ingegneria sismica
International Journal of Earthquake Engineering
Anno XLIII - Num. 3 - 2026

Development of a Multi-modal Data-driven access Control rule conflict Detection and self-repairing System

Leqi Xu[1,*]

[1] College of Sciences, Nanjing Agricultural University, Nanjing, Jiangsu Province, 210095 China

SUMMARY: *Aiming at the problems of authorization conflict, condition overlap and priority anomaly when heterogeneous access control policies run across systems, this paper constructs a formal model of multi-modal rules, maps policy text, permission table, access log and environment context into rule feature vectors, and designs algorithms for semantic matching, constraint judgment and risk classification. On this basis, a rule conflict self-repair system is developed, which implements repair strategy generation, candidate version verification, exception rollback and log tracking. The experiment is carried out based on 12000 access control rules, 18 000 access logs and 2860 conflict rule pairs. The results show that the Precision, Recall and F1 of the complete method reach 95.4%, 93.1%and 94.2%respectively, which is 19.9 percentage points higher than that of the field matching method. After automatic repair, the residual conflict rate is reduced to 4.9%, and the average response time is 138.7 ms under 12000 rules and 1000 concurrent requests, which verifies the detection accuracy of the method and the stability of the system.*

KEYWORDS: *Multi-modal data; Access control; Rule conflict detection; Self-repairing system*

1 Introduction

Digital business systems continue to expand in government services, financial transactions, medical data sharing, industrial Internet and Internet of things platforms. Access control has become an important basis for ensuring data security, continuous business operation and trusted system interaction. Traditional access control rules mostly rely on manual configuration, static authorization and periodic audit, which are suitable for a single system, fixed roles and clear permission boundaries. However, in the environment of zero-trust architecture, cross-domain collaboration and multi-terminal access, the access subjects, resource objects, operation behaviors, environmental states and risk levels are constantly changing, the sources of rules are more dispersed, the coexistence of policy models is more obvious, and the risk of rule conflict increases [1, 2].

The conflicts of access control rules are usually manifested as the inconsistency of admission and denial decisions, the intersection of role inheritance relationships, the overlap of resource ranges, the contradiction of time and space conditions, and the lack of policy priorities. With the application of XACML, ABAC, RBAC, smart contract access control and other models in different scenarios, the rule structure has been extended from a simple "subject-resource-action" matching to a composite structure containing attributes, contexts,

*23123225@stu.njau.edu.cn
https://doi.org/10.65102/is20261259 Published: 10 June, 2026

risk scores and execution conditions. Without a unified rule representation and automatic detection mechanism, conflicts are often discovered in the running phase, which is easy to cause unauthorized access, rejection of legitimate requests, interruption of business processes or distortion of audit results [3].

Multi-modal data-driven provides a new technical path for conflict detection of access control rules. Access control system not only accumulates policy text and permission configuration, but also generates identity attributes, behavior logs, device status, network environment, time and location, historical approval and abnormal alarm data. These data can reflect the risk characteristics of access behavior and the effect of rule execution from different angles. Through the unified coding of text rules, structured permission tables, log sequences and environmental contexts, the association between rule semantics, access behaviors and execution conditions can be established, and the conflict detection can be turned from static field comparison to semantic matching and constraint determination [4, 5].

Based on the above problems, this paper focuses on the development of access control rule conflict detection and self-healing system, and constructs a rule processing framework for multimodal data around the situation of dispersed rule sources, inconsistent semantic expression, and dynamic execution conditions. In the research, the access subject, resource object, operation behavior, environment condition, authorization result and rule priority are incorporated into the unified representation, so that access control rules from different sources and formats can be transformed into computable objects. On this basis, combined with rule feature coding, semantic matching and constraint determination method, the allow-deny conflict, condition overlap conflict and priority conflict are identified, and the detection results are output according to the risk degree. For the found conflicts, the system further generates repair schemes, completes rule adjustment, result verification, exception rollback and log tracking, and forms a closed-loop processing flow from conflict discovery to repair verification, which provides support for the stable operation of access control policies in complex business scenarios.

