights (default all = 10),
- T_{min} is a minimum history period (e.g., 7 days),
- T_{max} is a maximum allowed period (e.g., 180 days),
- Other factors, as described in Table 10.

Symbol	Meaning	Example Scaling (0–1)
E	**Environment weight**: Prod = 1, QA = 0.6, Test = 0.3	Deployment criticality
R	**Risk factor of industry + app visibility**: e.g., Banking_public = 0.6, Banking_private = 1, Retail = 0.6	NIST/ISO risk categorization [58]

U	**User base scale** (normalized): $\frac{log_{10}(N_{users})}{log_{10}(N_{\max\_users})}$ (max_users is an organizational upper bound.)	Assuming max_users = 100K, so for, 10 users → U = 0.2, for 100K → U = 1
A	**Application base scale** (normalized): $\frac{log_{10}(N_{applications})}{log_{10}(N_{\max\_apps})}$ (max_apps is an application upper bound.)	More apps = more variance
C	**Criticality score of the resource** (0–1)	Manually defined or inferred from policy
F	**Audit frequency factor**: 1 / audit cycle in months	E.g., monthly audit → F = 1, annual → F = 1/12

Table 10 Historical period factors

For example,

- Environment = Production → $E = 1$
- Industry = Healthcare → $R = 0.9$
- 10000 users → $U = \frac{log_{10}(10000)}{5} = \frac{4}{5} = 0.8$
- 50 applications → $A = \frac{log_{10}(50)}{5} \approx 0.34$
- Resource criticality → $C = 0.8$
- Audit every 3 months → $F = \frac{1}{3} \approx 0.33$
- Assuming all weights =10, $T_{min} = 15$, $T_{max} = 90$
- Hence, $\Delta T_H = 41.7 \approx 42$ days

f) Final Trust Factor and Risk Level Calculation Engine

Once both XACML and NGAC scores are computed, the CH combines them using a weighted average as computed in Equation (9). It uses the constants for each model, i.e. C_{XACML} and C_{NGAC}, where the latter is set higher than the former. This prioritizes dynamic context (NGAC)

with a higher model constant. Multiple experiments are conducted as detailed in Section 4.5 to find the optimal values for the model constants.

$$TF = \frac{C_{XACML} \times TF_{XACML} + C_{NGAC} \times TF_{NGAC}}{C_{XACML} + C_{NGAC}} \times 100 \quad (9)$$

The trust factor groups can be configured by the administrator based on the criticality of the application and its security requirements. In this framework, a trust factor threshold of 70% is considered the minimum for access approval. Anything below is denied. Hence, based on this factor, PDP assigns a risk level to a user as below, which is updated over time.

- Low risk: $TF \geq 85\%$
- Medium risk: $70\% \leq TF < 85\%$
- High risk: $TF < 70\%$

Finally, the decision is logged in SQL with all the details captured during the evaluation for future review, testing, and experiments. Table 11 shows the log information stored during the multiple results generated at different levels of the evaluation, including the trust scores, attribute weights, failed attributes or evaluation details.

Column Name	**Type**
result_id	INT IDENTITY (1 1)
scenario_id	INT
resourceID	INT
userID	INT
model_type	NVARCHAR (20)
result_date	DATETIME
xacml_threshold	DECIMAL (10 2)
xacml_constant	DECIMAL (10 2)
ngac_constant	DECIMAL (10 2)
xacml_result	BIT

failedAttributes	NVARCHAR (MAX)
policyScore	FLOAT (53)
failedPolicyID	INT
policySetScore	FLOAT (53)
failedPolicySetID	INT
subjectWeightedScore	INT
subjectTotalWeight	INT
objectWeightedScore	INT
objectTotalWeight	INT
xacmlTrustFactor	FLOAT (53)
unmatchedEssentialCount	INT
ngacTrustFactor	FLOAT (53)
denyCount	INT
denyThreshold	INT
permitCount	INT
accessCount	INT
final_trust_factor	DECIMAL (5 2)
final_result	BIT
assignedPermissionName	NVARCHAR (25)
test_run_id	INT
risk_level	NVARCHAR (50)
is_active	BIT
test_run_id	INT
risk_level	NVARCHAR (50)
is_active	BIT

Table 11 Evaluation results log table structure

The trust and confidence scores in the hybrid model were designed using new equations directly linked to user history, instead of standard modelling techniques. One of the research questions was how access decisions can be made transparent and traceable, and these equations answered that need. Trust-based access control literature (section 2.4) shows the value of using observable evidence such as compliant actions, violations, and contextual risks. Unlike complex models, the equations make the calculation process clear and repeatable, so administrators can see and verify

how decisions are reached. By weighting positive and negative behaviours, the model adjusts trust dynamically while staying lightweight and easy to audit.

The trade-off is that standard modelling techniques may capture more detailed patterns, but they are harder to explain and heavier to run. The approach was focused on novel formulas that can be directly integrated into the existing XACML and NGAC design. Rather than introducing a separate engine, the intention was to show how trust scoring can be naturally embedded and aligned with the established XACML and NGAC framework. Whether this should be the new norm depends on the use case, i.e. for systems that need clarity, audit, and smooth integration, equations are a strong option, while in other cases, advanced models may still be useful.

g) Policy Enforcement Point (PEP) and history reset switch

The PEP is responsible for initiating the request and also enforcing the decision. Once the PDP evaluates the request based on defined policies and returns a decision, the PEP interprets the HTTP response, containing the decision status (Permit/Deny), applicable permissions, and any redirect or notification URL, and enforces the appropriate action. This may include granting access, denying access with a reason, or redirecting the user to an application-specific page (e.g., login, error, or application dashboard). The PEP also logs the interaction for auditing and compliance. Figure 17 shows the final response for a scenario where the access was granted, and the time counter shows that it will be redirected to the application dashboard. Other information, like trust scores or risk levels, is hidden from the end user. This message with redirect timeout is only for the development mode; as soon as the framework is hosted in a production environment, it will directly redirect the user to the application URL when permitted or display the Access

denied message. The development and production modes are managed by the dotnet 8.0 framework properties, which automatically switch the application to production mode after build and deploy.

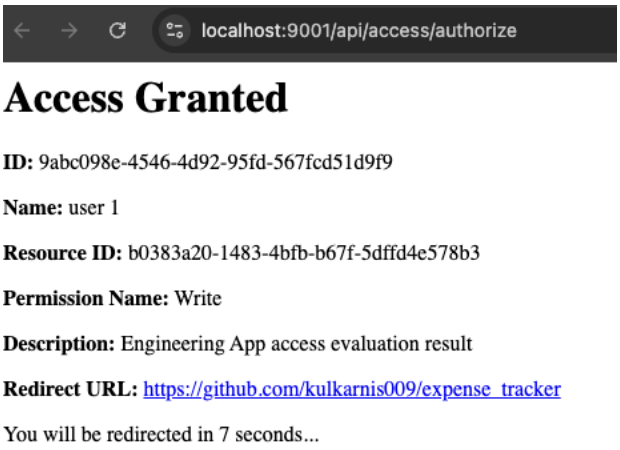


Figure 17 DTW-ABAC final decision with redirect URL

Users with high risk may experience repeated denials until they contact an administrator to reset their access. This process may involve additional user actions such as resetting the password, enabling multi-factor authentication (MFA), or applying device restrictions, depending on the organization's policy and the administrator's decision. To support this, the administrator can use the history reset switch to disable the user's prior access history by toggling the *is_active* boolean flag (associated with each log entry) for a specific user. The framework is designed to consider only active history entries, so this feature enables the user to start with a clean history post-reset. Hence, even if the disabled history falls under the historical period (ΔT_H) defined in Equation (8), the model will not consider it. Importantly, it avoids permanent deletion of historical records, thereby preserving audit integrity.

h) Production deployment option to manage the framework

Since the model components follow a RESTful design and use dotnet 8.0 framework and SQL Server, it can be easily deployed as an API (Application Programming Interface) and managed database in the cloud environment[59] [57]. The DTW-ABAC components, which do not need internet access (e.g. PDP and PAP), can be deployed in the private subnet, and others, such as PEP and PIP, can be deployed in the public subnet of the virtual private cloud (VPC) [60]. A custom JWT token and a VPC gateway service will transfer the data flow to the public subnet securely. The integration with an authenticator such as Entra ID or any other tool can be done with the PEP and PIP. The framework design enables the use of PAAS (platform as a service) deployment model services, such as AWS Lambda and RDS (Relational Database Service), to reduce maintenance and manage scalability.

3.3. Example Access Request

This section provides a concrete example which illustrates how the hybrid model (XACML + NGAC) evaluates access through weighted attribute policies and context-aware NGAC logic, including the full calculation flow that results in a permit decision (TF ≥ 70%).

Assume a software engineer (Alice) attempts to access the source code repository from a managed company laptop during outside working hours. Her recent behaviour and login history indicate that the user is low risk. The model will now evaluate her access using both XACML (static policies) and NGAC (contextual trust factors).

- The XACML evaluation: It consists of one policy set, two policies and three rules in total.

  Policy 1: Enforces a combination of identity and context-aware access control, ensuring that access is granted only when both user attributes and environmental conditions align with policy requirements.

Policy 1 => Rule 1 (p1r1): Enforces access based on the user profile details (static), ensuring only properly assigned users can proceed.

Attribute & Value	**Weight (W_i)**	**Essential**	**Match (M_i)**
role=Engineer	5	1	1
department=Design	3	1	1
projectCode=ABC123	2	0	1
shift=day	1	0	0

$$TF_{p1r1} = \frac{(5 \times 1 + 3 \times 1 + 2 \times 1 + 1 \times 0)}{(5 + 3 + 2 + 1)} = 0.909$$

Policy 1 => Rule 2 (p1r2): Enforces access based on physical location and device compliance, ensuring that requests originate from secure and approved environments.

Attribute & Value	**Weight (W_i)**	**Essential**	**Match (M_i)**
device=Compliant	4	1	1
Location=Vanderhoof	3	1	1

Region=North	2	0	0

$$TF_{p1r2} = \frac{(4 \times 1 + 3 \times 1 + 2 \times 0)}{(4 + 3 + 2)} = 0.778$$

Hence, Policy1 trust factor:

- Rule 1 weight = 0.4
- Rule 2 weight = 0.6

$$TF_{p1} = (0.909 \times 0.4) + (0.778 \times 0.6) = 0.8304$$

Policy 2 => Rule 1 (p2r1): Enforces resource-level access control, focusing on data sensitivity and access intent, specifically to ensure that only authorized users can access highly classified or confidential documents, and only under permitted operations (e.g., read-only). This prevents unauthorized disclosure or misuse of sensitive information based on the document's classification level, access type, and purpose.

Attribute & Value	**Weight (W_i)**	**Essential**	**Match (M_i)**
docType=Confidential	4	1	1
accessLevel=read	3	1	1
Classification=high	3	1	1

$$TF_{p2r1} = \frac{(4 \times 1 + 3 \times 1 + 3 \times 1)}{(4 + 3 + 3)} = 1.0$$

Hence, Policy 2 trust factor:

$$TF_{p2} = 1.0$$

The policy set (combining policy 1 and policy 2) trust factor:

$$TF_{policySet1} = \frac{TF_{p_1} + TF_{p_2}}{2} = \frac{0.8304 + 1.0}{2} = 0.915$$

Final XACML Trust Factor:

$$TF_{XACML} = TF_{policySet1} = 0.915$$

- The NGAC evaluation: Primarily enforces dynamic and behavioural attributes, including user clearance, current risk profile, and historical trust factors (confidence). The base trust factor comes from frequently changing attributes, while the final trust factor incorporates historical access records (permits/denies) and a risk factor.

  Attribute and match case:

Attribute & Value	**Weight (W_i)**	**Essential**	**Match (M_i)**
role=Engineer	4	1	1
team=SecureAccess	4	1	1
clearance=High	5	1	1
riskProfile=low	2	1	1
projectAccess=true	3	1	1

$$TF_{NGAC_{base}} = \frac{(4 \times 1 + 4 \times 1 + 5 \times 1 + 2 \times 1 + 3 \times 1)}{(4 + 4 + 5 + 2 + 3)} = \frac{18}{18} = 1.0$$

Denial and threshold check:

$D_{30} = 1$ (Only 1 denial in past)

$T_d = 5$ (deny threshold)

$$Since\ D_{30} < T_d\ \Rightarrow\ continue\ with\ TF_{NGAC}\ calculation$$

Historical period: $\Delta T_H = 42$ as explained in the Table 10.

Confidence Factor (H):

$$\mathrm{H} = \frac{A - (D \times D_{factor})}{T}$$

Let,

- The previous acceptance count (A) = 5
- The previous deny count (D) = 1
- Deny factor for low risk (D_{factor}) = 1
- The previous total request count (T) = 6

So,

$$\mathrm{H} = \frac{5 - (1 \times 1)}{6} = 0.66$$

Hence, final NGAC Trust Factor:

$$TF_{NGAC} = TF_{NGAC_{base}} \times H = 1 \times 0.66 = 0.66$$

- Final Trust Factor Calculation:

Let, $C_{XACML} = 0.65, C_{NGAC} = 0.85$

$$TF = \frac{C_{XACML} \times TF_{XACML} + C_{NGAC} \times TF_{NGAC}}{C_{XACML} + C_{NGAC}} \times 100$$

$$TF = \frac{0.65 \times 0.915 + 0.85 \times 0.66}{0.65 + 0.85} \times 100$$

$$\mathrm{TF} = 77.058\ \%$$

- Final Decision: PERMIT

- $TF = 77.058\ \%$
- Meets permit threshold ($TF \geq 70\ \%$)
- Risk level updated: Medium (previously it was low) because "$70\ \leq TF > 85\ \%$".

Based on the above example, it can be clearly seen how the framework has permitted the access request since all the mandatory and high-weighted attributes matched with a good historical record, but it has updated the user risk level to medium since a few optional attributes did not match, and there was a previous denial on record. Further, the updated risk level will affect the result next time if the optional attributes continue to fail.

3.4. Challenges

The development and evaluation of the DTW-ABAC framework posed several challenges, including policy modelling, real-time evaluation, data integration, and system scalability. The biggest challenge was to come up with an accurate attribute weight, which is inherently subjective. Since administrator-defined weights reflect organizational policies, they can introduce potential bias and lack empirical evidence. Similarly, deciding which attributes are marked as essential involves policy-specific knowledge and may lead to overly rigid denials if not calibrated carefully. Several experiments were conducted (see Section 4.5) to determine the essential attributes in this setup.

Although the enhanced XACML engine reduced the burden of frequent updates due to high-level attributes, it still faced the computational overhead when multiple policies or policy sets were involved. Additionally, hybrid scoring and penalty rules (e.g., 10% penalty for missed attributes) made debugging and policy validation harder. In order to solve these issues, additional logic for

simulation and visualization was developed as described earlier. With respect to NGAC, the H Factor changes the result significantly but also introduces challenges. The model considers the last 30 days as a history, which may not be enough to stabilize trust for low-traffic users. However, this challenge was resolved by creating a new equation to find the optimal historical period (as derived in Equation (8)).

Similarly, for the new users, the confidence may become low in a scenario where a few attributes are missing due to a lack of history or a previous approval-to-deny ratio. To tackle this issue, the framework provides flexibility to balance the model constants. The complete decision pipeline highly depends on timely and accurate data from Entra ID, which is an external entity. An incorrect attribute naming or incomplete values cause failed matches. The permission errors before purchasing the Entra ID P2 license affected the results initially. However, following the defined structure when adding simulated data helped resolve these issues. Since the framework can be connected with any authentication and identity management tool with minor changes, it reduces the dependency on Entra as well.

3.5. Summary

In summary, the methodology outlined in this chapter is designed to be both principled and extensible. It draws from existing formal models like XACML, NGAC and AR-ABAC but enhances them with numeric trust evaluation, risk-sensitive scoring, and dynamic policy enforcement. It also incorporates practical tools and data structures, such as SQL-based NGAC graphs and scenario simulation suites which ensure the model can be realistically deployed in real-world enterprise environments. This hybrid methodology also ensures that access decisions are correct according to rules and appropriate according to risk, trust, and historical behaviour.

Chapter 4

Experiments and Results

This chapter presents the experimental setup designed to validate the proposed hybrid access control model, as well as to compare results with standalone XACML and NGAC systems. The aim is to assess the model's correctness (e.g., accuracy, precision, recall, F1-score), adaptability to real-world access dynamics, robustness against attacks, and performance scalability. Additionally, this chapter outlines the design evolution from the initial model through iterations based on observed outcomes, contributing to the validation of the research objectives. The experiments are not only intended to test core access decisions (Permit/Deny) but also to measure trust factor behaviour, consistency of decisions, and responsiveness under varying loads and adversarial attempts. The hybrid model's internal components, such as attribute weight consideration, essential attribute enforcement, and dynamic NGAC context updates, are tested and fine-tuned under different input patterns and validated with the scenario results.

It is crucial to define how access scenarios were generated and how they were categorized into meaningful testing groups. These preliminary steps ensure that the experiments reflect realistic, security-relevant conditions rather than synthetic combinations that would not arise in practice. Although the categorization does not change the results patterns, it adds confidence to the model results.

4.1. Real-World Motivated Scenario Foundation

Recognizing the real-world operational needs that inspired the design of this hybrid model is essential before proceeding to the automated generation and evaluation of access scenarios. Each

scenario represents a single access request tested under any model. Table 12 presents ten manually crafted access scenarios drawn from diverse security-sensitive domains such as engineering, healthcare, financial systems, and international collaboration. These examples highlight both static policy reasoning, typically captured by XACML (e.g., role, time, location, and rule-based evaluation), as well as dynamic relationship or history-based decisions handled more effectively by NGAC (e.g., past access history, graph-structured role linkage, contextual emergency triggers). Each scenario is categorized by its access type and then mapped to contributions made by the hybrid framework, demonstrating how the hybrid architecture resolves complex conditions not fully manageable by either model alone. These scenarios serve two purposes, i.e. ground the system design in realistic operational settings and serve as benchmarks when validating programmatically generated scenarios that follow.

