DEBAC: Dynamic Explainable Behavior-Based Access Control

Lucía Cabanillas Rodríguez*†, Juan Manuel Montes-Lopez†, Diego R. López*, Pablo Serrano†
*Telefónica Innovación Digital, Madrid, Spain
†Universidad Carlos III de Madrid, Madrid, Spain

***Abstract*—Traditional access control mechanisms rely on static policies that lack flexibility in adapting to dynamic security threats. In this paper, we present DEBAC: a Dynamic Explainable Behavior-based Access Control architecture, which supports a dynamic assessment of a device trust. Our approach leverages Explainable Boosting Machines (EBMs) to compute a trust score in real-time while providing human-interpretable explanations for access decisions. This dynamic and explainable trust evaluation serves as the cornerstone for defining adaptive access policies, supporting a dynamic response to behavioral deviations. We demonstrate the feasibility of DEBAC with real-life data from a campus WLAN, proving the ability of EBMs to accurately distinguishing between devices while providing a human interpretable explanation for the classification.**

Keywords*—*Access Control, Dynamic Policy, User Behavior, Explainable ML.

I. Introduction

Traditional access control mechanisms, such as Role-Based Access Control (RBAC) and Attribute-Based Access Control (ABAC), are commonly deployed to regulate user permissions in information systems. However, these models rely on static policies that lack the flexibility to adapt to evolving security threats: RBAC assigns permissions based on predefined roles [1], making it unsuitable for adapting to dynamic security environments, while ABAC [2] improves flexibility by incorporating contextual attributes such as location, device type, or time of access, yet it still depends on manually defined rules, which may not account for behavioral variations or real-time threats. This rigidity makes it challenging to detect and respond to security incidents dynamically.

To address these limitations, recent research has explored Machine Learning (ML) based access control models that dynamically adjust access decisions based on user behavior and context. These models move beyond static rule enforcement by continuously evaluating a user's trustworthiness, behavioral anomalies, and environmental context. By doing so, they allow access policies to adapt in real-time to the observed conditions, instead of relying on a predefined set of rules. However, the adoption of ML in security and access control introduces one key challenge, namely, explainability [3]. Unlike traditional models with clear, human-readable policies, ML-based systems generate complex, data-driven decisions, making it difficult for administrators to interpret and justify access outcomes.

To address the above, we propose a solution where explanations for trust evaluation scores are provided to the decision point, along with explanations of the final decision about the access requests, ensuring both transparency and accountability. This allows security teams to analyze why access was granted or denied, and supports policy refinement by making access control decisions more interpretable and adaptable. More specifically, in this paper we propose DEBAC: a Dynamic Explainable Bevaviour-based Access Control architecture, which relies on explainable ML to evaluate device behavior and network activity, generating in real-time a behavioral trust score. In contrast to previous approaches, based on a set of fixed rules, our system adapts access permissions based on how a device behaves, improving security while allowing flexibility. While our implementation leverages Explainable Boosting Machines (EBMs) to assess how closely a device's activity matches its historical profile from the repository, DEBAC is designed to be model-agnostic, allowing for alternative explainable ML techniques to be integrated as needed., which assess how closely a device's activity matches its profile from the Repository. In a nutshell, DEBAC is composed of four main building blocks: data collection (logging network interactions), behavioral profiling (building a history of device activity), trust assessment (computing a trust score), and policy enforcement (adjusting access as needed).

The rest of the paper is organized as follows. Section II provides a detailed description of the proposed framework, presenting the system's architecture and how behavioral insights are integrated into access control decisions. Section III introduces the proof of concept, showcasing the explainability module of the proposed access control model. Section IV reviews existing research in the field, offering a comparative analysis of current approaches and highlighting how they relate to the proposed method. Finally, Section V summarizes the findings and outlines potential directions for future research in dynamic access control systems.

II. Proposed Framework

Here we introduce the framework for context-aware and explainable access control. First, we outline the architecture of the model (II-A), detailing its modules for enforcing access control and providing explainability. Next, we explain how device behavior is compared (II-B) to assess consistency over time. We then describe the use of EBMs to assign trust levels and generate interpretable explanations (II-C). Finally, we demonstrate how trust levels and behavioral insights can support the development of fine-grained access policies (II-D), allowing for adaptive and transparent security decisions.