2 Related work

Access control research focuses on the authorization relationship among identity, permissions and resources. The early models are mostly represented by RBAC, which aggregates permissions through roles, and is suitable for business systems with clear organizational hierarchies and stable job boundaries. With the increase of cloud platforms, Internet of things, open banking and cross-domain data sharing scenarios, ABAC, XACML and zero trust architecture have gradually become the research focus. ABAC can incorporate subject attributes, resource attributes, operation attributes and environment attributes into authorization judgment, XACML provides a more standardized policy description and decision execution framework, and zero trust emphasizes continuous verification and risk perception in the access process [6-9]. This kind of research promotes the evolution of access control from static authorization to dynamic authorization, but the problems such as the increase of the number of rules, the complex combination of conditions, and the dispersion of policy sources are more prominent.

For access control policy conflicts, existing researches mainly focus on formal verification, automata modeling, heterogeneous policy mapping and policy consistency checking. XACML policy verification research focuses on rule coverage, decision conflict, unreachable rules and whether the combination algorithm meets expectations. Formal methods can find potential contradictions before policy execution [10]. Dynamic access control research describes the process of permission adjustment with environment changes through state transition or

automata, which provides the basis for temporal constraint and state-dependent conflict detection [11]. Heterogeneous access control policy detection further deals with the expression differences between RBAC, ABAC, ACL and other models, and maps different policies into a unified structure for conflict identification [12, 13]. However, the permission constraints in real systems are often scattered in system texts, approval processes, interface logs, and operation and maintenance configurations, and a single model checking is still difficult to cover complex business scenarios.

In recent years, researches on distributed systems, smart contracts, graph-structured data and natural language policy generation have expanded the application boundaries of access control. Blockchain access control emphasizes the traceability and non-tampering of policy execution process, which is suitable for multi-agent collaboration scenarios such as Internet of Things, medical data and open banking [14, 15]. Graph structured data access control focuses on authorization constraints in nodes, edges and relationship paths, so that complex resource relationships can enter the policy expression process [16]. Research on natural language policy generation attempts to transform security policies, business specifications and management texts into executable access control rules, which provides a new idea for unstructured rule processing [17]. In terms of policy generation and assisted execution, related research has begun to focus on human-machine collaborative execution, system-level ABAC support, anomaly detection and security evaluation, which provides reference for the subsequent optimization of access control rules [18-20].

To facilitate the presentation of the relationship between existing research and the work in this paper, Table 1 summarizes the related research directions.

Table 1: Summary of related research on access control rule conflict detection and self-healing

Research Direction	Representative Studies	Main Content	Implications for This Paper
Zero Trust and Dynamic Access Control	[6-9]	Continuous verification, context awareness, risk scoring, and dynamic authorization	Conflict detection needs to incorporate identity, device, environment, and behavior status
Policy Conflict Detection and Formal Verification	[10-13]	Identifying rule coverage, decision conflicts, and unreachable rules	Supporting rule formalization, constraint judgment, and risk grading
Distributed and Scenario-based Access Control	[14-16]	Authorization research for smart contracts, graph-structured data, medical records, and open banking	Supporting multi-source rule parsing, cross-model expression, and complex resource relationship modeling
Policy Generation and Assisted Execution	[17-20]	Natural language policy generation, human–AI collaborative execution, system-level ABAC support, and anomaly detection	Providing references for repair strategy generation, execution verification, and log tracking

Existing research has formed a good foundation in access control model expression,

policy verification, dynamic authorization and intelligent assistant execution, but there are still some deficiencies in rule governance for complex business systems. The research on standardized policy language has insufficient utilization of system text, log behavior and environmental context, and it is difficult to deal with a large number of semi-structured and unstructured rules in real systems. Conflict detection research focuses on whether there are logical contradictions between rules, and the analysis of conflict risk level, business impact scope and repair order is not sufficient. Although policy generation and assisted execution research have improved the efficiency of rule configuration, the connection between policy generation and repair verification, exception rollback and audit trail is still not tight, and it is difficult to form a complete closed loop.