#	Scenario Summary	Access Type	DTW-ABAC Hybrid framework	
			XACML Role	NGAC Role
1	Engineer with low risk requesting app (limit based on denial history)	Risk-aware Logs	Handles user/app risk level attributes	Tracks denial history
2	Project file access by role and location	Hierarchical + Context	Policy for location and sensitivity	Graph links role-location
3	Critical access override for managers	Emergency Overrides	Rule exceptions	Propagates changes graph-wise
4	Access is allowed based on request patterns, time, and device compliance	Dynamic Time-Based	Enforces usage & compliance	Tracks historical access
5	Emergency action by trained staff only	Emergency Action Rules	Training-based access control	Links users to training nodes
6	Access denied if risk score exceeds threshold	Risk Behaviour Decision	Risk evaluation rule	Looks up risk history

7	Edit denials for documents authored by other team members	Fine-Grained with Relationships	Team-only modification policy	Manages team-author relationships
8	Dept A is requesting DNA data from Dept B for collaboration	Cross-Department Dynamic Access	Defines inter-departmental conditions	Reflects org structure + updates seamlessly
9	Emergency patient access by doctor (e.g., during cardiac arrest)	Emergency for Healthcare	Ensures revocation post-emergency	Applies emergency overrides via graph
10	Access to sensitive project files while travelling	Context-Aware in International Operations	Enforces relaxed travel-aware policy	Evaluates the project and location graph nodes (new countries are added easily)

Table 12 Access scenario summary table

4.2. Programmatic Access Scenario Generation

The manually crafted scenarios in the previous section represent critical access control challenges that inspired the system design and informed the attribute sets used in subsequent automated testing. However, in order to conduct rigorous testing, it is necessary to generate hundreds of diverse access scenarios programmatically. In real-world environments, access requests are not random but are influenced by organizational policies, contextual conditions, and user roles. Therefore, to reflect this complexity, the experiments began with the creation of a comprehensive access scenario dataset.

4.2.1 Cross-Product-Based Initial Dataset

Based on different access types, an initial pool of scenarios was generated by computing the cross-product of Users × Applications × Permissions. Each scenario tuple consisted of user ID, application ID and permission name. The different users and applications are distinct from each other in terms of attribute names and their values. A total of 760 scenarios were generated using this technique. This exhaustive combination ensures that the model is exposed to both typical and edge-case inputs. Table 13 shows the sample test scenario generated using the cross-product method.

User ID	Risk Level	App ID	Permission	Location	Time	Device Type
User1	Low	AppA	Read	Office	9:00 AM	Laptop
User1	Low	AppA	Write	Home	9:00 AM	Mobile
User2	Medium	AppA	Read	Remote	2:00 PM	Desktop
User2	Medium	AppB	Write	Remote	2:00 PM	Desktop

Table 13 Test scenarios using cross-product

4.2.2 Filtering Techniques for Realism

Since not all combinations are realistic or policy-relevant, a structured programmatic generator is developed to filter realistic and security-sensitive test scenarios that mimic finite state machine (FSM) transitions [61], [62], [63]. However, instead of modelling a full FSM for the entire policy engine, the research uses logical scenario paths inspired by FSM-based model testing strategies. These paths are composed of semantic transitions capturing the behavioural flow of access control decisions in the following chain:

$$\textit{Attribute Assignment} \rightarrow \textit{Trust Evaluation} \rightarrow \textit{Permission Request}$$

Each unique sequence through this chain simulates a logical path in an abstract FSM. By varying the attribute values and structural context at each stage, a wide range of possible evaluation states is covered, similar to achieving state and transition coverage in classical FSM testing. The following strategies were implemented within this path-diversification framework to ensure scenario relevance and impact:

a. Policy-Driven Filtering - Only combinations where user, resource, or context attributes plausibly satisfy at least one XACML policy rule were retained. This eliminates invalid or irrelevant permutations from the cross-product results and ensures meaningful scenario paths.

b. FSM-Based Path Diversification - The test generator simulates logical transition-like sequences (assignment → evaluation → decision), enabling scenario coverage across the access decision landscape**,** without explicitly generating full FSM diagrams. Each scenario path is structurally different, improving APFD (Average Percentage Fault Detection) as observed in studies mentioned in [63] which compared XACMET (XACML Testing and Modelling Environment) and Multiple Combinatorial strategies for XACML-based access control testing using a controlled experiment and demonstrated that scenario diversity (through path variation) can positively impact early fault detection rates.

c. Mutation-Based Negative Case Injection - For every positive scenario, a “near miss” version is programmatically created by mutating one attribute, such as altering role, removing required attributes, or falsifying location. This mirrors FSM mutation operators like CO (Change Output), which induces denial where the original was permitted and

CTS (Change Tail State), which changes attribute-based path transitions. These mutations are used to simulate adversarial, misconfigured, or incomplete access attempts.

Using these techniques, a total of 280 scenarios were selected from the initial pool of 760 scenarios. The generated scenarios are not merely Cartesian outputs but strategically enriched cases reflecting both standard and edge-case access situations. They are used to test the decision stability, trust responsiveness, and error detection capability of the hybrid access control model.

4.3. Scenario Categorization

To enable structured evaluation and metric comparison (e.g., TP/TN/FP/FN breakdowns), the scenarios were grouped into six well-defined categories. This taxonomy allows focused analysis on specific behaviour classes and was inspired by both real-world policy deployments and prior evaluation studies. Each category is designed to assess a specific attack vector as well, such as direct ("front door") and indirect ("back door") attempts, as well as other sophisticated adversarial strategies. An example scenario, with attribute details and evaluation process per category, is added in the Appendix 1. Although even if the model executes all 280 access scenarios exactly once, scenarios across different categories share the overlapping user-app combinations. This overlap is intentional and beneficial, allowing the model to leverage comprehensive access histories. Thus, when evaluating any given scenario, the system considers contextual information from scenarios in other categories as well. This interconnected evaluation ensures robust testing coverage, accurately reflecting realistic attack patterns and enabling the detection of nuanced adversarial behaviours that span multiple attack vectors. The details about each category are as follows.

4.3.1. Baseline Valid Access

This category includes legitimate, correctly authorized access requests that should be granted (True Positives), and a small set of benign but unauthorized requests that should be denied (True Negatives). The large number of True Positives ensures the system is well-tested for normal, intended use cases that must be reliably permitted. Only 2 TNs are included here since most real traffic is expected to succeed, and other categories more fully test TN cases involving adversarial or hidden attacks. This balance reflects real-world expectations where most legitimate traffic should succeed, but basic denial handling is still validated.

4.3.2. Adversarial or Malicious Attempts

This category covers deliberately crafted or spoofed access requests designed to defeat security controls. Examples include falsified or forged attributes, hidden or manipulated context, or other forms of attack simulation. These scenarios stress-test the system's ability to detect, block, and handle hostile or deceptive attempts intended to bypass policies or exploit vulnerabilities.

4.3.3. Behavioural or Historical Influence

The scenarios in this category simulate changes in access patterns based on a user's historical behaviour. For instance, it tests how the system reacts when a user who usually accesses resources during business hours suddenly attempts access at odd times or from unusual locations. It reflects adaptive, history-aware control policies that consider past behaviour to influence decisions.

4.3.4. Contextual or Temporal Drift

This category includes access requests where contextual attributes (like time of day, location, device type, or session expiry) deviate from the norm. For example, an employee accessing from

an unfamiliar country at midnight or an uncompliant device. These tests check if the system can handle changing and potentially risky situations.

4.3.5. Policy Conflict and Ambiguity Handling

This group includes situations where multiple policies apply simultaneously but have conflicting effects (e.g., one policy permits access while another denies it). These situations challenge the system to handle unclear situations using conflict resolution strategies. This ensures that decisions are predictable and easy to explain, even when policy rules are in conflict.

4.3.6. Structural Attribute Violations

This category targets cases where static attributes required for access decisions are missing, incorrect, or malformed. Examples include missing role assignments, department mismatches, or other identity and attribute errors. It ensures the system can detect and deny requests when essential identity data is invalid or incomplete, maintaining policy integrity.

To avoid confusion during results analysis, category-to-scenario follows a one-to-many relationship, as shown in Table 14. It also shows the total TP and TN (expected decisions) counts that were determined manually to maintain an independent evaluation benchmark. In cases influenced by organizational context, such as conflicting rules, time-based conditions, or environmental attributes, reasonable assumptions based on common practices were applied to ensure fairness and reproducibility across all evaluations.

No.	Category	# Scenario count (out of 280)	TP (Grant Access)	TN (Deny Access)
1	**Baseline Valid Access**	40	38	2
2	**Adversarial or Malicious Attempts**	45	10	35
3	**Behavioural or Historical Influence**	50	40	10
4	**Contextual or Temporal Drift**	48	31	17
5	**Policy Conflict and Ambiguity Handling**	47	32	15
6	**Structural Attribute Violations**	50	21	29

Table 14 Access scenario categories

The motivation behind categorization was to add clarity in the evaluation of metrics for each category, revealing model strengths and weaknesses.

4.4. Core Functional and Comparative Tests

To evaluate the overall effectiveness of the proposed hybrid model, a comprehensive set of core tests was conducted. These are divided into two primary segments: Hybrid vs. Standalone Models (XACML and NGAC) Evaluation and Hybrid Model Parameter Impact Evaluation (Tuning). Each group includes targeted metrics and tuning techniques designed to assess decision quality, trust dynamics, and attribute influence. Together, they provide a structured framework to validate the model across realistic, adversarial, and dynamic access scenarios.

4.4.1. Hybrid vs Standalone Models Confusion Matrix Comparison

This component compares the hybrid model performance with the standalone XACML and NGAC implementations under identical test conditions. The objective is to determine the added value and behaviour of the hybrid logic, especially in situations involving contextual and

temporal variation. The result for any scenario is defined as either TP (True Positives, i.e. correctly permitted access for legitimate users), FP (False Positives, i.e. incorrectly permitted access for unauthorized users), TN (True Negatives, i.e. correctly denied access to unauthorized users), or FN (False Negatives, i.e. incorrectly denied access to legitimate users) [64]. All three models (DTW-ABAC, XACML and NGAC) were tested for the access scenarios in each category, and the performance was evaluated using the confusion matrix. In the Hybrid model, the internal constants were set as C_{XACML} = 0.65 & C_{NGAC} = 0.85 (the rationale for choosing these values is explained later in Section 4.5.1), as well as set the maximum denials depending upon the application present in the specific scenario. The Risk factors in the hybrid model remain as 1 (Low risk), 1.5 (Medium Risk) and 2 (High Risk) as explained in Section 3.2.3.

1. Baseline Valid Category

In the Baseline Valid Access category, 38 scenarios involved straightforward, legitimate access requests that should have been granted without exception, and 2 scenarios were a small set of benign but unauthorized requests that should be denied, as explained in the section 4.3.1.

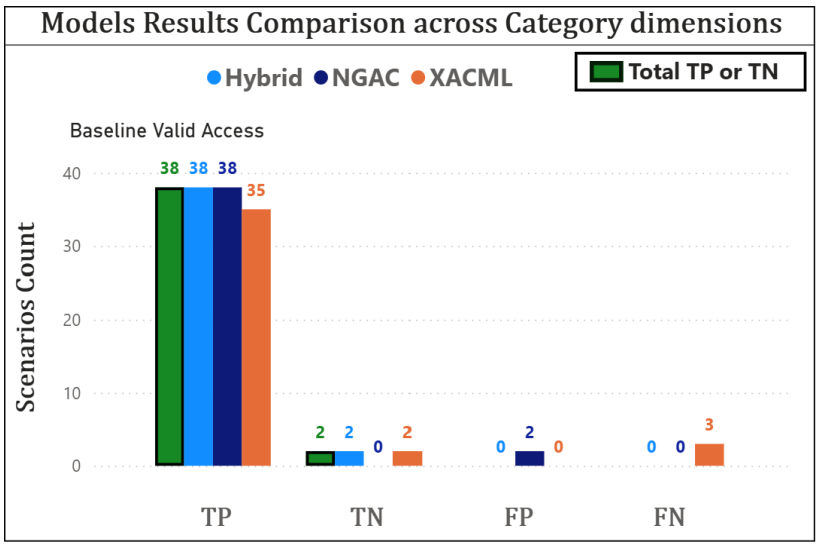


Figure 18 Baseline valid access category results

Analyzing the results presented in Figure 18 revealed that XACML produced false negatives due to its strict policy evaluation semantics. Specifically, three access requests (8%) were denied because the policies included complex target and condition structures that required exact attribute matches. In several cases, although the user had the correct role or permissions conceptually, minor mismatches or missing optional attributes, such as slight variations in role naming or the absence of time or environment attributes, caused the request to fall outside the applicable policy scope. These were not policy errors by themselves but reflected how XACML's default-deny behaviour, coupled with deny-overrides combining algorithms, failed to account for expected flexibility in baseline access. On the other hand, NGAC exhibited false positives in this category, which, upon analysis, were traced to its attribute-based graph traversal mechanism, allowing access via broader propagation paths than originally intended. Users accessed permissions through inherited or higher-level attribute associations without adequate contextual filters. This led to access, which bypassed the finer constraints expected by the scenario definitions despite being technically valid under the graph structure. The Hybrid model successfully avoided both types of errors by integrating XACML’s precision in attribute evaluation with NGAC’s flexible propagation, applying trust-layer filtering and essential attribute validation to ensure that valid requests were neither unjustly denied nor too broadly permitted.

2. Adversarial or Malicious Attempts Category

In the Adversarial or Malicious Attempts category, the goal was to simulate illegitimate access scenarios, including intentional misuse, privilege escalation, or requests that violated defined constraints. It also includes simulated access requests using VPN (virtual private network) to generate geolocation variance.

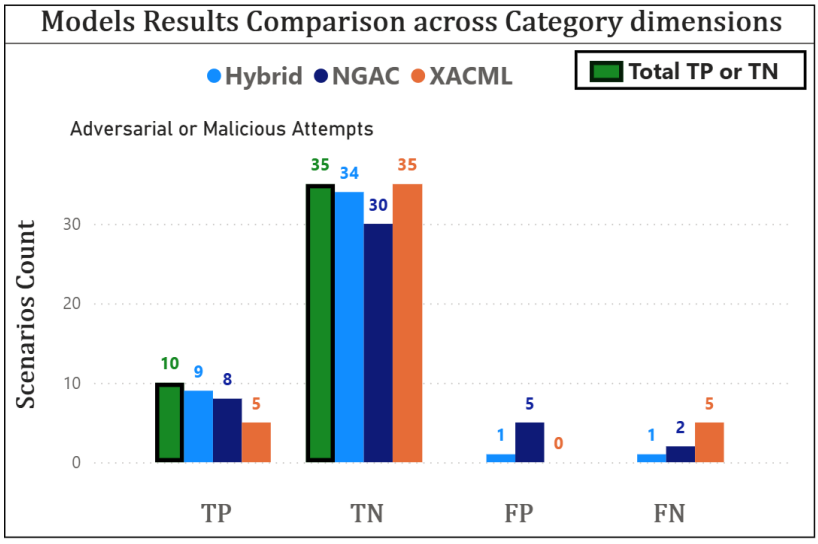


Figure 19 Adversarial or malicious attempts results

During the investigation, it became evident in Figure 19 that XACML produced zero false positives, reflecting its conservative nature and default-deny stance; its fine-grained, rule-based enforcement mechanism rigidly blocked all access attempts that did not match well-defined conditions. However, this rigidness came at a cost: XACML also recorded five false negatives (FN), meaning it incorrectly denied legitimate access requests. These FN cases occurred because the requests were structurally valid but lacked certain contextual elements (such as time, risk, or behavioural indicators) not fully accounted for in the policy logic. This limitation arose from its rule-based matching, which did not flexibly accommodate variations in contextual attributes. NGAC reported 5 FPs due to a lack of integrated evaluation for request intent, temporal conditions, and anomaly signals, resulting in access being granted through structurally valid but contextually inappropriate paths, particularly in scenarios involving lateral movement or role abuse. Additionally, it recorded two false negatives, where legitimate user requests were wrongly denied because the policy lacked sufficient granularity to handle contextual variations and resolve graph-based conflicts. The Hybrid model outperformed both by recording only one false positive and one false negative. It successfully blocked most adversarial attempts by layering

contextual trust assessments, such as access history deviation, session state, and anomaly indicators, on top of NGAC's structural enforcement. At the same time, it avoided XACML's rigidity by relaxing strict match conditions when broader patterns clearly indicated a malicious attempt, thus reducing false negatives. The single FP arose in a case where NGAC's propagation briefly superseded the hybrid filter due to attribute misclassification, while the only FN was tied to a borderline case where contextual deviation was insufficient to trigger the denial threshold. Overall, the Hybrid model's combined reasoning, precise rule validation, dynamic relationship tracking, and threshold-based filtering enabled it to distinguish between subtle adversarial intent and legitimate patterns in general, while recognizing that tuning these parameters could prioritize stricter security or user convenience depending on organizational risk tolerance.

3. Behavioural or Historical Influence Category

In the Behavioural or Historical Influence category, the scenarios were designed to capture deviations from typical user behaviour, usage history, or long-term access patterns. These include situations where a user's prior actions, assigned roles, or past access decisions, whether permitted or denied, should influence current access control decisions.