A. Architecture

The proposed access control architecture is presented in Figure 1. As we explain next, it includes explainable machine

learning to support a dynamic, behavior-driven access control. As mentioned above, this architecture adapts to evolving security conditions by analyzing user behavior in real-time, in contrast to previous models that rely on a set of static rules. The system is composed of the following key elements:

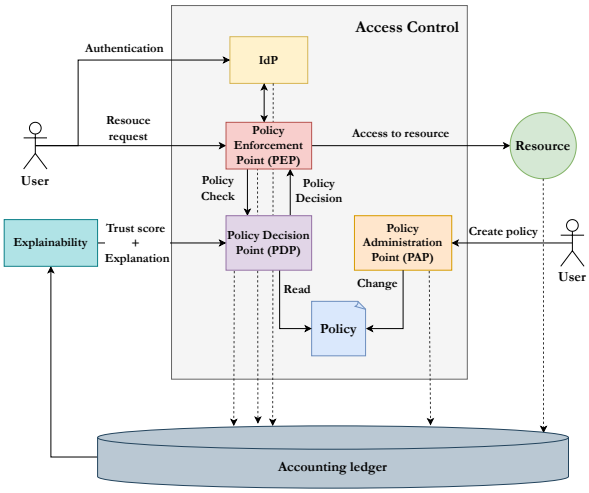


Fig. 1: Architecture

Resources: The protected assets that users seek to access, which may include both physical and virtual elements such as services, applications, or infrastructure components.

Users: These are individuals or entities interacting with the system, either by requesting access to resources or by modifying/creating policies to meet operational needs. In certain contexts, a network entity may serve as either a resource or a user in the decision-making process for access control.

Identity Provider (IdP): A centralized service responsible for identity management, ensuring consistent authentication and identity verification across different systems and services.

Policy Enforcement Point (PEP): The first line of defense in access control, responsible for intercepting user access requests and enforcing security policies. It ensures that only authorized users can proceed with their access requests, based on decisions made by the Policy Decision Point (PDP).

Policy Decision Point (PDP): The core component that makes access control decisions. The PDP evaluates access requests against predefined policies and determines whether access should be granted or denied. Unlike traditional models, the PDP in DEBAC incorporates real-time user behavior patterns into the decision-making process, ensuring that access is granted not only based on static features but also on contextual and behavioral factors.

Policy Administration Point (PAP): This component is responsible for creating, managing, and modifying access policies. The PAP enables dynamic adjustments to policies based on evolving security conditions, contextual changes, or specific requirements from data owners.

Accounting Ledger: This ledger records all the activities, ensuring traceability, auditability, and compliance with security regulations.

Explainability: A key element in behavior-aware access control, this module continuously monitors user activity patterns to assess if their behavior aligns with expectations for the resource they are requesting. By leveraging ML techniques, it detects anomalies and deviations from typical usage patterns, assisting the PDP in making informed, context-aware access decisions.

B. Behavioral profiles for policy Adaptation

We rely on a behavioral Trust Level to evaluate whether a device's current behavior aligns with its historical patterns. To address cold-start scenarios with no prior data, we propose a gradual deployment strategy. Existing logs are analyzed before DEBAC becomes fully operational, progressively establishing behavioral profiles. The Trust Level detects deviations in typical user behavior and enables dynamic policy adjustments by building profiles based on past network activity, generating expected behaviors for comparison. These profiles are tailored to specific scenarios, as demonstrated in the proof of concept in Section III. Access requests are evaluated by comparing observed behaviors with these established profiles.

DEBAC computes two profiles for each device using observed features: a **historical** profile, representing expected behavior based on past activity, and a **real-time** profile, reflecting current activity during the time window. The model compares these profiles to compute a *trust score*, quantifying the alignment between current and historical behavior. This score is derived using both numerical and categorical features.

- For numerical features, e.g., time of connection or session length, the model computes the difference between the observed and the expected values.
- For categorical features, e.g., most used APs or HTTP User-Agent strings, the model computes the intersection size of the historical and the real-time profile.