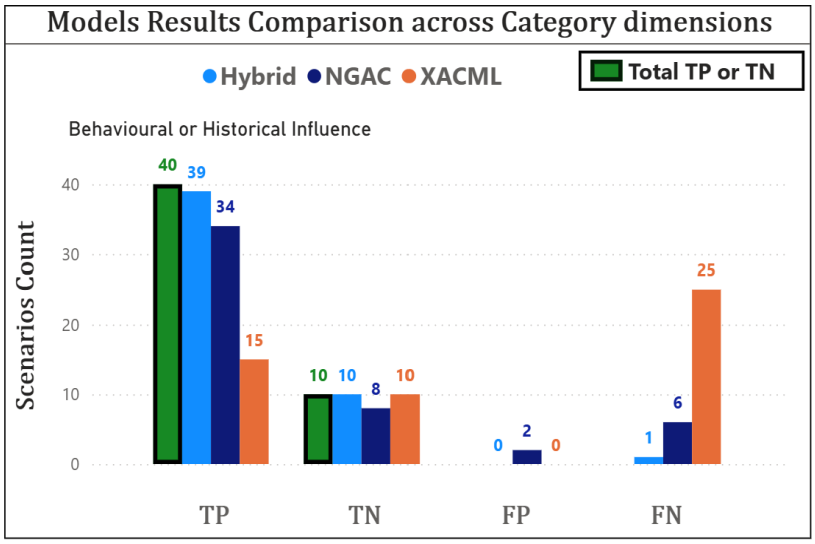


Figure 20 Behavioural or historical influence results

From the observed results in Figure 20, XACML recorded a notably high number of false negatives (25), far exceeding NGAC's 6 and Hybrid's 1. This indicates that XACML consistently failed to permit access in cases where deviations were legitimate but misclassified as suspicious. The primary reason lies in XACML's static policy design: it lacks native support for stateful or temporal reasoning, and its rules are generally unable to incorporate behavioural metrics such as frequency of access, anomaly scores, or past violations. Consequently, unless the policy explicitly encodes all historical edge cases, XACML remains blind to context shifts over time, leading to a sharp drop in responsiveness to the legitimate behavioural changes and a corresponding spike in FNs. NGAC fared moderately better, with 6 false negatives, indicating it blocked a few legitimate requests due to failing to recognize fewer common patterns in user attributes. It also showed 2 false positives, where unauthorized access was incorrectly permitted. These FPs occurred when NGAC failed to identify an unfamiliar context as risky, allowing access even though the user's attribute configuration had abruptly changed, highlighting its vulnerability to certain structural anomalies. The Hybrid model again achieved the best balance, recording just 1 FN and 0 FPs. It captured behavioural anomalies by integrating historical data

layers, such as access frequency thresholds, recent activity logs, or deviations from typical usage patterns, into its decision process, which neither XACML nor NGAC handled effectively on their own. By combining NGAC's graph-based dynamism with XACML's conditional constraints and incorporating adaptive weight factors, Hybrid was able to distinguish legitimate variation from true violations with high precision, significantly reducing both types of errors in this category.

4. Contextual or Temporal Drift Category

The Contextual or Temporal Drift category differs fundamentally from the Behavioural or Historical Influence category in that it focuses not on long-term user behaviour patterns, but on real-time environmental and situational variables that can change between sessions, such as time of day, device used, network location, risk levels, or session states. While the Behavioural category evaluates user consistency over time, the Contextual category evaluates situational appropriateness at the moment of access. Since these conditions are highly dynamic and volatile, policies must adapt in real-time to subtle shifts that may indicate elevated risk or misuse. While both the Contextual and Behavioural categories deal with variable factors, the Behavioural category often incurs more FPs and FNs due to its reliance on patterns that may be noisy, inconsistent, or anomalous. In contrast, the Contextual category, though complex because it looks at the user's identity, history, and environment, it usually results in fewer errors. This is because the models are typically more accurate at capturing and responding to contextual or temporal shifts than to unpredictable behavioural anomalies.

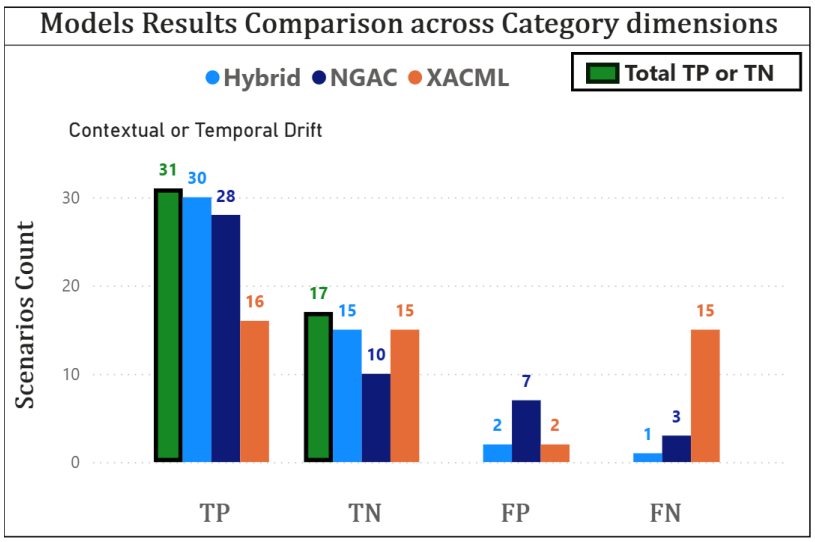


Figure 21 Contextual or temporal drift results

As presented in Figure 21, XACML shows a high number of false negatives (15), significantly more than NGAC (3) or Hybrid (1). This sharp underperformance is due to XACML's limited capacity to process dynamic runtime context. Although XACML supports environment attributes in its model, its policies are often written with static thresholds or assumptions, which become ineffective when dealing with unpredictable shifts in session variables or risk indicators. Without real-time feedback or policy adaptation, XACML fails to permit many access attempts that are contextually appropriate (such as those from trusted devices at unusual times or from atypical locations) simply because they do not match predefined static rules, resulting in a high number of false negatives. NGAC, with 3 false negatives and 7 false positives, performs slightly better in recognizing contextual violations but struggles with overgeneralization. Its attribute graph allows flexible policy application across varying contexts, but the absence of tight contextual binding or real-time evaluation (e.g., no native tracking of risk scores, time windows, or device fingerprints) causes it to either over-block legitimate access when an unfamiliar session context arises (leading to FN), or under-detect nuanced threats that fall within structurally permitted paths (leading to FP). The 7 false positives in NGAC suggest that access was incorrectly granted when

unfamiliar contextual combinations (such as unusual networks or atypical login times) went undetected, which failed to trigger cautious access decisions. The Hybrid model, with just 2 false positives and 1 false negative, effectively mitigates both types of errors by fusing the strengths of its components. It leverages NGAC's structural adaptability while embedding real-time contextual verification layers, such as current session parameters, time-based constraints, and device trust scores, into the access evaluation pipeline. It avoids XACML's rigid overdependence on static context and compensates for NGAC's lack of contextual specificity by injecting environment-aware checks and deviation thresholds into its policy decisions. As a result, Hybrid identifies inappropriate access under shifting contextual conditions while still recognizing legitimate variation, achieving a superior balance in this highly dynamic category.

5. Policy Conflict and Ambiguity Handling Category

The Policy Conflict and Ambiguity Handling category addresses scenarios where multiple policies apply simultaneously, often with overlapping conditions, contradictory decisions, or ambiguous outcomes. These conflicts arise when, for example, one policy permits access based on role, while another denies it based on department, or when policies differ in how they prioritize attributes such as clearance level, context, or resource sensitivity. Effective conflict resolution requires not just evaluating individual policy rules but also understanding how to reconcile them through combining algorithms, precedence rules, or structural hierarchies. Errors in this category typically indicate a failure to resolve such ambiguities correctly, leading to unjustified access grants (false positives) or unnecessary denials (false negatives).

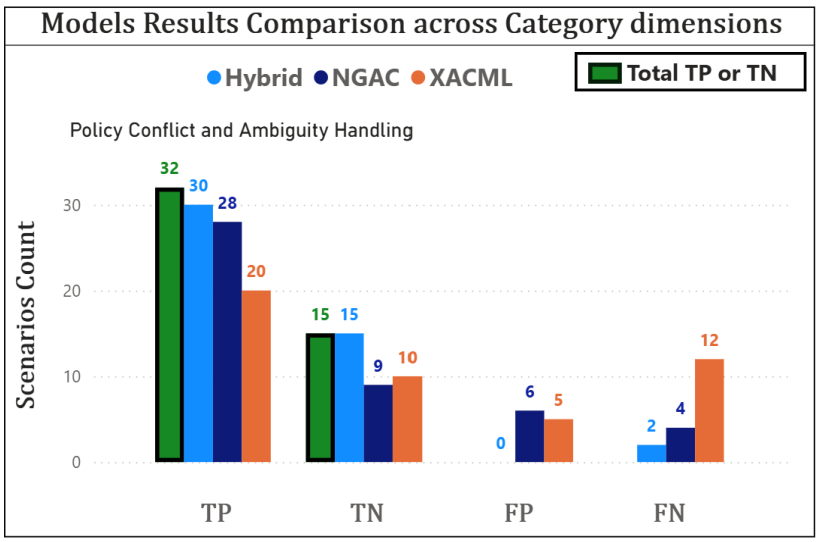


Figure 22 Policy conflict and ambiguity handling results

As per the results shown in Figure 22, NGAC recorded the highest number of false positives (6), followed by XACML with 5, while the Hybrid model avoided any false positives entirely. NGAC's graph-based model is powerful for flexible permission propagation but lacks fine-grained policy resolution mechanisms. When multiple attribute paths provide access, NGAC does not inherently resolve which path should dominate. This leads to situations where conflicting permissions are both accessible, and in the absence of explicit precedence, NGAC tends to default toward permission propagation, resulting in over-permissive outcomes. XACML, despite its strict enforcement model, also encountered false positives due to inconsistent application of combining algorithms, such as permit-overrides or first-applicable. In certain cases, it misprioritized a permissive rule over a more restrictive one, particularly when multiple overlapping policies evaluated to conflicting results and no clear precedence was defined or enforced, leading to access being granted where it should not have been. False negatives in this category further highlight the models' weaknesses in ambiguity handling. XACML again shows the most significant count, with 12 FNs, indicating that it often denied access unnecessarily due to unresolved conflicts or conservative evaluation. This happens when

policies are overly strict, and the combining algorithm defaults to deny in case of uncertainty or partial matches, especially when policy scopes overlap but do not fully agree. NGAC, with 4 false negatives, performed better but still failed to detect certain allowable access paths due to ambiguity in attribute inheritance and when intermediate nodes conflicted in role-based access versus contextual overrides. The Hybrid model, with only 2 false negatives and zero false positives, achieved a more accurate balance by introducing an explicit conflict resolution mechanism that synthesizes NGAC's structural flexibility with XACML's rule-based specificity. It not only evaluates the decision outputs of overlapping policies but also incorporates a resolution layer that checks for model constants (C_{XACML} & C_{NGAC}), attribute weight, prior risk level, and trust score alignment. The model constants assign final weighting to the model, helping to resolve conflicts and produce a clear, quantified result. This allowed Hybrid to permit access when legitimate but avoid over-granting, and to block access only when conflicts truly indicated risk, resolving ambiguity both structurally and semantically with higher precision than either model alone.

6. Structural Attribute Violations

The Structural Attribute Violations category refers to scenarios where access decisions depend on the correct configuration, assignment, and hierarchical relationships of user and resource attributes, such as roles, departments, clearance levels, or access tiers. These violations typically involve misuse, misassignment, or manipulation of attribute values to bypass access restrictions. For example, a user may be incorrectly assigned to a senior role, or a resource may be misclassified, leading to access that should not be permitted. Detecting such violations requires the access control model not only to validate attribute presence but also to assess attribute correctness, essentiality, and structural integrity within the policy graph or rule logic.

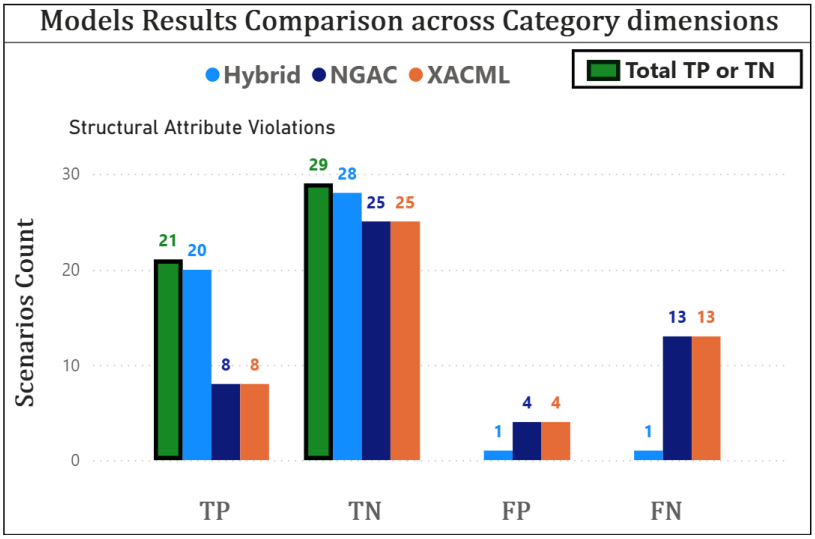


Figure 23 Structural attribute violations results

In this category, the results in Figure 23 show that both NGAC and XACML performed poorly, each recording 13 false negatives, highlighting a substantial failure to grant access to structurally valid requests. For NGAC, the issue stems from its flexible attribute propagation model, where users inherit permissions through graph-based relationships. Without mechanisms to explicitly enforce structural integrity (such as mandatory attribute presence checks, uniqueness constraints, or hierarchical role validation), NGAC misinterpret valid configurations as incomplete or inconsistent, leading to unjustified denials. For instance, a user correctly assigned to a role like "project_manager" is denied access if their supporting associations (e.g., "staff" or "department_member") are not clearly reinforced in the graph. XACML, in contrast, lacks structural awareness entirely. While it evaluates individual attribute values against static rules, it does not interpret or enforce relational dependencies among those attributes. As a result, it may deny access even when the attribute set is logically valid but doesn't match the exact static condition, for example, rejecting access where a user has "project_lead" but the policy expects an exact combination like "project_lead" plus "region_member." Without explicit modelling of attribute dependencies or structural rules, both models fail to recognize legitimate access

requests, resulting in a high number of false negatives. Both models also recorded 4 false positives each, meaning they incorrectly permitted access when attribute structures were in fact invalid. In XACML, this typically occurred when policies lacked structural depth, causing the model to approve access based on superficial attribute matches without verifying their intended relationships. For example, a user assigned the attribute “manager” has been granted access even if their supporting attributes (such as “employee” or “department_member”) were missing, simply because the rule matched the top-level label. NGAC’s false positives were often the result of permissive attribute propagation, where access was allowed through indirect or unintended paths in the graph. For instance, a user linked to a general “staff” node traverses to a privileged resource via intermediate roles like “project_lead” without satisfying structural requirements, such as being part of the specific project group. In both cases, the lack of strict enforcement over attribute hierarchy and dependency allowed structurally flawed access to be incorrectly granted. The Hybrid model, in contrast, dramatically reduced both types of errors (registering only one false negative and one false positive) by implementing mechanisms that explicitly addressed structural attribute integrity. A key factor in this success was attribute weighting and essentiality checks, which allowed the system to differentiate between core, trust-critical attributes (e.g., primary role, department) and peripheral or optional ones (e.g., project tags). The Hybrid model prevented circumvention through partial or misconfigured attributes by assigning higher weights to essential attributes and verifying their presence and structural alignment before granting access. Additionally, it applied trust-based reasoning to detect anomalies in attribute combinations, thereby blocking violations that passed unnoticed in XACML or NGAC. This structured, layered evaluation enabled the Hybrid model to uphold both policy semantics and structural correctness, ensuring high assurance in attribute-based access enforcement.

Overall, the visual comparison demonstrates that the Hybrid model consistently achieves the most reliable, balanced, and context-sensitive access decisions across all evaluated categories. Figure 24 shows a comparative summary of the confusion matrix and results from this experiment.

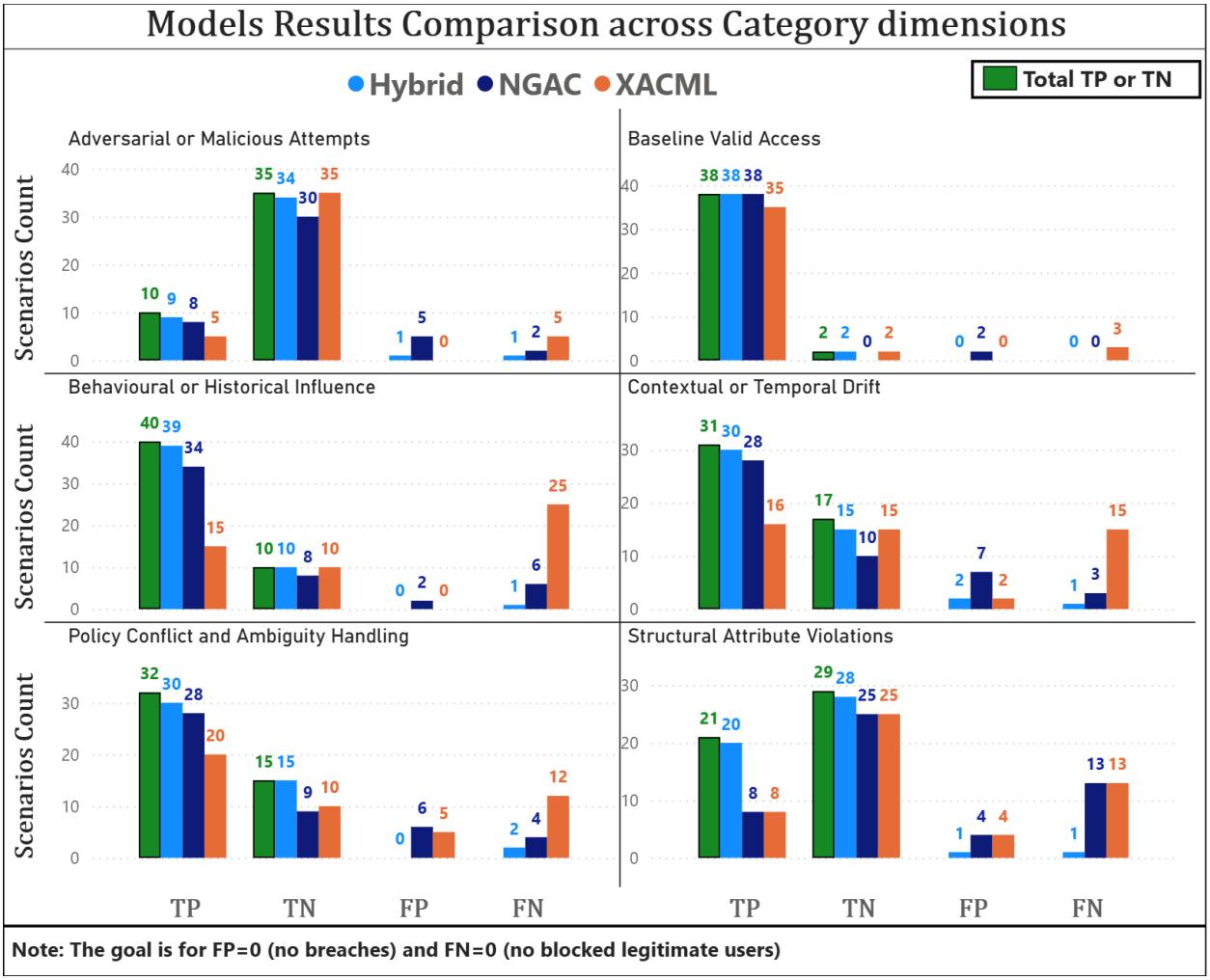


Figure 24 XACML, NGAC, and Hybrid models results comparison summary

Additionally, how each model behaves differently for categories than the Baseline Valid Access is illustrated in Figure 25. This split view highlights model performance in normal (Baseline) versus challenging (Other) scenarios. While Baseline Valid Access shows near-perfect results

with minimal errors for Hybrid, NGAC, and XACML categories, the Other Categories (Adversarial or Malicious Attempts, Behaviour or Historical Influence, Contextual or Temporal Drift, Policy Conflict and Ambiguity Handling and Structural Attribute Violations) reveal significant FPs and FNs. The volume of real-world observed aggregate correctness (TP, TN) hides the relatively small volume of incorrectness (FP, FN).

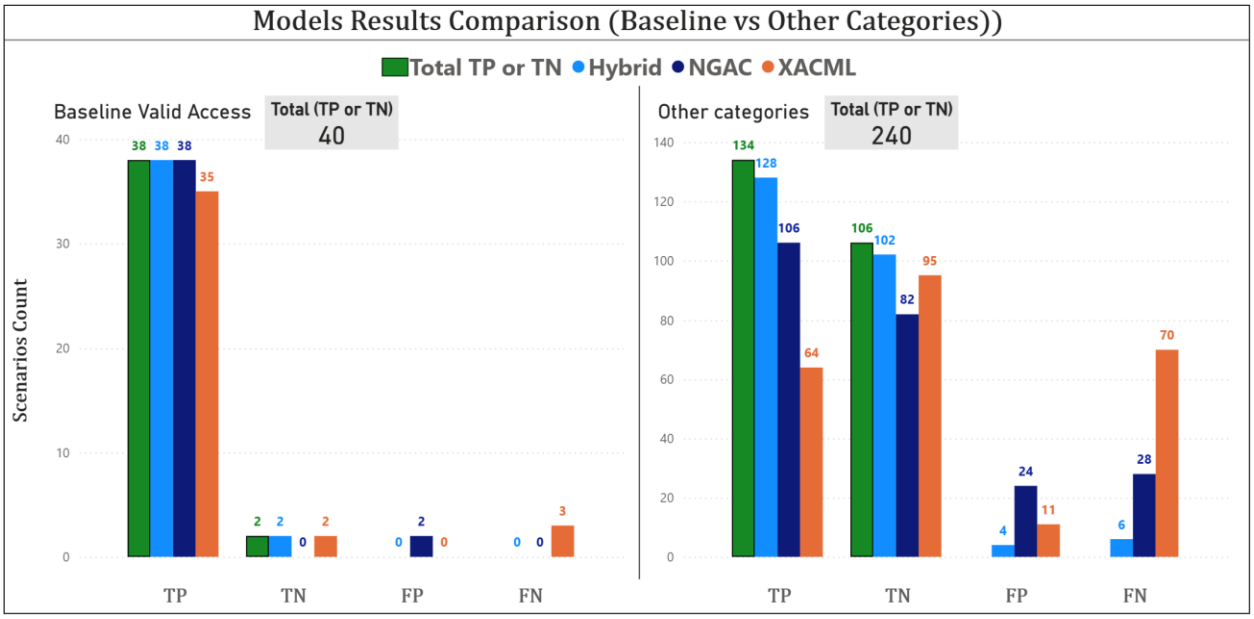


Figure 25 XACML, NGAC, and Hybrid models results comparison summary (baseline vs other categories)

Unlike NGAC, which tends to be overly permissive due to attribute propagation without sufficient contextual checks, and XACML, which often suffers from excessive rigidity and static rule limitations, the Hybrid approach manages to mitigate both extremes. It does so by combining precise attribute validation with dynamic evaluation of risk, trust, and contextual relevance. This enables the model to minimize both false positives and false negatives even in complex scenarios involving temporal drift, structural violations, or policy conflicts. The results affirm the Hybrid model's strength in resolving ambiguity, enforcing structural integrity, and adapting to both historical patterns and real-time environmental factors, making it a

comprehensive and robust framework for addressing the nuanced demands of modern access control environments.

4.4.2. Classification Model Performance Comparison (Hybrid vs Standalone)

Further, based on evaluation metrics, the overall performance can be derived using the formulas well-known in the literature [65], which are presented below, along with the implications of having low or high values in the model's decisions.

1. Accuracy: It is useful for evaluating overall performance.
    a. **High:** Most access decisions are correct overall.
    b. **Low:** Many decisions (grant/deny) are wrong, reducing trust in the system.

$$Accuracy = \frac{TP + TN}{Total\ scenarios} \tag{10}$$

    Note: Although accuracy is mathematically valid, the nearly 2:1 ratio of positives (172) to negatives (108) means errors on negatives (e.g., false positives) contribute less to the overall score. This can make accuracy appear high even if the model fails to reliably deny unauthorized access. Therefore, accuracy alone may be misleading in this security context and should be analyzed along with other metrics.

2. Precision: Measures how many permit decisions were correct. This is critical when the cost of false positives is high, as in potential unauthorized access or breaches, which can have severe and hard-to-remediate consequences.
    a. **High**: When access is granted, it's rarely to unauthorized users (low false positives).
    b. **Low**: Many granted accesses are actually unauthorized (security risk).

$$Precision = \frac{TP}{TP + FP} \tag{11}$$

3. Recall (Sensitivity or True Positive Rate): Measures how well the model identifies scenarios where access should be granted. While important, it is typically a secondary concern here, since false negatives mainly delay legitimate work and can often be mitigated with measures like step-up authentication.
    a. **High**: Most authorized users are granted access (few false negatives).
    b. **Low**: Many authorized users are wrongly denied (poor usability).

$$Recall = \frac{TP}{TP + FN} \tag{12}$$

4. F1-Score or Harmonic mean: This metric is useful when both precision and recall are important and a balance is needed between FPs and FNs. It becomes especially relevant in AC systems where denying legitimate users (FN) or granting unauthorized users (FP) both carry risk.
    a. **High**: Good balance between precision and recall.
    b. **Low**: Trade-offs exist-either too many unauthorized grants or too many denied legitimate users.

$$F1 - Score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{13}$$

Note: In high-security scenarios, however, reducing FPs (security breaches) is often prioritized, even at the cost of increasing FNs. This trade-off can be mitigated by introducing secondary checks like step-up authentication or delayed re-evaluation to recover from initial denials.

The performance comparison shown in Figure 26 reinforces and validates the trends previously observed in the detailed error analysis. While all models achieve reasonably high accuracy in simpler or static scenarios, the nuanced differences become evident when examining precision, recall, and F1-score collectively, metrics that better reflect real-world robustness in access control systems.

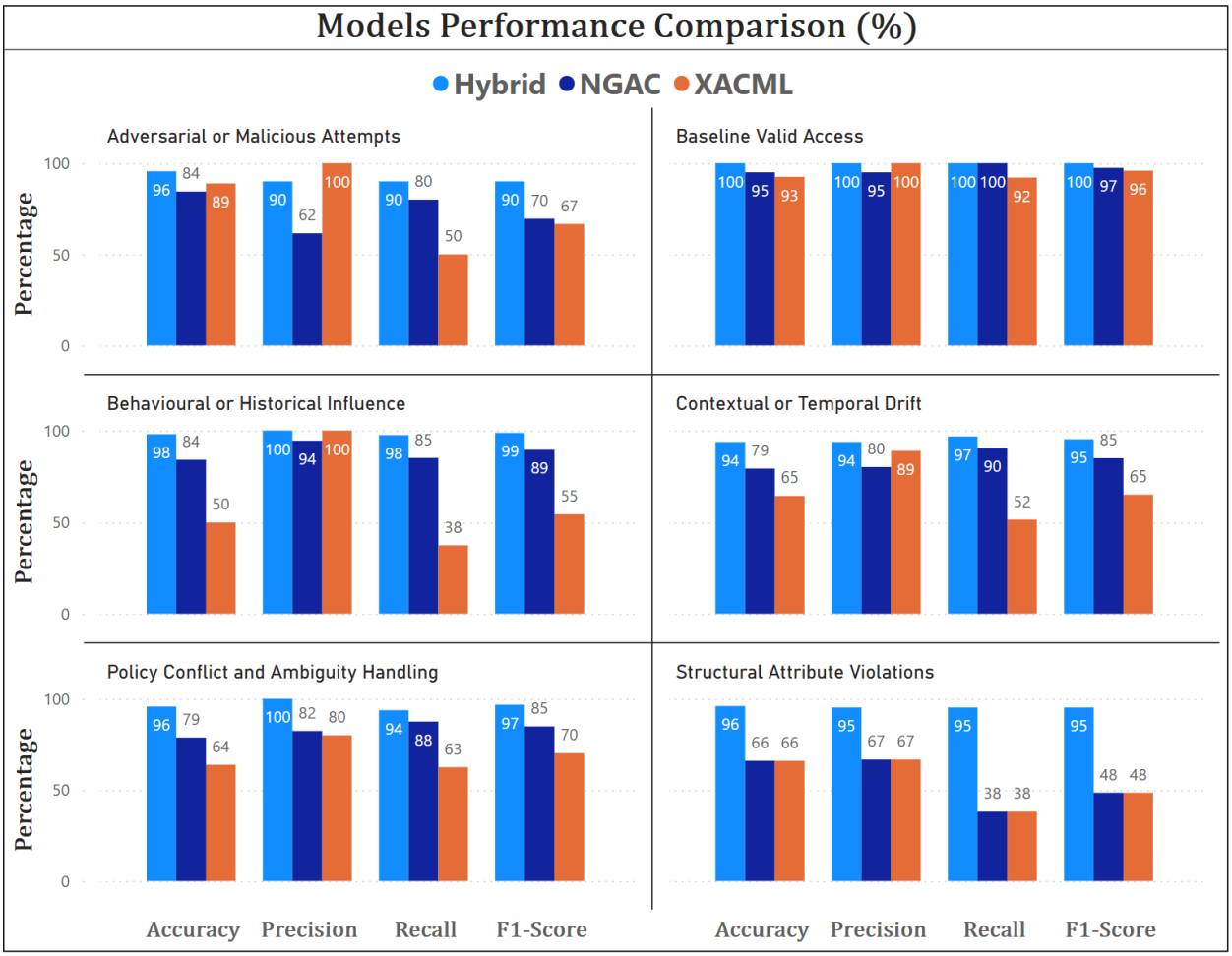


Figure 26 XACML, NGAC, and Hybrid models performance comparison

The Hybrid model stands out across all categories, not simply because it performs well in isolation, but because it does so consistently without compromising one metric in favour of another. This balanced performance highlights its ability to generalize across both static and

dynamic access contexts, a capability that is essential in environments where security, usability, and adaptability must be simultaneously preserved.

Precision, in this case, reflects how confidently the model can distinguish legitimate access from potential attacks without triggering excessive false positives. High precision in the Hybrid model indicates that it avoids over-permissiveness (Scenarios where an attacker can successfully breach the system and are counted in FPs), particularly in scenarios where access paths may appear valid structurally but are contextually inappropriate. In contrast, NGAC's flexible attribute propagation lowers precision. While it helps avoid unnecessary denials that hinder users and add remediation costs, it can also lead to granting access when essential context is missing or unclear. On the other hand, XACML tends to perform better in precision but drops significantly in recall, especially in categories involving behavioural variation or contextual changes. This points to its tendency to over-restrict access when input conditions deviate even slightly from static policy expectations.

The drop in recall means that XACML, although cautious, fails to accommodate legitimate variation in real-world usage, leading to frequent false denials. The Hybrid model's recall, however, remains consistently strong. This indicates its superior ability to recognize valid but non-obvious access requests, those shaped by historical patterns, dynamic trust, or attribute-derived context. This is further affirmed by its high F1-scores, which combine both precision and recall, measuring overall effectiveness. A strong F1-score across categories confirms that the Hybrid model does not trade off between security and accessibility but instead adapts to each scenario with contextual intelligence. Importantly, accuracy, while a useful general metric, can be misleading if viewed in isolation, particularly in imbalanced datasets or systems where

permitted and denied cases are not evenly distributed. The Hybrid model maintains high accuracy alongside strong precision and recall, suggesting that it isn't merely relying on dataset balance but is genuinely making better decisions across a broad range of conditions. This reflects its layered structure, where structural matching (from XACML), contextual adaptability (from NGAC), and trust-weighted evaluation (from both) come together to support precise and risk-aware access control.

4.4.3. Performance comparison

In this experiment, the performance of XACML, NGAC, and DTW-ABAC was compared based on the execution time required to evaluate access. The key definitions used in the following sections are:

- Runs: A single run is defined as an access request or a group of independent access requests that are evaluated once. Hence, the runs (a plural form) is a process of running access scenarios multiple times, implying multiple requests by users over time. As the runs increase, the DTW-ABAC evaluation process uses previous decision history to make future decisions more precise and helps the model to be progressively aware.
- Trust Factor: It is a calculated field described in Chapter 3 and used at every level of evaluation. Trust Factor (TF) shows the confidence in the subject, object, or environment attributes matching the policy definitions, considering the attribute weights (priority).

The 280 scenarios were executed over 20 runs for each model on a local machine, i.e. MacBook Pro (2023 model) with an M2 chip, 16 GB RAM, and a 10-core processor. The machine was restarted before testing each model to clear out CPU load and memory cache. Furthermore, the communication time with Entra ID has been excluded from the analysis, as it is common for all

models and was conducted only once. Figure 27 shows the overall sum of execution time over multiple runs for all three models; however, the lines for DTW-ABAC and NGAC are blended since the numbers are the same. Figure 28 shows the execution time taken for an individual run. At individual runs, the minor fluctuation appears due to the underlying infrastructure, e.g. CPU and operating system scheduling, connection stack, etc., and not due to the model components.

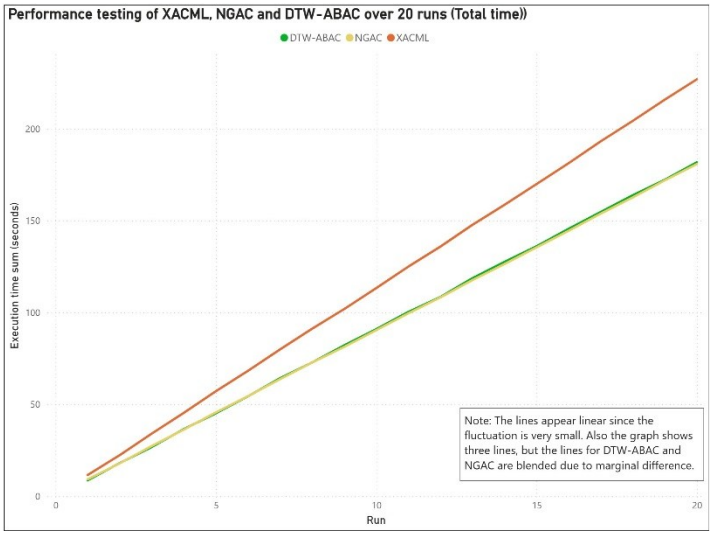


Figure 27 Performance testing of XACML, NGAC, and DTW-ABAC over 20 runs (Total time)

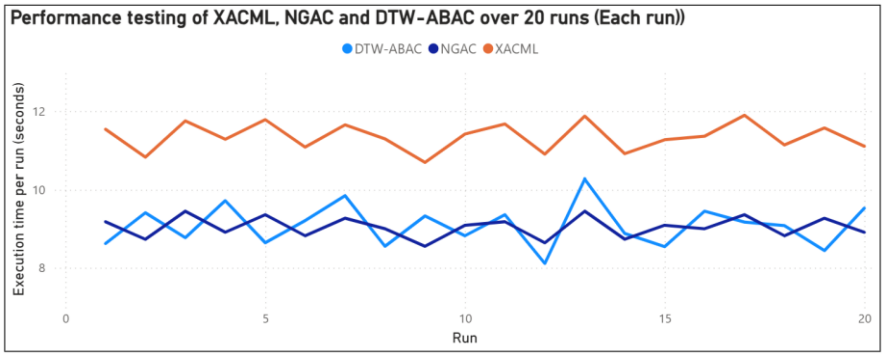


Figure 28 Performance testing of XACML, NGAC, and DTW-ABAC over 20 runs (Each run)

- **XACML (standalone):**

  XACML generally took longer than the other models (20-25%), which is significant in execution. Since each request contains only a limited set of attributes and policies, the XML-based parsing, rule combining, and condition evaluation overheads were noticeable

but not excessive per scenario. Literature also supports that XACML can be heavier than alternatives, especially at larger scales, due to its policy evaluation mechanisms.

- **NGAC (standalone):**

  NGAC performed faster than XACML. NGAC relied on graph operations (set membership and relationships), which were computationally lighter than XML parsing and rule combining. Studies have highlighted that NGAC scales more efficiently as the number of attributes and policies grows.

- **DTW-ABAC (hybrid):**

  Its performance was very close to NGAC (± 3-5%), with negligible differences:

  1) It divided the evaluation tasks across the two engines and ran them in parallel, so execution time was closer to the maximum of the two partial tasks, not the sum.
  2) It included an essentiality check (short-circuiting early when critical attributes failed), sometimes saving time.
  3) It performed additional trust factor and risk calculations, sometimes cancelling part of the parallelism advantage.
  4) The net effect was that DTW-ABAC usually ran at nearly the same time as NGAC, occasionally faster (short-circuits and divided tasks), and occasionally marginally slower (trust computation).