The key insight behind DEBAC (validated in the proof of concept) is that a properly defined collection of features can uniquely identify a user or a device while supporting explainability. We note that some patterns, like arrival times, are more likely to quickly emerge (e.g., users with a predefined schedule), while others could take longer to emerge and may change over time (e.g., set of preferred locations). To account for this, we continuously monitor and update these features, ensuring that profiles are only used for comparison once they are sufficiently established, and updating them as needed.

C. Policy Decisions and Explanations

To compare the historical and real-time profiles, we rely on Explainable Boosting Machines (EBMs) [4], an interpretable ML model that balances accuracy and transparency. EBMs are based on Generalized Additive Models (GAMs), where predictions are expressed as a sum of feature-specific functions: $g(\mathbb{E}[Y]) = f_1(X_1) + f_2(X_2) + \cdots + f_n(X_n)$. Each feature contributes independently through a learned function, making the decision process interpretable. These functions are trained using gradient boosting, where shallow decision trees iteratively minimize loss, refining each feature function in cycles. Unlike traditional GAMs, EBMs allow limited pairwise

interactions ($f_{i,j}(X_i, X_j)$), incorporating only the most relevant ones. This structure enables EBMs to capture non-linear dependencies while maintaining explainability.

In DEBAC, the EBM takes two input profiles: the historical profile from the repository and the real-time profile from the current session. It then compares them using behavioral features to assess their similarity. The trust score represents the probability that both profiles belong to the same user. A trust score close to one indicates that the device's current behavior aligns with its historical pattern, while a trust score close to zero suggests deviations that may require further security actions.

Using an explainable model is critical for both decision-making and forensic analysis. When a device is assigned a low trust score, the EBM can point specific behavioral discrepancies such as, e.g., an unusual login time, access to previously unused communication ports, or a switch in the list of frequently visited APs. These insights enable administrators to make informed security decisions, such as adjusting policies, triggering re-authentication, or investigating potential security threats.

D. A Dynamic PDP based on behavioral insights

Traditional access control mechanisms rely on static policies, requiring manual intervention to update rules as security conditions change. This approach is slow, error-prone, and ineffective against advanced threats like identity impersonation attacks. In contrast, DEBAC enhances access control by integrating real-time behavioral insights, enabling dynamic and adaptive policy enforcement.

As introduced earlier, our model builds behavioral profiles, compares real-time activity against historical patterns, and generates a Trust Level (ranging from 0 to 1) to quantify how expected a user's behavior is. This score directly influences access decisions, allowing policies to adapt dynamically based on real-time security analysis.

At the core of this approach, the Policy Decision Point (PDP) evaluates access requests in real time by integrating insights from the Explainability module. It allows to refines policies based on behavioral metrics, which vary depending on the specific use case. For example, it may consider timestamp deviations, IP address patterns, or other behavioral indicators to ensure flexible, fine-grained, and context-aware access control.

To achieve this, our framework adopts Policy as Code (PaC) with `Rego`[1], enabling automated, explainable, and auditable policy adaptation. PaC treats access control policies as code, allowing them to be written, stored, and evaluated programmatically. We implement this approach using Rego, a declarative language designed for policy definition and evaluation. Unlike imperative programming, which follows step-by-step procedures, Rego uses logic-based rules to specify the conditions that must be met, ensuring a more adaptable and expressive access control mechanism.

The PDP in DEBAC is designed to:

- Use behavioral insights from the Explainability module to adjust access conditions dynamically.
- Leverage Policy as Code (PaC) [5] to define, modify, and enforce policies in a version-controlled manner.
- Support fine-grained access control, considering features, roles, real-time contextual factors, and deviations from historical behavior.

In Box II.1 we implement a Rego policy example which models an access control mechanism where a user must satisfy certain conditions to gain access to a resource. This access is evaluated dynamically based on real-time information from the Explainability module.

Box II.1: Rego Policy Example

```
1  default allow = false
2  claims := payload if {
3      [_, payload, _] := io.jwt.decode(
           bearer_token)
4  }
5  bearer_token := t if {
6      v = input.request.headers.
           authorization
7      startswith(v, "Bearer ")
8      t = substring(v, count("Bearer ")
           , -1)
9  }
10 is_admin if claims.realm_access.roles
       [_] == "admin"
11 behavior_data := http.send({
12     "method": "GET",
13     "url": "https://example.com/
           behavior",
14     "query": {
15         "user_id": claims.id
16     }
17 }).body
18 behavior_verified if {
19     behavior_score := behavior_data.
           score
20     behavior_score > 0.8
21 }
22 ip_verified if {
23     ip_deviation := behavior_data.
           ip_deviation
24     ip_deviation <= behavior_data.
           allowed_ip_deviation
25 }
26 allow if {
27     input.request.path == "/resource"
28     is_admin
29     behavior_verified
30     ip_verified
31 }
32 status_code := 200 if {
33     allow
34 } else := 403
```

The following is a step-by-step explanation of how the policy operates.

1. Bearer Token Extraction and Validation: First, the policy extracts the Bearer token from the Authorization header in the incoming request (Lines 5–9). The Bearer token is then decoded to extract the claims (Lines 2-4), which contain the

[1]https://www.openpolicyagent.org/docs/latest/policy-language/

user's information. Although the actual verification process is simplified here for illustration, it typically involves checking the token's signature using a public key.

2. Role Authorization: The policy checks if the user's role allows them to access the requested resource. In this case, it checks whether the user has the "admin" role (Line 10), which is a requirement for accessing the resource.

3. Behavior Verification: The policy then sends a request to the Explainability module (a separate service) to retrieve the user's behavior data based on their ID (Lines 11-17). This data includes a behavior score, which reflects the user's past actions and patterns. If the user's behavior score is greater than a predefined threshold (e.g., 0.8, checked in Line 20), the policy considers the behavior acceptable.

4. IP Deviation Check: In addition to behavior verification, the policy checks whether the IP address from which the request is coming is within an acceptable range (Lines 22-25). This check ensures that access is being requested from a familiar or trusted network location.

If all conditions are met (valid token, role authorization, behavior verification, and acceptable IP address), the user is granted access to the resource (Lines 26-31). If any of the conditions fail, access is denied, and an HTTP status code of 403 (Forbidden) is returned (Lines 32-34).

III. Proof of Concept

In this section, we validate the feasibility of DEBAC by implementing a proof of concept of the explainability module and test it using real-life data from a campus WLAN. More specifically, we focus on assessing whether the explainability module, combined with the accounting ledger, is able to accurately interpret and compare different device behaviors, so as to provide a trust level to an observed trace.

We focus on a campus WLANs since they provide a rich environment: they must support both institutional and personal devices, creating security challenges due to high mobility and varying access privileges. Robust access control is therefore crucial to detecting threats such as privilege escalation, unauthorized access, and lateral movement. For example, if a device that typically operates within a known IP range suddenly accesses restricted resources from an unfamiliar IP, this may indicate credential misuse, VPN tunneling, or an attempt to bypass access controls.

To model and assess the trustworthiness of devices within the WLAN, we extract a set of key behavioral features that capture both spatiotemporal patterns and network usage characteristics. These variables serve as the foundation for constructing the profiles:

- **Time of First Connection** (`tarrival`) The time a device first connects each day, helping to establish a temporal usage pattern.
- **Most Frequently Visited APs** (`topAPs`) The top 3 APs a device connects to, capturing its typical movement across campus.
- **HTTP User-Agent** (`http`) The most frequently used HTTP User-Agent string assigned to the device, representing its browsing behavior and operating environment.
- **Assigned IP Addresses** (`IP`) The top 3 internal and external IPs assigned to the device, capturing its typical network presence.
- **Ports Used** (`Port`) The 3 most frequently accessed network ports, indicating the device's typical service interactions.

These behavioral features allow the model to build a detailed profile for the typical activity of each device, which provides a solid baseline for comparison. Rather than directly searching for anomalies, the system focuses on understanding what is normal for each device—its usual connection times, frequently visited access points, common protocols, and interaction patterns. By continuously updating these profiles, the model enables a more nuanced assessment of behavior. To ensure security measures are applied only when necessary, the system continuously compares a device's current activity with its established behavioral pattern. If a significant deviation is detected, the system will either allow or deny access based on the severity of the deviation. If the deviation is deemed suspicious, access will be denied to ensure security.