The analysis confirmed that DTW-ABAC did not introduce any execution overhead compared to NGAC. While XACML showed relatively higher times, this difference remained modest in the tested setting.

In summary, the Hybrid model's consistent superior performance across all metrics indicates a well-rounded, context-sensitive enforcement capability. Its strength lies in how it handles both static rules and dynamic attributes, integrating them into a unified decision engine. This results in more accurate and contextually correct access decisions, along with greater operational stability, reduced user friction, and stronger resistance to both over-blocking and policy permissiveness.

4.5. Hybrid Model Parameter Impact Evaluation (Tuning)

This section will show how multiple tests were performed to tune the parameters configured in the hybrid model. Parameters include the constants or weights assigned to integrated models (C_{XACML} & C_{NGAC}) used in chapter 3, the progressive evolution in the decision process, and an ablation study to observe the model variations by including/excluding the attribute weights and essentiality factors.

4.5.1. C_{XACML} & C_{NGAC} constants tuning impact on trust factor (Objective: Measure trust and decision stability across constant values.)

Figure 29 presents the impact of different tuning configurations on the trust factor dynamics of the Hybrid access control model over a series of evaluation runs. Unlike experiments involving standalone NGAC or XACML models, this analysis specifically examines how the Hybrid model's internal blending constants (C_{XACML} & C_{NGAC}) influence the rate and stability of trust factor convergence, as mentioned in the equation (9). These constants control the relative weight assigned to the outcomes of XACML and NGAC engines during access decisions within the Hybrid framework (Figure 9). Importantly, all other variables across these tuning runs, including attribute weights, essential attribute settings (*isEssential* flags), policy definitions, graph topologies, and node configurations, were held constant to ensure that only the isolated effect of

the tuning constants was observed. Additionally, for each model variation, the history was cleared (permit = 0, deny = 0 and total previous requests = 0) to observe the result in a cold start or fresh start setup. The graph shows 5 different values assigned to the model constant; those values were derived by performing an iterative test with different constant values and then selecting the ones which made a significant impact on the results. The optimal numbers are based on the number of scenarios tested, and they can be improved in future with a larger dataset.

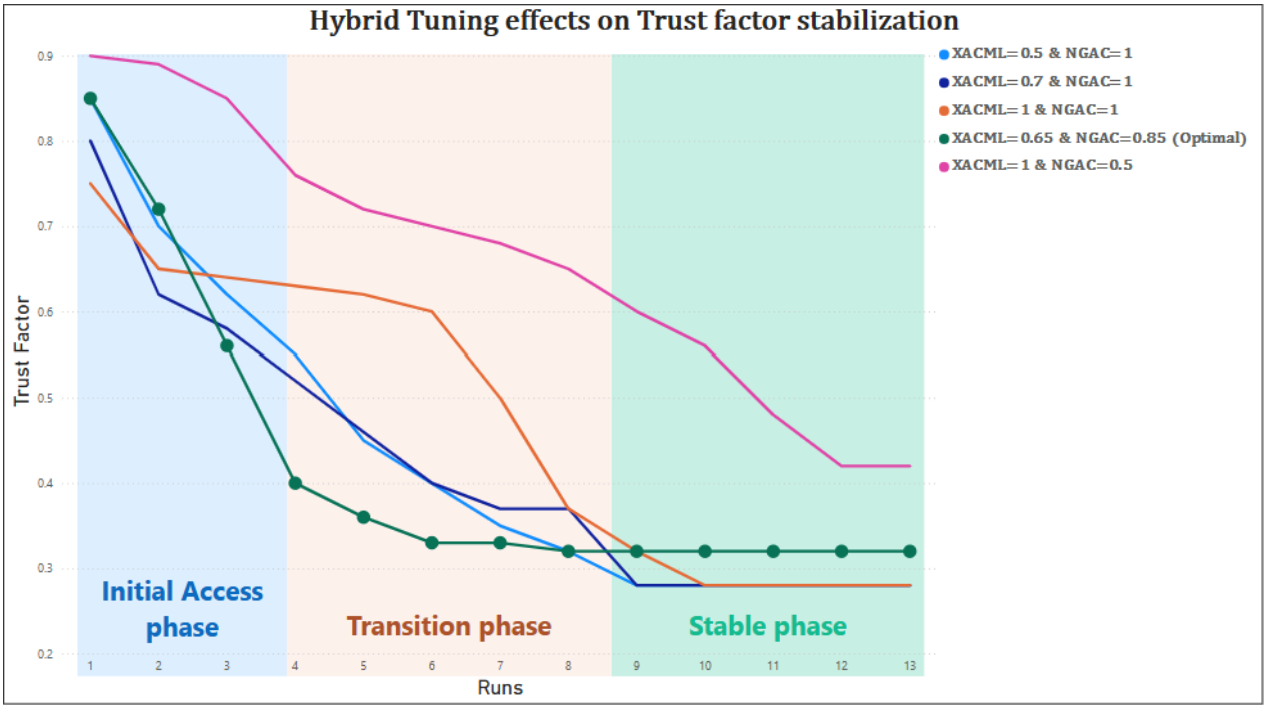


Figure 29 Hybrid tuning effect on trust factor stabilization

Each run on the x-axis refers to the complete evaluation of the access scenario, executed once per run. In the actual setup, all 280 scenarios were tested with different model weights. The patterns were the same for all scenarios since only the model weights were changing, and other parameters were kept constant. Notably, the trust factor computation includes historical data, i.e., the trust value at any point in a run is influenced by the current scenario's decision correctness as

well as by cumulative past performance, which includes both permits and denials. This history-aware design ensures that the system captures not just momentary decisions, but the overall consistency and risk sensitivity of the Hybrid model across evolving access conditions.

The phases indicated in the background shading of the plot denote distinct behavioural transitions observed in the trust factor evolution:

- Initial Access Phase (Runs 1–3): The model is exposed to the first few batches of access scenarios. Trust levels begin high, as no prior negative behaviour is recorded, but they decline rapidly as the model begins encountering and adapting to complex or risky requests.
- Transition Phase (Runs 4–8): The system enters a critical learning phase, where the trust factor experiences continued drops, reflecting both accumulated false decisions and increasing policy conflicts or drift cases. This is a phase of adaptation and convergence where the model learns to balance risk and permissiveness.
- Stable Phase (Runs 9–13+): By this point, the model's decisions have mostly aligned with expected access patterns, though the pink line lags behind due to heavier reliance on XACML.Trust stabilizes (remains steady), with minimal fluctuations, indicating a mature balance between correct permissions and denials.

Among the different configurations tested, the curve labelled as C_{XACML} = 0.65 & C_{NGAC} = 0.85 (marked as Optimal) demonstrates the fastest and most stable convergence toward a low, yet steady trust factor (not too high and not too low). This indicates that this tuning best supports effective risk discrimination, i.e., the model correctly penalizes uncertain or adversarial access attempts early while avoiding excessive harshness that could destabilize the trust score in later runs. In contrast, configurations like C_{XACML} = 1 & C_{NGAC} = 0.5 maintain a much higher trust

factor throughout, reflecting a lag in penalizing risky or ambiguous access. This setup overweights the XACML component and underutilizes the dynamic responsiveness of NGAC, leading to prolonged optimism and reduced sensitivity to behavioural or contextual anomalies. Meanwhile, configurations with C_{XACML} = 1 & C_{NGAC} = 1 or lower XACML weights (e.g., 0.5 or 0.7) show moderate convergence behaviours. However, without appropriately calibrating the NGAC contribution, some configurations either become too reactive, dropping sharply into over-restrictiveness, or too lenient, failing to distinguish well between legitimate and suspicious trends. In the transition phase, especially, the slope of decline reflects how quickly the hybrid model adjusts its decisions in response to evolving scenario patterns. Configurations with balanced or NGAC-favoured weights exhibit a smoother, more strategic descent, suggesting improved adaptability.

In summary, this graph illustrates not only the impact of Hybrid model tuning on trust factor dynamics but also how different weightings influence the model's learning trajectory through historical scenario exposure. It confirms that precise tuning of blending constants is essential to achieving optimal performance in dynamic, context-rich environments, where access permission is not just a matter of static policy matches but of sustained behavioural intelligence and adaptive enforcement.

4.5.2. Trust factor change effects on the functional metrics

To evaluate the Hybrid access control model’s ability to learn from access history and adapt to dynamic access contexts, a controlled multi-run experiment was conducted. The goal was to assess how well the model could refine its trust-based decision-making over time, without altering any core parameters. By repeatedly exposing the system to a consistent set of scenarios

across multiple runs, the test aimed to reveal the model's capacity for behavioural memory, risk recalibration, and improved discrimination between valid and malicious access attempts. Figure 30 illustrates how the Hybrid access control model adapts and improves over time when evaluated across ten sequential runs of 280 scenarios each. At the beginning of the first run:

- **No prior behaviour or decision history was considered**: This mimicked a cold-start or first-time evaluation scenario.
- **All attribute weights, policy rules, and thresholds** (such as trust score thresholds for allow/deny) were held constant and defined as per the configuration outlined in Section 3.2.3.
- No manual interventions or heuristic modifications were applied during the runs; changes in performance metrics were entirely the result of adaptive risk re-evaluation based on accumulated behaviour patterns.

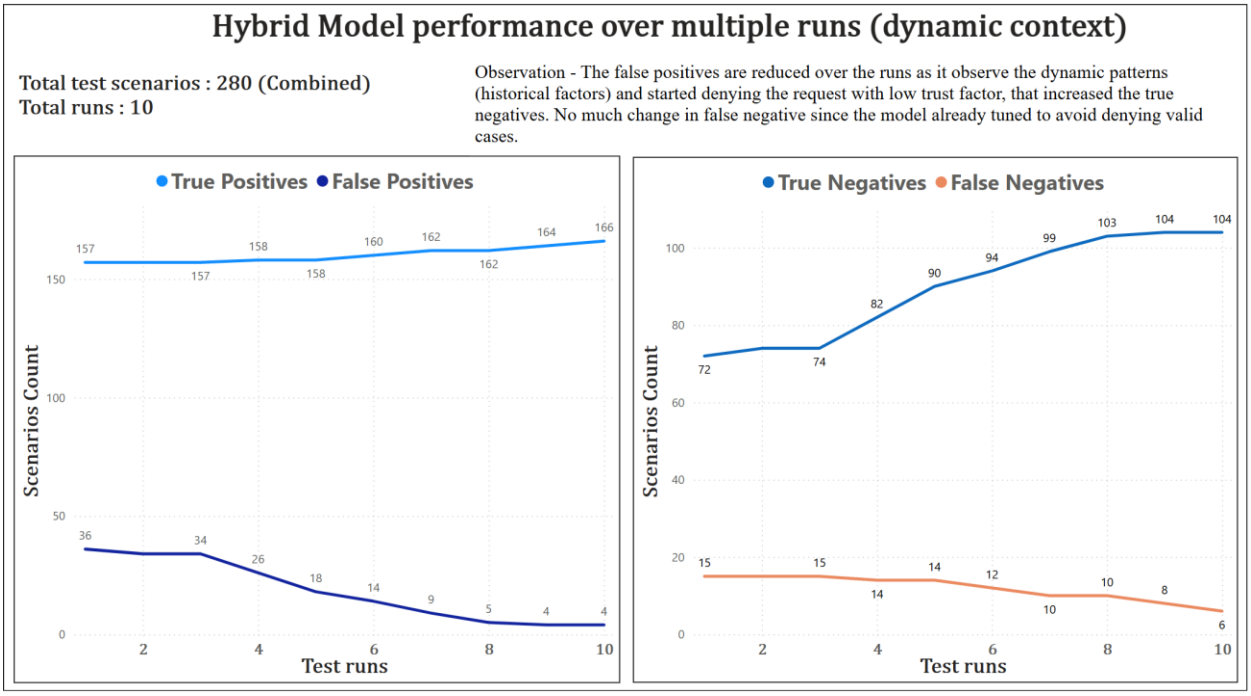


Figure 30 Hybrid model performance over multiple runs

When examining True Positives (TP) and False Negatives (FN) together (the metrics that represent how the model handles valid access requests), it can be observed that TP gradually increases from 157 in the first run to 166 by the tenth run, while FN decreases from 15 to 6. This inverse relationship highlights the Hybrid model's strengthening ability to correctly identify and permit legitimate access while reducing incorrect denials. The gains are modest but significant, particularly because the model was already tuned to avoid over-blocking valid users. It results in a slight enhancement to the recall formula, i.e. $\left(\frac{TP}{TP+FN}\right)$, which indicates the minor adjustments the model makes as it gathers historical trust data and fine-tunes its assessment of borderline situations.

On the other side, the combination of False Positives (FP) and True Negatives (TN) reveals how the model improves in denying inappropriate access. FP declines steeply from 36 to 4, while TN rises from 72 to 104 over the same sequence of runs. This trend shows that as the model undergoes testing with more runs, it becomes more effective at identifying and rejecting malicious or structurally invalid requests. The trust mechanism plays a key role here, i.e. as the system detects patterns of misuse, it lowers trust factors for those users or contexts, leading to more confident denials in subsequent runs. This results in a higher number of true negatives and a significant reduction in mistakenly permitted risky accesses.

Together, these shifts show that the Hybrid model becomes more discriminative and reliable over time. It preserves access for legitimate users while also actively learns to block inappropriate access based on behavioural patterns and cumulative experience. Even without altering any core parameters, the trust-aware, history-driven adaptation allows the system to optimize its decisions dynamically, achieving a more stable and secure balance between permissiveness and control.

4.5.3. Ablation study on model components such as attribute weighting and essentiality

In cybersecurity research and access control system design, an ablation study serves as a critical diagnostic method for understanding the individual contribution of each model component to the system's overall effectiveness. Within the context of the DTW-ABAC model, which integrates layered evaluation through both structural rule-based logic (XACML) and contextual dynamic trust computation (NGAC), the ablation study becomes especially relevant. DTW-ABAC introduces two central enhancements beyond traditional ABAC approaches: the incorporation of attribute-specific weights and the designation of certain attributes as essential for access to be permitted. These two components (weights and the *isEssential* flag) influence how trust factors are computed and fundamentally shift how decisions adapt to partial matches, uncertainty, and contextual drift.

In traditional ABAC systems, attributes are evaluated in a binary fashion: they either match or do not. This rigid structure treats all attributes as equally important and offers no mechanism to differentiate between a missing low-impact attribute (e.g., location) and a missing critical one (e.g., clearance level). The DTW-ABAC model attempts to overcome this limitation by introducing numerical weights to indicate the relative importance of each attribute. A higher weight increases an attribute's contribution to the rule's trust factor, thereby amplifying its influence on the access decision. Similarly, the *isEssential* flag allows designers to specify attributes that must match for access to ever be considered, regardless of the cumulative trust score from other attributes. This dual mechanism introduces granularity as well as resilience, permitting the system to reason through uncertainty in a principled way.

To test whether these enhancements actually contribute to model performance or simply add complexity, an ablation study was conducted by systematically removing or altering them. Figure 31 shows the performance results of several access scenarios (mostly edge cases) tested under different model conditions. The baseline is the full DTW-ABAC model, which uses both weights and essential flags across a hybrid structure combining XACML and NGAC.

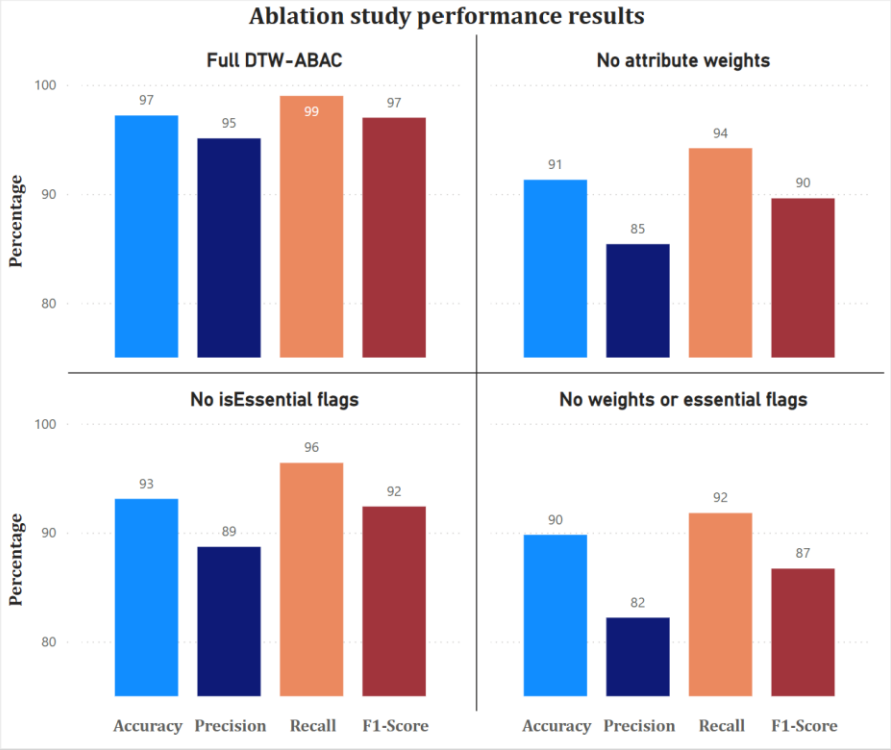


Figure 31 Classification metrics ablation study for DTW-ABAC variants

In the first variation, attribute weights are removed, and all attributes are treated equally. This means the trust factor calculation becomes a simple average of matched attributes, and there is no distinction between high-impact and low-impact attributes. As a result, the model permits access even when only superficial, non-critical attributes match, leading to higher false positives. Conversely, the model also blocks access if low-weighted but unmatched attributes drag down the overall trust factor, thus creating false negatives.

In the second variant, the attribute weights are retained, but the *isEssential* logic is disabled. This scenario evaluates whether weights alone are sufficient to prioritize important attributes or if explicit essentiality is required to enforce hard constraints. Results here often show that while weighting helps balance the decision mathematically, it lacks the decisiveness required in high-stakes scenarios, such as preventing access when clearance or authentication attributes are missing. Without the *isEssential* gatekeeping, access might still be granted based on high matches in other areas, even though a core requirement has failed, creating a potential vulnerability.