A. Dataset

The dataset for this study is based on the `RADIUS` accounting logs from authentication service. Each day, around 10,000 users connect to the network via the approx. 300 APs deployed across campus. On average, a user connects to 14 different APs per day, expanding to 30 APs over a week, with this number stabilizing after two weeks. This indicates that a device's mobility pattern becomes well-defined within two weeks. Similar stabilization is observed in other behavioral attributes, such as IP address ranges or time of arrival, confirming that device behavior can be effectively profiled within this time window.

B. Performance Evaluation

To evaluate the model's effectiveness in classifying device behavior and computing a trust level, we rely on the accuracy and the F1 score. The accuracy measures the proportion of correctly classified instances, while the F1 score balances precision and recall, ensuring a well-calibrated distinction between expected and unexpected behavior. Our model achieves an accuracy of 94.3% and an F1 score of 95%, confirming its ability to accurately assess device behavior while minimizing unnecessary alerts.

C. Trust Assessment and Explanations

In this section, we analyze the explanations provided by the model when assigning trust to a device. These explanations can be (1) global, representing the overall importance the EBM assigns to each behavioral feature, computed as the average contribution of each feature across all predictions; or (2) local, indicating the specific factors that influenced the Trust Level for an individual device at a given moment, computed based on the feature contributions for that particular instance.

Figure 2 shows the Global Feature Importance assigned by the model, indicating which variables are used for predictions.

The figure includes both the importance of individual variables and the impact of pairwise feature interactions, as discussed in Section II-C. The results reveal that the most influential factor is the difference in `IP` addresses between compared profiles. This is likely because users typically connect from a consistent set of `IP`s, and when profiles have distinct `IP` sets, the model can quickly determine that they belong to different users. Similarly, `http` metadata plays a key role, as it serves as an identifier of a device's browsing behavior. The model relies heavily on this feature, particularly when there are significant differences or unexpected similarities between profiles.

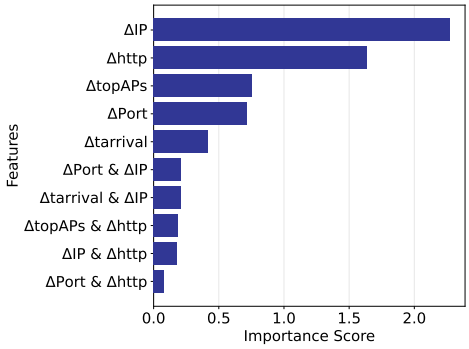


Fig. 2: Global Feature importance of the EBM.

As discussed, the model also provides explanations for specific instances, allowing for a more detailed analysis of its decisions and supporting forensic investigations. Figure 3 illustrates one particular example where the behavior of a device on a given day is compared to its historical behavior stored in the Accounting Ledger. The figure presents the relevant features along with their positive and negative contributions to the Trust Level. Positive scores indicate alignment with the historical profile, while negative scores highlight deviations that may lower trust.

The explanation reveals that the model tends to classify these two profiles as belonging to the same user when there is an overlap in `http` metadata, `IP` addresses, and `topAPs` used. For instance, despite a 6-hour difference in the time of arrival, the model assigns a high Trust Level (0.994) to the device, indicating that its network behavior remains consistent, reinforcing its identity.

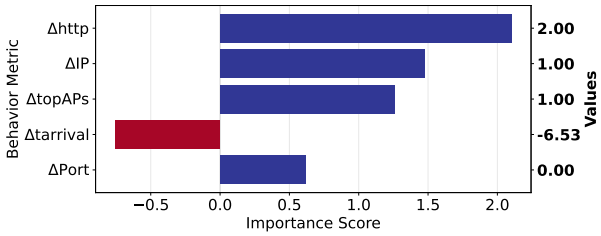


Fig. 3: Explanation provided by the EBM for a specific instance.

D. Dynamic Trust Assessment

To further understand how our model dynamically assesses device behavior, we analyze how a device's trust level evolves over a day. This analysis provides insight into how behavioral deviations influence the model's trust assessment in real-time and highlights the most relevant factors driving changes in trust levels.