Another variant disables both weights and essential flags, reverting to a flat trust model similar to naive ABAC, where each attribute contributes equally and all attributes are treated as optional. This model performs the weakest, confirming that the system loses precision and adaptability without the ability to differentially score or require attributes.

This ablation study gives a clear picture of the necessity of both attribute weights and essentiality enforcement in real-world access control. When both components are active, the model demonstrates higher F1-scores, improved recall for context-aware access, and lower rates of both false positives and false negatives. It is particularly effective in edge cases, such as partially-matched requests during temporal drift or in dynamic environments where users have fluctuating trust scores.

4.5.4. Attribute-weights tuning (Manual changes and observations)

Multiple independent test runs were performed by assigning different weights to the attributes used in the research to find out the optimal weights and essentiality property, based on which

weights result in higher TP and TN. After every test run, the results were moved to different locations so that the historical factor would not influence future runs. Initially, the attribute weight was set to 1 (default) for all the attributes and captured the hybrid model's results. Further, started increasing the attribute weight based on the usage frequency (number of times it was used in the XACML policies and NGAC graphs) and policy impact (if the attribute was referenced in the critical policy rule), as well as changing the isEssential flags and observed different results. Thus, the process of adjusting attribute weights was guided by analyzing iterative test results, suggesting a direction for future work to develop a measurable method. Table 15 shows some of the important attributes with weight, *isEssential* property (1 or 0) and summary to provide the reason behind the weights. High-weighted attributes include role, clearance, deviceCompliance, trustScore, accessPriority, whereas low-weighted attributes include shift, regionAffinity and location. However, the weights will change depending on the attribute usage frequency and position in the XACML policy and NGAC graphs.

Attribute Name	**Weight (W_i)**	**isEssential**	**Reasoning Summary**
role	5	1	Core determinant of user function and access eligibility
clearance	5	1	Security tiering is required for high-risk or sensitive resources
projectTag	3	0	Useful for narrowing access, but not critical alone
department	3	1	Required in tightly scoped policies (e.g., HR vs. Engineering access separation)
location	2	0	Contextual, relevant for policy drift or geographic restrictions
deviceCompliance	4	1	Crucial for verifying secure endpoint access

environmentType	3	0	Adds temporal or operational nuance, but not mandatory
shift	1	0	Helpful for time-sensitive access, low enforcement criticality
accessLevel	4	1	Determines read/write/admin permissions; critical for authorization boundaries
accessPriority	5	1	Priority indicator for emergency access
regionAffinity	2	0	Often, a soft preference or fallback condition
team	3	0	Indicates lateral group identity; helpful but not always enforced
docType	3	1	Required for policies tied to data classification levels
classification	4	1	Indicates document/resource sensitivity; critical in access limitation
loginPatternScore	3	0	Supports behavioural analysis, but is considered adaptive and supplementary

Table 15 Optimal weights and "isEssential" property for the attributes

4.6. Risk Level changes over multiple runs

This test was conducted using a refined subset of 160 access scenarios, selected from the initial pool of 280. The selected batch was intentionally curated to stress-test the Hybrid model under varied and challenging conditions. Specifically, the subset included: 45 adversarial or malicious attempts, 50 structural attribute violations, 45 contextual or temporal drift scenarios, and 20 valid baseline requests. These proportions were chosen to increase the representation of ambiguous or high-risk scenarios, allowing the model's dynamic adaptability to be evaluated across multiple iterations.

The experiment was run over 20 sequential evaluation cycles (runs). Each run executed all 160 scenarios in the same order and configuration to maintain consistency and allow the model to evolve based on historical access patterns. The trust and risk evaluations in each run were updated per user scenario, using the Hybrid model's trust-based scoring formula outlined in Equation (6) in Chapter 3. This formula dynamically adjusts a user's trust score based on past behaviour, attribute matching, and decision outcomes. The resulting score determines a user's risk classification: **low**, **medium**, or **high**.

Experiment Setup Details:

- **Initial State**: All users were treated as new during the first run. Their historical trust scores were initialized to a neutral baseline, implying no prior positive or negative behaviour. This ensures that initial decisions were purely based on attribute matching and policy structure.
- **User and Application Simulation**: The 160 scenarios were distributed across 8 simulated user profiles. Those users were included in unique access scenarios across 10 applications covering deviant behaviours to emulate real-world adversity. These users persisted across runs, accumulating a trust history.
- **Scenario Design**: Scenarios were manually labelled and categorized (e.g., malicious, structural, contextual drift) during preprocessing. No random reshuffling was done between runs to ensure repeatability.
- **Trust Score Parameters**: The scoring formula incorporated penalties for critical attribute mismatches and gradual decay for trust scores. Essential attributes contributed more weight to the risk score than optional ones, and repeated violations increased the

risk factor. While exact constants (e.g., Model constants, Attribute weights) are determined in other experiments, no manual adjustments were made between runs to preserve fairness.

As illustrated in Figure 32, the progression of risk classification shows a clear trend:

- **Run 1**: Users are primarily classified as **low-risk**, as no behavioural history exists yet. No users fall into the medium-risk or high-risk category at this point.
- **Runs 2–5**: The **medium-risk** category begins to emerge, capturing users whose access patterns are borderline or inconsistent.
- **Runs 6–15**: The **low-risk** population declines steadily as early signs of policy violations are detected. Users with repeated violations shift toward **high-risk**.
- **Runs 16–20**: Risk distribution stabilizes. Most users are now in either the **high** or **medium** risk categories, with the **low-risk** group limited to consistently authorized users across all runs.

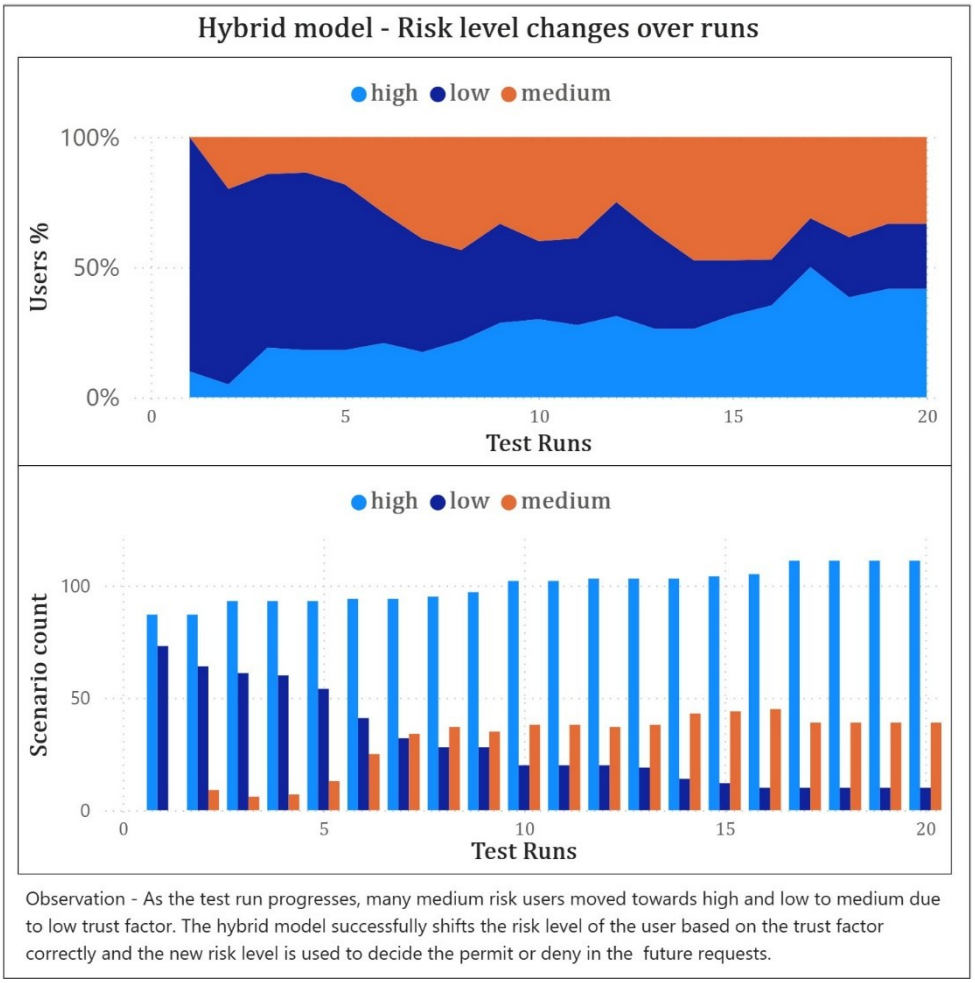


Figure 32 Hybrid model risk level changes over multiple runs

This dynamic evolution of user classification demonstrates the model's ability to self-adjust risk estimates without manual intervention. It starts optimistically but becomes more cautious over time as it learns from behaviour. This shifting risk landscape directly influences future access decisions by adjusting the **confidence factor (H)** in Equation (6), thereby strengthening the model's capacity for long-term trust calibration.

4.7. Result sensitivity in different real-world datasets

The hybrid access control model was not validated on a proprietary dataset because no openly available dataset could be found that included both policies, graphs and user-level attributes together. Such datasets are typically restricted due to intellectual property and security concerns, as enterprises do not release sensitive access control information. Instead, the official documents and published standards mentioned in the literature review were referred to as design representative policies, graph structure, and attribute lists, as mentioned in the methodology and appendix sections.

To strengthen realism, simulated scenarios were generated through structured methods such as policy-driven filtering and FSM-based modelling. These scenarios incorporated conditions reported in prior research, i.e. benign, adversarial, contextual drift, and structural violations, ensuring that the evaluations aligned with known real-world access control challenges.

The evaluation framework itself is mathematical and independent of workload size. Trust factor calculations and policy evaluations apply consistently, whether the dataset is small or large. However, the accuracy depends on attribute richness: sparse or poorly structured attributes weaken reliability, while richer datasets strengthen the evaluation precision.

The model was tested under varied conditions, including changing request volumes, shifting attribute weights, behavioural drift, and adversarial cases, and the results remained stable and repeatable. This provides evidence that the framework, even when built on reconstructed datasets, reflects a wide range of real-world access control conditions.

4.8. Summary

The testing and experimentation chapter was grounded in a real-world access control context, simulating scenarios commonly found in enterprise environments such as data classification, role-based access, contextual restrictions, and dynamic trust assessments. The foundation began with the manual design of a controlled product suite, where users, roles, resources, and policies were constructed to reflect practical organizational structures. To enhance realism and reduce static bias, a finite state machine (FSM)-based generation mechanism was then developed to programmatically produce access requests with varying attribute patterns, behavioural histories, and contextual shifts. These synthetic requests captured a wide spectrum of access scenarios, including benign, risky, and adversarial patterns.

To ensure meaningful interpretation, generated access attempts were then categorized into six distinct types, i.e. valid baseline, adversarial, behavioural drift, contextual drift, policy conflict, and structural violations, allowing clearer grouping of results and stronger external validity. The comparative evaluation of XACML, NGAC, and the hybrid DTW-ABAC model was conducted by logging true positives, false positives, true negatives, and false negatives across each category. This classification helped discover the strengths and weaknesses of each model, revealing how over-blocking in XACML or over-permissiveness in NGAC manifests in quantifiable decision failures.

Beyond categorical accuracy, the study measured key performance metrics such as precision, recall, F1-score, and accuracy to highlight trade-offs in enforcement behaviour. These metrics were further examined across different model configurations by tuning hybrid parameters like the weight constants for XACML and NGAC and observing their impact on the overall trust

factor output. By adjusting the hybrid's internal constants, trust factor distributions shifted, often resulting in changes to the assigned risk level of access requests. This illustrated the hybrid model's capacity to adapt its decision boundary based on risk sensitivity or policy intent. Finally, an ablation study was performed to evaluate the specific contributions of attribute weighting and the use of *isEssential* flags. Removing or flattening these components degraded the model's contextual intelligence and weakened its ability to prioritize critical attribute matches, reaffirming their necessity. Together, the experiments validate DTW-ABAC as not only a high-performing model but also one rooted in practical, auditable, and interpretable access control logic.

4.9. Future Experimental Opportunities

While the current experiments thoroughly validated the hybrid framework's effectiveness, several experimental avenues remain open for future exploration:

4.9.1 Larger and More Diverse Datasets:

The present experiments were performed over a controlled set of 280 scenarios, providing detailed insights into system behaviour. However, future research could expand this analysis by applying the framework to larger, real-world datasets. This would enable evaluation of performance, scalability, and decision accuracy under conditions more reflective of enterprise-scale deployments.

4.9.2 Balanced Dataset for True Positive and True Negative Scenarios:

The current test scenarios had specific distributions of legitimate versus malicious access attempts. Future experiments could develop and evaluate more balanced datasets,

containing evenly distributed true positives and true negatives. Such an approach would better assess the model's performance under varying conditions.

4.9.3 Red Teaming and Adversarial Simulations:

Further validation could include comprehensive red-teaming exercises, simulating sophisticated real-world adversaries attempting to bypass or manipulate the access controls. These exercises would provide deeper insights into system robustness and expose any subtle weaknesses not captured by the current scenarios.

4.9.4 Ablation study at the attribute level

Future work should include conducting an ablation study by systematically removing one attribute at a time, rather than removing one model component (such as attribute weight or *isEssential* flags). After removal of each attribute, the model's performance should be evaluated using performance metrics. The results should then be ranked based on the primary evaluation metric, such as Precision, which is critical for minimizing false positives in security breach prevention, and secondarily by F1-score to balance Precision and Recall without significantly penalizing either.

Chapter 5

Conclusion and Future Work

In today's dynamic and distributed digital environments, traditional access control mechanisms are increasingly falling short. Systems that rely solely on predefined rules or static user-role mappings often fail to respond appropriately to real-time changes in user behaviour, environmental conditions, or evolving organizational risks. This has created a critical gap in ensuring secure, context-aware, and trust-sensitive access to sensitive data and services. This research began by identifying the shortcomings in current access control solutions, particularly in their inability to dynamically adapt to runtime factors or leverage past user behaviour to evaluate future decisions. To address these gaps, a practical implementation of a novel hybrid access control framework named DTW-ABAC has been developed that unifies the strengths of two existing access protocols, XACML and NGAC, and has been evaluated through a series of comprehensive experiments. The research questions presented in Chapter 1 (Section 1.8) are addressed as follows.

1. Hybrid Access Control Framework Combining XACML and NGAC

   Research Question:

   How can the hybrid integration of static and dynamic models improve access correctness in terms of Accuracy, Precision, Recall, and F1-score?

   Findings:

   XACML provides the ability to express detailed, rule-based policies, while NGAC introduces graph-based structures that enable reasoning over dynamic user-resource relationships and evolving system contexts. The layered integration of these two paradigms

resulted in a robust and adaptive decision engine that evaluates dynamic environmental and behavioural contexts in addition to static rules. For instance, access decisions could now be influenced by conditions such as previous access history, risk level and accumulated trust metrics, alongside the standard subject-object-environment triad. This hybrid structure proved especially effective in resolving the tension between the restrictiveness of XACML's predefined rule sets and the over-permissiveness of NGAC's context-aware graph evaluation. While identity providers such as Microsoft Entra ID offer robust authentication mechanisms, they lack the fine-grained, context-driven authorization models required for truly adaptive security. Entra ID was used in this research for identity verification and user attribute provisioning, but the decision-making framework is deliberately designed to remain independent of Entra or any specific identity provider. This ensures portability, vendor-neutral integration, and long-term flexibility for future migrations to other ecosystems.

2. Trust-Based Risk Evaluation with Adaptive Risk Reclassification

   Research Question:

   How can user access history and trust metrics be used to update risk posture dynamically?

   Findings:

   The second major contribution of this research is on Trust-Based Risk Evaluation. Traditional access control often treats all authenticated users as equal, ignoring their historical behaviour or contextual trustworthiness. This work introduced a trust computation model that continuously updates user risk posture based on their access consistency and alignment with organizational standards. The calculations were supported by introducing new mathematical formulas, which are used at each step of the evaluation. The system classifies users into dynamic risk bands (low, medium, high), and these classifications

influence future access decisions. For instance, a user who begins to access unfamiliar resources at unusual hours or from anomalous locations might be flagged for elevated enquiry. This approach empowers proactive scrutiny, allowing access boundaries to shift dynamically in response to deteriorating trust rather than reacting post-incident. The model ensures that access control is not just rule-bound but behaviorally intelligent.

3. Scenario Diversification through FSM-Guided Policy Testing Framework

Research Question:

How can realistic and policy-relevant access scenarios be systematically generated for robust evaluation?

Findings:

To ensure that the hybrid access control framework is tested under realistic and meaningful conditions, this research combined both manual and programmatic scenario generation. Manually crafted cases, inspired by real-world domains like healthcare and engineering, highlighted how XACML and NGAC respond in different ways. These served as benchmarks for validating the broader dataset. A larger set of access requests was generated by combining users, applications, and permissions. This raw pool was then filtered using FSM-inspired logic to create structured access scenarios. Each scenario followed a logical path, i.e. attribute assignment, trust evaluation, and decision-making. To increase test depth, small changes were introduced to create “near-miss” cases that simulate errors or adversarial behaviour. In total, 280 scenarios were selected and grouped into six independent categories, each representing a unique class of access behaviour. This helped measure how the model performed across different conditions. The categories also supported more focused analysis of fault detection and decision accuracy. This approach ensured the system was not just

functionally correct but resilient to edge cases, unpredictable behaviour, and evolving contexts. It demonstrated the practical benefits of the hybrid model's design choices, including attribute weighting strategies, trust evaluation mechanisms, and the integration of XACML and NGAC for more reliable access control.

4. Attribute Governance and Weighted Policy Enforcement Strategy

   Research Question:

   How can attribute criticality and governance be embedded within access control frameworks?

   Findings:

   In addition to dynamic evaluation, this research emphasized the importance of Attribute Governance and Weighted Policy Enforcement. Not all attributes carry equal significance in access decisions, yet many existing systems treat them uniformly. The proposed governance framework differentiated between critical (e.g., role, clearance level) and supporting attributes (e.g., department, device type), assigning them varying weights based on their security implications. Policies could now be written with attribute prioritization in mind, requiring that certain high-weight attributes match before others are even considered. This added layer of precision made the framework more secure, transparent and easier to manage. It allowed policy authors to enforce more nuanced access rules while maintaining clarity in how decisions were derived.

5. Explainable Decision Model with Traceable Evaluation Paths and Integration Support

   Research Question:

   How can access control models ensure transparency, auditability, and operational applicability?

   Findings:

   The research addressed a common criticism of modern access systems, i.e. the lack of transparency and traceability in decision-making. Security decisions often appear opaque to users and even administrators, making it difficult to understand why access was granted or denied. To counter this, an explainable decision model was designed, where each access evaluation followed a traceable path from attribute collection to the final authorization decision. This structured flow supports detailed audit logs, rule-level justifications, and clear visibility into the logic behind each decision. Additionally, the model is designed for integration across varied systems, ensuring consistent governance while enabling policy refinement through post-decision analysis.

In summary, this thesis offers a comprehensive solution to the modern access control problem. The implemented framework sets a new benchmark for adaptive authorization systems by combining policy expressiveness, contextual adaptability, trust-awareness, governance clarity, and explainability. The design choices made, such as keeping the model identity-provider independent (Entra ID) and enabling test and production modes, make it both practical and forward-compatible for real-world deployment in enterprise, healthcare, education, and government sectors. This work not only validates the feasibility of a hybrid access control model but also demonstrates its necessity in a world where security must be both rigid enough to prevent abuse and flexible enough to adapt to change. With the rising complexity of access

environments that are driven by remote work, cloud adoption, and identity integration, such an approach is no longer a bonus but a strategic necessity.

5.1. Future work

While the DTW-ABAC hybrid framework presented in this research successfully demonstrates adaptive and trust-aware access control, there are several promising directions to extend its capabilities and real-world applicability.

First, future work should focus on ensuring that the data feed from Entra ID to the DTW-ABAC model operates on a regular schedule or, ideally, in near-real time. Delays or inconsistencies in this synchronization introduce security risks, such as the potential for a disgruntled employee or a compromised account to exploit stale or cached elevated permissions. Maintaining a timely feed is also essential to reducing the volume of attribute and policy requests sent from DTW-ABAC to Entra ID, thereby improving system efficiency and minimizing latency in access decision-making.

Second, the current framework design must be validated under enterprise-level scalability. While the DTW-ABAC follows a RESTful, stateless architecture that supports horizontal scaling behind a load balancer [57], and the evaluation engine developed in .NET 8.0 offers lightweight hosting and high concurrency [66], these remain theoretical assurances. SQL Server provides well-established scaling patterns, including read replicas, partitioning, and in-memory features [67], yet the framework has not been empirically tested under large-scale workloads. Future work should therefore involve controlled experiments and large-dataset benchmarks to provide rigorous, statistically valid evidence of performance and resilience in enterprise environments.

Third, the framework's layered architecture generates extensive access logs, capturing attribute evaluations, policy decisions, contextual inputs, and trust reclassifications over time. Currently, these logs are primarily used for traceability and audit. However, they present a rich opportunity for building a comprehensive reporting and analytics layer. Future work can focus on designing Key Performance Indicators (KPIs) to summarize trends in user risk posture, frequency of policy conflicts, access denials by category, or drift in contextual attributes. Visualization dashboards can help security teams proactively detect anomalies, evaluate policy effectiveness, and refine trust thresholds. Integration with SIEM (Security Information and Event Management) platforms could further streamline operational oversight and incident response.

Fourth, although the testing has been done to find out the attribute weights and essentiality, the process of setting it in the environment is currently manual and static, thus requiring several iterations to set up. Future improvements can leverage AI (Artificial Intelligence)/ML (Machine learning) algorithms such as feature importance models, reinforcement learning, or unsupervised clustering to dynamically adjust the attribute weights based on access history, policy performance, or emerging risks. This will enhance accuracy and reduce administrative burden in evolving environments. For instance, attributes frequently appearing in false negative outcomes (where access is wrongly denied) may indicate misalignment with policy intent and could be reassessed or decreased in weight. In contrast, attributes consistently associated with true positive decisions and low-risk access may be considered strong indicators of legitimate behaviour and assigned greater importance. Such self-tuning capability aligns with the broader trend of autonomous policy optimization.

Finally, future research can also explore interoperability with decentralized identity frameworks, built around decentralized identifiers (DIDs) and cross-domain attribute verification. This can be implemented where the attributes are issued and verified across organizational or governmental boundaries. Combined with fog and AI enhancements, this could enable a scalable, self-learning, and geographically aware access control infrastructure.

Together, these enhancements would increase the intelligence, performance, and scalability of the DTW-ABAC model in addition to positioning it as a viable candidate for next-generation adaptive access control in critical, distributed, and high-risk domains.

References

[1] Information Technology Laboratory, “Personal identity verification (PIV) of federal employees and contractors,” Jan. 2022. doi: 10.6028/NIST.FIPS.201-3.

[2] P. A. Grassi *et al.*, “Digital identity guidelines: authentication and lifecycle management,” Gaithersburg, MD, Jun. 2017. doi: 10.6028/NIST.SP.800-63b.

[3] The World Bank, “Customer Authentication in Payments,” *Fast Payments Toolkit SEPTEMBER 2021 B*, Washington D.C., United States, 2021. Accessed: Jun. 06, 2025. [Online]. Available: https://fastpayments.worldbank.org/sites/default/files/2021-10/Customer_Authentication_Final.pdf

[4] Microsoft, “Microsoft Security and Entra Documentation Collection.” Accessed: May 02, 2025. [Online]. Available: https://learn.microsoft.com/en-us/entra

[5] Statistics Canada, “Survey of Digital Technology and Internet Use, 2023,” Sep. 2024. [Online]. Available: https://www150.statcan.gc.ca/n1/daily-quotidien/240917/dq240917c-eng.htm

[6] W. Stallings, L. Brown, M. D. Bauer, and M. Howard, *Computer security : principles and practice (5th edition)*. Pearson, 2023.

[7] Gartner, “Gartner Magic Quadrant for Privileged Access Management,” Sep. 2024. Accessed: Jul. 05, 2025. [Online]. Available: https://www.gartner.com/en/documents/5741683

[8] Netwrix Corporation, “2024 Hybrid Security Trends Report,” May 2024. [Online]. Available: https://www.netwrix.com/2024-hybrid-security-trends-report.html

[9] R. Wang, C. Li, K. Zhang, and B. Tu, “Zero-trust based dynamic access control for cloud computing,” *Cybersecurity*, vol. 8, no. 1, Dec. 2025, doi: 10.1186/s42400-024-00320-x.

[10] V. Hu, D. Ferraiolo, and D. R. Kuhn, “Assessment of Access Control Systems,” 2007. [Online]. Available: https://www.researchgate.net/publication/239546500

[11] M. Penelova, “Access Control Models,” *Cybernetics and Information Technologies*, vol. 21, no. 4, pp. 77–104, Dec. 2021, doi: 10.2478/cait-2021-0044.

[12] A. K. Y. S. Mohamed, D. Auer, D. Hofer, and J. Küng, “A systematic literature review for authorization and access control: definitions, strategies and models,” Oct. 25, 2022, *Emerald Publishing*. doi: 10.1108/IJWIS-04-2022-0077.

[13] A. N. Parahita, I. Eitiveni, D. Nurchahyo, M. Efendi, R. Shafarina, and A. P. Aristio, “Customer Relationship Management System Implementation Process and its Critical Success Factors: A Case Study,” in *2021 International Conference on Advanced*

Computer Science and Information Systems, ICACSIS 2021, Institute of Electrical and Electronics Engineers Inc., Dec. 2021. doi: 10.1109/ICACSIS53237.2021.9631314.

[14] B. Nour, H. Khelifi, R. Hussain, S. Mastorakis, and H. Moungla, "Access Control Mechanisms in Named Data Networks: A Comprehensive Survey," Dec. 2020, [Online]. Available: http://arxiv.org/abs/2012.04624

[15] S. C. Forbacha and M. J. A. Agwu, "Design and Implementation of a Secure Virtual Private Network Over an Open Network (Internet)," *American Journal of Technology*, vol. 2, no. 1, pp. 1–36, Apr. 2023, doi: 10.58425/ajt.v2i1.134.

[16] K. Ragothaman, Y. Wang, B. Rimal, and M. Lawrence, "Access Control for IoT: A Survey of Existing Research, Dynamic Policies and Future Directions," Feb. 01, 2023, *MDPI*. doi: 10.3390/s23041805.

[17] S. Bhatt and R. Sandhu, "ABAC-CC: Attribute-based access control and communication control for internet of things," in *Proceedings of ACM Symposium on Access Control Models and Technologies, SACMAT*, Association for Computing Machinery, Jun. 2020, pp. 203–212. doi: 10.1145/3381991.3395618.

[18] M. U. Aftab *et al.*, "Traditional and Hybrid Access Control Models: A Detailed Survey," *Security and Communication Networks*, vol. 2022, 2022, doi: 10.1155/2022/1560885.

[19] Z. Mao, N. Li, H. Chen, and X. Jiang, "Trojan Horse Resistant Discretionary Access Control," *Association for Computing Machinery*, pp. 237–246, Jun. 2009, doi: 10.1145/1542207.1542244.

[20] G. Zhao and D. W. Chadwick, "On the modeling of Bell-LaPadula security policies using RBAC," in *Proceedings of the Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises, WETICE*, 2008, pp. 257–262. doi: 10.1109/WETICE.2008.34.

[21] D. R. Kuhn, E. J. Coyne, N. Li, J. Byun, and E. Bertino, "Adding Attributes to Role-Based Access Control," 2010. [Online]. Available: http://csrc.nist.gov/groups/SNS/rbac/rbac-standard

[22] Jin Xin, Krishnan Ram, and Sandhu Ravi, "A Unified Attribute-Based Access Control Model Covering DAC, MAC and RBAC," in *Data and Applications Security and Privacy XXVI*, F. and G.-A. J. Cuppens-Boulahia Nora and Cuppens, Ed., Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 41–55.

[23] N. Farhadighalati, L. A. Estrada-Jimenez, S. Nikghadam-Hojjati, and J. Barata, "A Systematic Review of Access Control Models: Background, Existing Research, and Challenges," Jan. 23, 2025, *Institute of Electrical and Electronics Engineers Inc.* doi: 10.1109/ACCESS.2025.3533145.

[24] Government of British Columbia, *Personal Information Protection Act (PIPA)*. Canada: British Columbia Laws (BC Laws), 2025. Accessed: Mar. 11, 2025. [Online]. Available: https://www.bclaws.gov.bc.ca/civix/document/id/complete/statreg/03063_01

[25] Government of British Columbia, *Freedom of Information and Protection of Privacy Act (FIPPA)*. Canada: British Columbia Laws (BC Laws), 2025. Accessed: Mar. 11, 2025. [Online]. Available: https://www.bclaws.gov.bc.ca/civix/document/id/complete/statreg/96165_00

[26] O. Oasis, "OASIS eXtensible Access Control Markup Language (XACML) TC," Jan. 2013. [Online]. Available: https://groups.oasis-open.org/communities/tc-community-home2?CommunityKey=67afe552-0921-49b7-9a85-018dc7d3ef1d#XACML30

[27] L. Patra, U. P. Rao, P. Choksi, and A. Chaurasia, "Controlling Access to eHealth Data using Request Denial Cache in XACML Reference Architecture for ABAC," in *2022 IEEE 3rd Global Conference for Advancement in Technology, GCAT 2022*, Institute of Electrical and Electronics Engineers Inc., 2022. doi: 10.1109/GCAT55367.2022.9971895.

[28] D. F. Ferraiolo, R. Chandramouli, V. C. Hu, and D. R. R. Kuhn, "A Comparison of Attribute Based Access Control (ABAC) Standards for Data Service Applications," Gaithersburg, MD, Oct. 2016. doi: 10.6028/NIST.SP.800-178.

[29] B. Bezawada, K. Haefner, and I. Ray, "Securing home IoT environments with attribute-based access control," in *ABAC 2018 - Proceedings of the 3rd ACM Workshop on Attribute-Based Access Control, Co-located with CODASPY 2018*, Association for Computing Machinery, Inc, Mar. 2018, pp. 43–53. doi: 10.1145/3180457.3180464.

[30] V. C. Hu *et al.*, "Guide to Attribute Based Access Control (ABAC) Definition and Considerations," Gaithersburg, MD, Jan. 2014. doi: 10.6028/NIST.SP.800-162.

[31] S. Ameer, J. Benson, and R. Sandhu, "An Attribute-Based Approach toward a Secured Smart-Home IoT Access Control and a Comparison with a Role-Based Approach," *Information (Switzerland)*, vol. 13, no. 2, Jan. 2022, doi: 10.3390/info13020060.

[32] FirstAttribute AG, "DynamicSync - The cloud solution for dynamic groups in Azure," 2024. [Online]. Available: https://www.dynamicgroup.net/en/dynamicsync/

[33] P. Mell, J. Shook, R. Harang, and S. Gavrila, "Linear Time Algorithms to Restrict Insider Access using Multi-Policy Access Control Systems," *J Wirel Mob Netw Ubiquitous Comput Dependable Appl*, vol. 8, no. 1, pp. 4–25, Apr. 2017, doi: 10.22667/JOWUA.2017.03.31.004.

[34] NIST and U.S. Department of Commerce, "About NIST." Accessed: Mar. 22, 2025. [Online]. Available: https://www.nist.gov/about-nist

[35] R. Kalaria, A. S. M. Kayes, W. Rahayu, E. Pardede, and A. Salehi Shahraki, "Adaptive context-aware access control for IoT environments leveraging fog computing," *Int J Inf Secur*, vol. 23, no. 4, pp. 3089–3107, Jul. 2024, doi: 10.1007/s10207-024-00866-4.

[36] S. Ameer, J. Benson, and R. Sandhu, "Hybrid Approaches (ABAC and RBAC) Toward Secure Access Control in Smart Home IoT," *IEEE Trans Dependable Secure Comput*, vol. 20, no. 5, pp. 4032–4051, Sep. 2023, doi: 10.1109/TDSC.2022.3216297.

[37] A. M. Tall, C. C. Zou, and J. Wang, “Access Control in the Era of Big-Data Driven Models and Simulations.” [Online]. Available: https://www.hhs.gov/hipaa/for-professionals/index.html

[38] D. Ferraiolo, L. Feldman, and G. Witte, “Exploring the next generation of access control methodologies, ITL bulletin for November 2016,” *Information Technology Laboratory, NIST*, USA, pp. 1–5, Nov. 2016.

[39] M. Abdelgawad, I. Ray, S. Alqurashi, V. Venkatesha, and H. Shirazi, “Synthesizing and Analyzing Attribute-Based Access Control Model Generated from Natural Language Policy Statements,” in *Proceedings of ACM Symposium on Access Control Models and Technologies, SACMAT*, May 2023, pp. 91–98. doi: 10.1145/3589608.3593844.

[40] A. Kesarwani and P. M. Khilar, “Development of trust based access control models using fuzzy logic in cloud computing,” *Journal of King Saud University - Computer and Information Sciences*, vol. 34, no. 5, pp. 1958–1967, May 2022, doi: 10.1016/j.jksuci.2019.11.001.

[41] K. Riad, Z. Yan, H. Hu, and G. J. Ahn, “AR-ABAC: A New Attribute Based Access Control Model Supporting Attribute-Rules for Cloud Computing,” in *Proceedings - 2015 IEEE Conference on Collaboration and Internet Computing, CIC 2015*, Institute of Electrical and Electronics Engineers Inc., Mar. 2016, pp. 28–35. doi: 10.1109/CIC.2015.38.

[42] J. John and K. John Singh, “Trust value evaluation of cloud service providers using fuzzy inference based analytical process,” Nature Research, Aug. 2024. doi: 10.1038/s41598-024-69134-8.

[43] M. Soleymani, N. Abapour, E. Taghizadeh, S. Siadat, and R. Karkehabadi, “Fuzzy Rule-Based Trust Management Model for the Security of Cloud Computing,” *Math Probl Eng*, vol. 2021, 2021, doi: 10.1155/2021/6629449.

[44] C. E. Rubio-Medrano, Z. Zhao, and G. J. Ahn, “Riskpol: A risk assessment framework for preventing attribute-forgery attacks to ABAC policies,” in *ABAC 2018 - Proceedings of the 3rd ACM Workshop on Attribute-Based Access Control, Co-located with CODASPY 2018*, Mar. 2018, pp. 54–60. doi: 10.1145/3180457.3180462.

[45] C. Morisset, T. A. C. Willemse, and N. Zannone, “A framework for the extended evaluation of ABAC policies,” *Cybersecurity*, vol. 2, no. 1, Dec. 2019, doi: 10.1186/s42400-019-0024-0.

[46] O. Balogun and D. Takabi, “An Insider Threat Mitigation Framework Using Attribute Based Access Control,” May 2023, doi: 10.48550/arXiv.2305.19477.

[47] Edrian S. Abduhari *et al.*, “Access Control Mechanisms and Their Role in Preventing Unauthorized Data Access: A Comparative Analysis of RBAC, MFA, and Strong Passwords,” *Natural Sciences Engineering and Technology Journal*, vol. 5, no. 1, pp. 418–430, Dec. 2024, doi: 10.37275/nasetjournal.v5i1.62.

[48] P. B. Maoneke, S. Flowerday, and N. Isabirye, “Evaluating the strength of a multilingual passphrase policy,” *Comput Secur*, vol. 92, May 2020, doi: 10.1016/j.cose.2020.101746.

[49] J. A. Shubham and N. Tiwari, “Beyond Passwords: Trends and Innovations in Password less Authentication,” *SSRN Electronic Journal. Elsevier BV*, Mar. 2025, doi: 10.2139/ssrn.5195701.