Figure 4a and Figure 4b illustrate the trust level assigned to two different devices when compared with their profiles in the Accounting Ledger, evaluated every 10 minutes. The results highlight the dynamic nature of trust assessment, revealing distinct behavioral patterns. During Stable Trust Periods, when a device follows its expected behavior—such as connecting at usual times and using common access points and ports—its Trust Level remains high and stable, ensuring uninterrupted access. However, Gradual trust Shifts occur over the day, where minor deviations, such as roaming across campus and connecting through a different IP address, lead to slight trust adjustments.

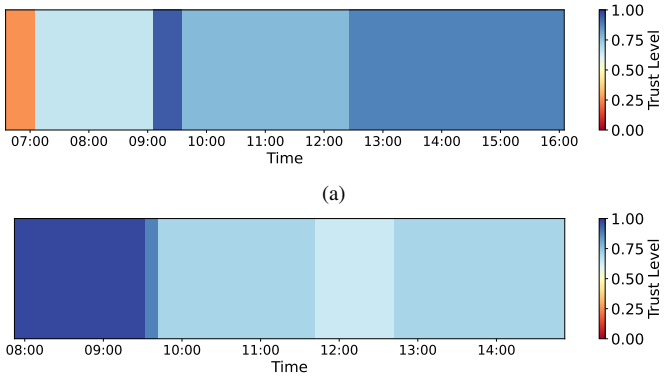


Fig. 4: Trust Level assigned to two devices as a function of the time.

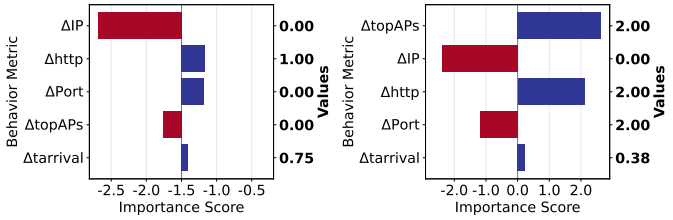
For example, Figure 4a shows a device with a Trust Level that increases over time. To understand what influenced these changes, we analyze the Trust Level explanations provided by the model. As shown in Figure 5a, the initial Trust Level is low because the device connects to unfamiliar APs and uses `IP` addresses it has not previously used. However, after remaining on the network for a few hours (Figure 5b), the Trust Level gradually increases and stabilizes as the device connects to its commonly visited APs and its `http` User-Agent matches its historical profile. Once its behavior aligns with past patterns, the Trust Level remains high and stable, indicating that its activity stays within expected behavioral bounds.

Moreover, in the case shown in Figure 4b, where the device loses trust, the explanations reveal that this loss occurs because the device spends more time connected to previously unvisited APs.

These results illustrate the ability of EBMs to support DEBAC in making dynamic and explainable decisions, ensuring that access control policies can adapt based on real-time behavioral analysis while providing clear justifications for trust evaluations.

IV. Related Work

Machine Learning-Based Access Control. To overcome the limitations of static access control models, researchers



(a) Explanation of the Trust at **7:00**. (b) Explanation of the Trust at **13:00**.

Fig. 5: Explanations of the Trust Level for the trace in Fig 4a.

have introduced trust-based and behavior-aware access control frameworks. One study proposes a hybrid trust model that combines RBAC and ABAC, incorporating behavioral analysis to assess user trust dynamically [6]. This approach monitors access patterns in real time, detecting anomalies in spatial and temporal activity to adjust permissions accordingly. Unlike static policies, this model adapts dynamically to security risks, reducing reliance on manual intervention.

Expanding on this idea, another study introduces Attribute/Behavior-Based Access Control (ABBAC), which extends ABAC by integrating user behavior analysis [7]. ABBAC processes log data to detect deviations in access patterns, allowing it to identify suspicious behavior even from authorized users. By utilizing machine learning techniques such as Random Forest and k-Nearest Neighbors, ABBAC enhances real-time anomaly detection, preventing unauthorized access that might bypass conventional attribute-based policies.

Explainability in Machine Learning for Security. As machine learning models become increasingly integrated into security applications, ensuring explainability is essential for trust and usability. A study on Trust-Based Access Control in Cloud Computing introduces a machine learning-based trust evaluation strategy that predicts trust values using K-Nearest Neighbors, Decision Trees, Logistic Regression, and Naïve Bayes [8]. Their results show improved efficiency and lower error rates compared to static models

Another study highlights the importance of explainability in network traffic classification, demonstrating how feature attribution techniques such as SHAP (Shapley Additive Explanations) improve security monitoring [9]. Similarly, explainer-agnostic frameworks have been introduced to evaluate the quality and stability of explanations across different ML models, ensuring that security-related predictions are not only accurate but also intelligible to administrators [10].