[50] Guthrie B, Harris N, Data A, and Murphy J, “Magic Quadrant for Access Management,” *Gartner, Inc.*, vol. ID G00805920, Dec. 02, 2024. Accessed: May 23, 2025. [Online]. Available: https://www.gartner.com/doc/reprints?id=1-2JFXRFAU&ct=241125&st=sb

[51] G. Kim, S. Jeong, and K. Kim, “Open Policy Agent based Multilateral Microservice Access Control Policy,” *Korean Institute of Smart Media*, vol. 12, pp. 60–71, Oct. 2023, doi: 10.30693/SMJ.2023.12.9.60.

[52] Auth0, “Customize with Rules.” [Online]. Available: https://auth0.com/docs/customize/rules

[53] ForgeRock, “Scripted Authorization in ForgeRock Access Management.” [Online]. Available: https://backstage.forgerock.com/docs/am/7.3/authorization-guide/scripted-policy-condition.html

[54] Amazon Web Services (AWS), “Controlling access to AWS resources using resource tags.” [Online]. Available: https://docs.aws.amazon.com/IAM/latest/UserGuide/access_tags.html

[55] A. M. Tall and C. C. Zou, “A Framework for Attribute-Based Access Control in Processing Big Data with Multiple Sensitivities,” *Applied Sciences (Switzerland)*, vol. 13, no. 2, Jan. 2023, doi: 10.3390/app13021183.

[56] Team Atlan, “Data Access Governance Tools: What Evaluation Qualities & Criteria Matter in 2025,” May 02, 2025. Accessed: May 03, 2025. [Online]. Available: https://atlan.com/know/data-governance/data-access-governance-tools/#:~:text=and%20vendor%20support.-,What%20are%20some%20of%20the%20most%20common%20pitfalls%20when%20choosing,open%20architecture%2C%20supporting%20endless%20extensibility.

[57] Microsoft, “Best practices for RESTful web API design.” Accessed: Sep. 08, 2025. [Online]. Available: https://learn.microsoft.com/en-us/azure/architecture/best-practices/api-design

[58] “Guide for conducting risk assessments,” Gaithersburg, MD, 2012. doi: 10.6028/NIST.SP.800-30r1.

[59] W. Haque, S. Kulkarni, and A. Arora, “Cloud Provisioning: A Highly Optimized, Cost-effective, and Efficient Deployment Model,” in *CCIOT 2024 - Proceedings of 2024 9th International Conference on Cloud Computing and Internet of Things*, Association for Computing Machinery, Inc, Jan. 2025, pp. 16–23. doi: 10.1145/3704304.3704307.

[60] S. Sahoo, A. Swain, S. K. Mohapatra, S. Pattanaik, and A. Senapati, “Utilizing AWS Cloud-Enhanced VPN and VPC Integration to Improve Data Security and Access Control in Library Management Systems,” in *2nd International Conference on Intelligent Cyber Physical Systems and Internet of Things, ICoICI 2024 - Proceedings*, Institute of Electrical and Electronics Engineers Inc., 2024, pp. 188–194. doi: 10.1109/ICoICI62503.2024.10696648.

[61] J. Zander, I. Schieferdecker, and P. J. Mosterman, *Model-Based Testing for Embedded Systems*, 1st ed. Boca Raton: CRC Press, 2012. doi: 10.1201/b11321.

[62] C. D. N. Damasceno, P. C. Masiero, and A. Simao, “Similarity testing for role-based access control systems,” *Journal of Software Engineering Research and Development*, vol. 6, no. 1, Dec. 2018, doi: 10.1186/s40411-017-0045-x.

[63] S. Daoudagh, F. Lonetti, and E. Marchetti, “Assessing Testing Strategies for Access Control Systems: A Controlled Experiment,” in *International Conference on Information Systems Security and Privacy*, Science and Technology Publications, Lda, 2020, pp. 107–118. doi: 10.5220/0008974201070118.

[64] M. W. Sanders and C. Yue, “Mining least privilege attribute based access control policies,” in *ACM International Conference Proceeding Series*, Dec. 2019, pp. 404–416. doi: 10.1145/3359789.3359805.

[65] Google Developers, “Classification: Accuracy, recall, precision, and related metrics.” Accessed: Jul. 06, 2025. [Online]. Available: https://developers.google.com/machine-learning/crash-course/classification/accuracy-precision-recall

[66] Microsoft, “What’s new in .NET 8.” Accessed: Sep. 08, 2025. [Online]. Available: https://learn.microsoft.com/en-us/dotnet/core/whats-new/dotnet-8/overview

[67] Microsoft, “SQL Server technical documentation.” Accessed: Sep. 08, 2025. [Online]. Available: https://learn.microsoft.com/en-us/sql/sql-server/what-s-new-in-sql-server-2019?view=sql-server-ver15&preserve-view=true

Appendix 1

This section provides a detailed description of an access scenario for each category, including XACML policy rules and NGAC graph structures in the hybrid model and the attributes used with assigned weights. The scenarios were created based on the FSM-based transitions [61], [62], [63], as explained in Chapter 4 (Section 4.2.2).

Category 1: Baseline Valid Access

User1 (ClinicalResearcher) initiates a routine access request to view patient records via the registered app "HealthDataPortal". This request represents a legitimate, typical access aligned with standard authorization rules and historical user behaviour.

- **High-level Attributes evaluated (stable, rarely changed):**
    - Role: ClinicalResearcher *(Essential, Weight=5)*
    - Department: Research *(Essential, Weight=3)*
    - Clearance_Level: 4 *(Essential, Weight=5)*
    - Document_Type: PatientRecords *(Essential, Weight=3)*
    - Classification: Confidential *(Essential, Weight=4)*
- **Hybrid Evaluation (XACML Engine in PDP Policy Rule)**:
    - **Permit if**: all or Essential matches or in range with a weight percentage (e.g. Clearance_Level >= 3)
    - **Else**: Explicitly Deny.

- **Low-Level, Dynamic Attributes (frequently changed) & NGAC Graph Checks:**
    - DeviceCompliance: True *(Essential, Weight=4)*
    - AccessLevel: Read *(Essential, Weight=4)*
    - LoginPatternScore: Normal *(Non-essential, Weight=3)*
    - Location: On-Premises *(Non-essential, Weight=2)*

- **Hybrid Evaluation (NGAC Engine in PDP Policy Rule):**
    - Permit if all or essential attributes with a weight percentage
    - Else: Deny.

- **Final outcome (with Historical Context in NGAC Evaluation):**
    - User1 has consistently accessed "HealthDataPortal" from compliant devices and recognized locations, maintaining stable, anomaly-free access patterns.

Category 2: Adversarial or Malicious Attempts

User2 (EngineeringIntern) attempts to access confidential financial reports through the app "FinanceSecure" by spoofing high-level and contextual attributes. The request is crafted to appear legitimate, but underlying indicators suggest a malicious attempt. Standalone XACML incorrectly permits the request (False Positive) due to spoofed high-level attributes. Hybrid model correctly denies access by validating dynamic context and historical behaviour.

- **High-Level and Stable (rarely changed) Attributes Evaluated:**
    - Role: EngineeringIntern *(Essential, Weight=5, falsely asserted as FinanceAnalyst)*
    - Department: Engineering *(Essential, Weight=3)*
    - Clearance: Level 2 *(Essential, Weight=5)*
    - Document Type: FinancialReports *(Essential, Weight=3)*
    - Classification: Confidential *(Essential, Weight=4)*
- **Hybrid Evaluation (XAML Engine in PDP):**
    - **Permit only if all**:
        - Role is "FinanceAnalyst" or "FinanceManager"
        - Department matches "Finance"
        - Clearance is Level 4 or higher

    - Document Type matches "FinancialReports"
    - Classification is at or below "Confidential"
  - **Else**: Explicitly Deny.

- **Low-Level and Dynamic Attributes (Frequently changed):**
    - DeviceCompliance: False *(Essential, Weight=4; spoofed as True)*
    - AccessLevel: Write *(Essential, Weight=4; spoofed)*
    - LoginPatternScore: Suspicious *(Non-essential, Weight=3)*
    - Location: External VPN *(Non-essential, Weight=2; unusual location)*
- **Hybrid Evaluation (NGAC Engine in PDP):**
  - **Permit only if all**:
    - Device compliance verified as True (actual device state)
    - Appropriate access level matches legitimate user assignments
    - Historical login patterns within normal thresholds (no anomalies)
    - Location consistent with legitimate historical access
  - **Else**: Explicitly Deny (flagging request as adversarial).

- **Final outcome:**

User2 has no history of accessing financial systems and typically accesses only engineering resources. The spoofed high-level attributes may fool XACML, but NGAC detects anomalies, unverified device compliance, elevated access level, and access from an unfamiliar external VPN. The hybrid model correlates these signals and correctly denies the request, identifying it as malicious and unauthorized.

Category 3: Behavioural or Historical Influence

User3 (HRManager) requests access to employee performance records via the registered app **"EmployeeInsights"**. This scenario tests adaptive historical evaluation. Although some contextual attributes slightly deviate (e.g., late-evening access, minor device compliance irregularities), the user's strong historical access pattern and high-weight attribute matches lead to permission through weighted policy evaluation in the hybrid model. (whereas standalone XACML and NGAC result in denial (FN)).

- **High-Level and Stable (rarely changed) Attributes Evaluated**
    - Role: HRManager (Essential, Weight=5)
    - Department: HR (Essential, Weight=3)
    - Clearance: Level 4 (Essential, Weight=5)
    - Document Type: EmployeePerformance (Essential, Weight=3)
    - Classification: Confidential (Essential, Weight=4)
- **Hybrid Evaluation (XAML Engine in PDP):**
    - **Permit if all**: all or Essential matches or in range (e.g. Clearance >= Level 3)
    - **Else**: Explicitly Deny.

- **Low-Level and Dynamic Attributes (Frequently changed):**
    - DeviceCompliance: Partially Compliant *(Essential, Weight=4; minor issue)*
    - AccessLevel: Read *(Essential, Weight=4)*
    - LoginPatternScore: Slightly Abnormal *(Non-essential, Weight=3)*
    - Location: Home Network *(Non-essential, Weight=2; unusual but previously allowed occasionally)*
    - EnvironmentType: AfterHours *(Non-essential, Weight=3)*
- **Hybrid Evaluation (NGAC Engine in PDP):**
    - Calculate total weighted attribute score:
        - Essential attributes strongly satisfied (Role, Clearance, AccessLevel)

    - Minor deviations in non-essential attributes (DeviceCompliance, LoginPatternScore, Location, EnvironmentType)
- Check historical access pattern:
    - User3 historically has strong compliance, routinely accesses similar data with no security violations.
    - Occasional minor deviations are historically permitted without negative outcomes.
- **Permit if**: Historical access baseline is strong and total attribute weight meets required threshold despite minor contextual deviations.
- **Else**: Deny or trigger additional verification.

- **Final outcome:**
    - User3 has repeatedly accessed **"EmployeeInsights"** successfully from a partially compliant device during late-evening hours from their home network. Due to this positive historical context and strong essential attributes (high-weight role and clearance), the NGAC policy permits this request despite minor contextual deviations.

Category 4: Contextual or Temporal Drift

Scenario Description:

User4 (SupportEngineer) requests emergency after-hours access to technical incident reports via the registered app **"IncidentTracker"** from an unfamiliar remote location. This scenario tests policy adaptation to contextual and temporal deviations. Standalone XACML and NGAC deny the request (False Negative). Despite unusual time and location, the request is permitted via

Hybrid model due to the elevated "accessPriority" attribute, flexible international access policy and verified access history, compensating for contextual drift.

XACML (High-Level, Stable Attributes & Policy):

- Attributes evaluated (stable, rarely changed):
    - Role: SupportEngineer *(Essential, Weight=5)*
    - Department: ITSupport *(Non-Essential, Weight=3)*
    - Clearance: Level 3 *(Essential, Weight=5)*
    - Classification: Confidential *(Essential, Weight=4)*
- XACML Policy Rule:
    - **Permit if**: all or Essential matches or in range with a weight percentage
    - **Else**: Explicitly Deny.

NGAC (Low-Level, Dynamic Attributes & Graph Checks):

- Attributes evaluated (dynamic, frequently changed):
    - DeviceCompliance: True *(Essential, Weight=4)*
    - AccessLevel: Read *(Essential, Weight=4)*
    - AccessPriority: Emergency *(Essential, Weight=5)*
    - EnvironmentType: AfterHours *(Non-essential, Weight=3; deviation)*
    - Location: Remote InternationalRemote *(Non-essential, Weight=2; unusual location)*
- NGAC Policy Rule:
    - Evaluate attribute weights and contextual conditions:
        - Device is fully compliant.
        - Emergency access priority is explicitly set (high-weight, essential).
        - After-hours and remote international location represent contextual drift from typical access patterns.
    - Historical baseline evaluation:

        - User4 typically accesses from office locations during regular hours but has previously performed emergency accesses successfully.
    - **Permit if**: Essential attribute "AccessPriority" = Emergency is present, compliant device verified, and historical baseline includes prior successful emergency cases, overriding contextual drift.
    - **Else**: Deny or trigger additional verification.

Historical Context (NGAC Evaluation):

- User4 has successfully conducted emergency access sessions in the past, creating a trusted baseline. Despite unusual current contextual factors (international location, after-hours), NGAC permits access due to the explicitly elevated emergency priority attribute, verified device compliance, and historically proven capability in handling emergency situations responsibly.

Category 5: Policy Conflict and Ambiguity Handling

Scenario Description:

User5 (ProjectConsultant) attempts to access strategic project documentation via the app "StrategyDocs". Two XACML policies conflict, i.e., one permits consultant access to project files, another denies external consultants from accessing sensitive documents. Simultaneously, NGAC applies dynamic contextual restrictions based on location and document sensitivity. XACML and NGAC may produce conflicting or reinforcing Denial results, but the Hybrid model resolves the ambiguity and delivers the correct final decision by evaluating overall risk and context.

Hybrid Evaluation (XACML and NGAC Rules Applied Separately):

XACML Engine in PDP (High-Level, Stable Attributes & Policies):

- Attributes evaluated (stable, rarely changed):
    - Role: ProjectConsultant *(Essential, Weight=5)*
    - Department: ExternalConsulting *(Non-Essential, Weight=3)*
    - Clearance: Level 4 *(Essential, Weight=5)*
    - Document Type: ProjectStrategy *(Essential, Weight=3)*
    - Classification: Sensitive *(Essential, Weight=4)*
- **Policy Rule A (Allowing Consultants):**
    - **Permit if all**: Permit if essential attributes with a weight percentage
- **Policy Rule B (Restricting External Consultants):**
    - **Deny if any**:
        - Department matches "ExternalConsulting" AND
        - Document Classification = "Sensitive"

Policy rule B clearly conflicts with Policy rule A decision in this scenario.

NGAC Engine in PDP (Low-Level, Dynamic Attributes & Graph Checks):

- Attributes evaluated (dynamic, frequently changed):
    - DeviceCompliance: True *(Essential, Weight=4)*
    - AccessLevel: Read *(Essential, Weight=4)*
    - EnvironmentType: RegularHours *(Non-essential, Weight=3)*
    - Location: PartnerNetwork *(Non-essential, Weight=2)*
- **NGAC Policy Rule (Contextual Restrictions):**
    - **Deny if**: Location is "PartnerNetwork" AND classification is "Sensitive" (high-risk context).
    - **Permit if**: DeviceCompliance = True, AccessLevel ≥ Read, and Location = InternalNetwork.

- **Conflict between NGAC and XACML**:

  NGAC denies sensitive access from PartnerNetwork, conflicting with XACML Permit (Policy rule A), but reinforcing XACML Deny (Policy rule B).

Final Policy Decision (XACML and NGAC Resolution in Hybrid model):

- XACML policies conflict:
  - Policy rule A permits access based on Role, Clearance.
  - Policy rule B denies due to ExternalConsulting department and sensitive data.
  - Hybrid considers both policies and weighs essential attributes; although one permits, the restrictive policy rule carries a higher risk weight, resulting in Deny.
- NGAC outcome: Contextual evaluation flags external location + sensitive classification as high-risk, enforcing Deny based on real-time conditions.
- Final outcome: Deny

Category 6: Structural Attribute Violations

Scenario Description:

User6 (MarketingAssociate) attempts to access confidential marketing analytics reports through the registered app "MarketAnalyticsPro". This scenario simulates structural violations, where the role attribute is not updated due to propagation delay (because the user is recently promoted), but the device is compliant, access level is correct (read-only), and other critical attributes match. Standalone XACML or NGAC will deny (causing False Negative), but Hybrid permits with the correct result based on total trust score and attribute weight.

Hybrid Evaluation (XACML and NGAC Rules Applied Separately):

XACML Engine in PDP (Stable High-Level Attributes & Policy Rule):

- **Attributes evaluated**:
    - **Role**: (Missing assignment) (Non-Essential in Hybrid, Weight=3)
    - **Department**: Marketing (Essential, Weight=4)
    - **Clearance**: Level 2 (Essential, Weight=5)
    - **Document Type**: MarketingAnalytics (Essential, Weight=3)
    - **Classification**: Confidential (Non-Essential, Weight=3)

- **Policy Rule**:
    - **Permit if**: Clearance, Department, Document Type (essential) are satisfied and weighted score ≥ threshold.
    - **Deny if**: Any essential attribute above is missing or invalid, and weight threshold not met.

NGAC Engine in PDP (Dynamic, Low-Level Attributes & Trust Rule):

- **Attributes evaluated**:
    - **DeviceCompliance**: True (Essential, Weight=5)
    - **AccessLevel**: Read-only (Essential, Weight=4)
    - **LoginPatternScore**: Normal (Non-essential, Weight=2)
    - **Location**: Internal Network (Non-essential, Weight=2)
    - **Access History**: Positive (H is more than 1)

- **Policy Rule**:
    - **Permit if**:
        - DeviceCompliance is True
        - AccessLevel matches scope (read-only)
        - Historical access to similar resources from the same device and user is positive
        - Total weighted score ≥ required threshold

- **Deny if**: DeviceCompliance is False or AccessLevel is mismatched or trust score is too low

Note: Since the major attributes include DeviceCompliance, Clearance, Department, Classification, and Document Type, the overall attribute weight compensates for the missing role. The positive access history further increases trust. Standalone models may deny due to rigid role checks (False Negative), but Hybrid correctly permits access using holistic evaluation.