Our work builds on these advancements by integrating explainable ML techniques into access control, ensuring that trust scores and access decisions remain interpretable. By leveraging Explainable Boosting Machines (EBMs), we provide transparent decision-making while maintaining the predictive power of machine learning. Unlike traditional anomaly detection systems, our model focuses on understanding deviations, allowing for fine-grained access control adjustments based on explainable insights.

V. Conclusions and Future Work

In this work we presented a dynamic access control model based on device behavior analysis and explainable machine learning techniques. The proposed architecture allows for continuous evaluation of device behavior in network environments, enabling more adaptive and transparent access decisions. The evaluation conducted in a WLAN environment has demonstrated that the system effectively detects deviations in behavioral patterns and an example of a dynamic access control policy to enhance security. Additionally, the model provides interpretable explanations for its decisions, offering valuable insights for security administrators.

As future work, we plan to implement the presented architecture to evaluate the integration between the Explainability module and the Policy Decision Point. In addition, we intend to explore new use cases, which may require more complex models to adapt to varying security needs. Additionally, we intend to integrate our approach with other threat detection systems to enhance its ability to identify emerging security risks.

Acknowledgment

The research is supported by the iTrust6G project, Grant agreement N 101097083.

References

[1] R. Sandhu, D. Ferraiolo, and R. Kuhn, "The nist model for role-based access control: towards a unified standard," in *RBAC '00*. New York, NY, USA: Association for Computing Machinery, 2000, p. 47–63. [Online]. Available: https://doi.org/10.1145/344287.344301

[2] V. Hu and D. Ferraiolo, "Guide to attribute based access control (abac) definition and considerations - nist special publication 800-162," January 2014, accessed: 2025-01-31. [Online]. Available: https://doi.org/10.6028/NIST.SP.800-162

[3] M. Pawlicki, A. Pawlicka, R. Kozik, and M. Choraś, "Explainability versus security: The unintended consequences of xai in cybersecurity," in *SecTL '24*. New York, NY, USA: Association for Computing Machinery, 2024, p. 1–7. [Online]. Available: https://doi.org/10.1145/3665451.3665527

[4] Y. Lou, R. Caruana, J. Gehrke, and G. Hooker, "Accurate intelligible models with pairwise interactions," in *KDD '13*. New York, NY, USA: Association for Computing Machinery, 2013, p. 623–631. [Online]. Available: https://doi.org/10.1145/2487575.2487579

[5] P. A. Networks, "What is policy-as-code?" [Online]. Available: https://www.paloaltonetworks.com/cyberpedia/what-is-policy-as-code

[6] H. Zhang, G. Wen, D. Liu, and G. Dong, "A dynamic access control method based on user behavior trust evaluation in edge cloud environment," in *AIBDF '23*. New York, NY, USA: Association for Computing Machinery, 2024, p. 92–97. [Online]. Available: https://doi.org/10.1145/3660395.3660412

[7] M. Afshar, S. Samet, and H. Usefi, "Incorporating behavior in attribute based access control model using machine learning," in *2021 IEEE International Systems Conference (SysCon)*, 2021, pp. 1–8.

[8] P. M. Khilar, V. Chaudhari, and R. R. Swain, *Trust-Based Access Control in Cloud Computing Using Machine Learning*. Cham: Springer International Publishing, 2019, pp. 55–79. [Online]. Available: https://doi.org/10.1007/978-3-030-03359-0_3

[9] S. N. Zeleke, A. F. Jember, and M. Bochicchio, "Integrating explainable ai for effective malware detection in encrypted network traffic," *arXiv preprint arXiv:2501.05387*, 2025.

[10] C. E. M. VILLALOBOS, K. D. Costa, B. Modenesi, and A. Koshiyama, "Evaluating explainability in machine learning predictions through explainer-agnostic metrics," 2024. [Online]. Available: https://openreview.net/forum?id=U4hrISfFKs