Around the above shortcomings, the contributions of this paper compared with the existing research are reflected in three aspects. Firstly, at the rule expression level, the restriction of single policy language or static permission table is broken through, and the policy text, permission configuration, identity resource attribute, access log and environment context are incorporated into the rule object construction process, which enhances the unified expression ability of heterogeneous rules. Second, at the level of conflict detection, the conflict judgment is no longer limited to the logical contradiction of allow and deny, but combines semantic proximity, scope intersection, conditional constraints and risk level to improve the ability of implicit conflict identification and high-risk conflict ranking. Thirdly, at the system governance level, the detection results are connected with repair execution, result verification, exception rollback and audit trace, so that the management of access control rules is transformed from off-line inspection to continuous governance. The above contributions are able to complement the deficiencies of existing research on multi-source rule fusion, risk-based detection, and repairing closed-loop.

3 Conflict detection method for multi-modal access control rules

3.1 Formal modeling of multimodal access control rules

Multimodal access control rules are derived from policy text, permission configuration tables, identity attribute libraries, resource labels, access logs, and context-aware data. Rules from different sources have differences in expression, field granularity and update frequency. If you directly enter the conflict detection process, it is easy to have problems such as field missing, semantic inconsistency and unclear condition boundaries. In order to ensure a unified input for subsequent feature coding and constraint determination, it is necessary to abstract access control rules into structured objects including subjects, resources, operations, conditions, authorization results and priorities.

$$R_i=\langle S_i,O_i,A_i,C_i,E_i,P_i\rangle \tag{1}$$

where, Ri denotes the ith access control rule; Si represents the access subject. Oi represents resource objects; Ai represents the action. Ci stands for access condition; Ei represents the authorization result, which takes the value permit or deny. Pi denotes the rule priority. The representation can cover the core authorization elements in rules such as RBAC, ABAC, ACL and XACML.

The access subject is not a single user identity, but an attribute set composed of identity, role, organizational relationship, terminal status and risk level. Its formal expression is as follows.

$$S_i=\{uid_i,role_i,dept_i,device_i,risk_i\} \quad (2)$$

where, uidi represents user identity; rolei indicates the role type. depti indicates the organization or department it belongs to. devicei indicates the trusted state of the terminal. riski represents the current risk level of the subject. Through the set of principal attributes, the authorization differences of the same user in different roles, devices, and risk states can be identified.

Resource objects also need to be described in fine granularity to avoid judging the access scope only by the resource name. The set of resource attributes is defined as follows.

$$O_i=\{rid_i,type_i,label_i,level_i,path_i\} \quad (3)$$

where, Oi represents the attribute set of the resource object corresponding to the ith access control rule; ridi represents the resource number. typei represents the resource type. labeli represents the resource label. level indicates the resource security level. pathi denotes the resource path or interface address. Through the resource object attribute set, it can determine whether there are inclusion, cross, inheritance or sensitivity level conflicts between resources pointed by different rules.

The access conditions mainly reflect the dynamic boundaries where the rules take effect, including context variables such as time, location, network, terminal, and behavior frequency. Its vectorized representation is as follows.

$$C_i=[t_i,l_i,n_i,d_i,f_i] \quad (4)$$

where, Ci represents the access condition vector corresponding to the ith access control rule; ti denotes the access time condition; li denotes the access location condition; ni stands for network environment; di indicates the device state. fi represents the frequency of access or the intensity of behavior. The condition vector can describe the dynamic boundary where the rules take effect, and provide the basis for the subsequent overlapping of spatio-temporal constraints, mutual exclusion of environmental conditions and identification of abnormal access.

In order to ensure that rules from different sources can enter the same computing link, rule fields need to be uniformly defined in the modeling phase, as detailed in Table 2.

Table 2: Multimodal access control rule field definition Table

Field	Meaning	Main Source	Example
Subject	Access subject	Identity system, role table, terminal identifier	User, role, device
Object	Resource object	Resource directory, API list, database	File, API, data table
Action	Operation behavior	Permission configuration, access log	read, write, delete
Condition	Execution condition	Time, location, network, device status	Workday, intranet, trusted terminal
Effect	Authorization result	Policy file, rule base	permit, deny
Priority	Rule priority	Management configuration, policy source	High, medium, low

After the above fields are unified, policy text, structured permission table, access log and environment data can be mapped into the same rule space. In order to intuitively present the mapping relationship between multi-source rules and structured objects, a unified representation model of multi-modal access control rules is constructed, as shown in Figure 1.

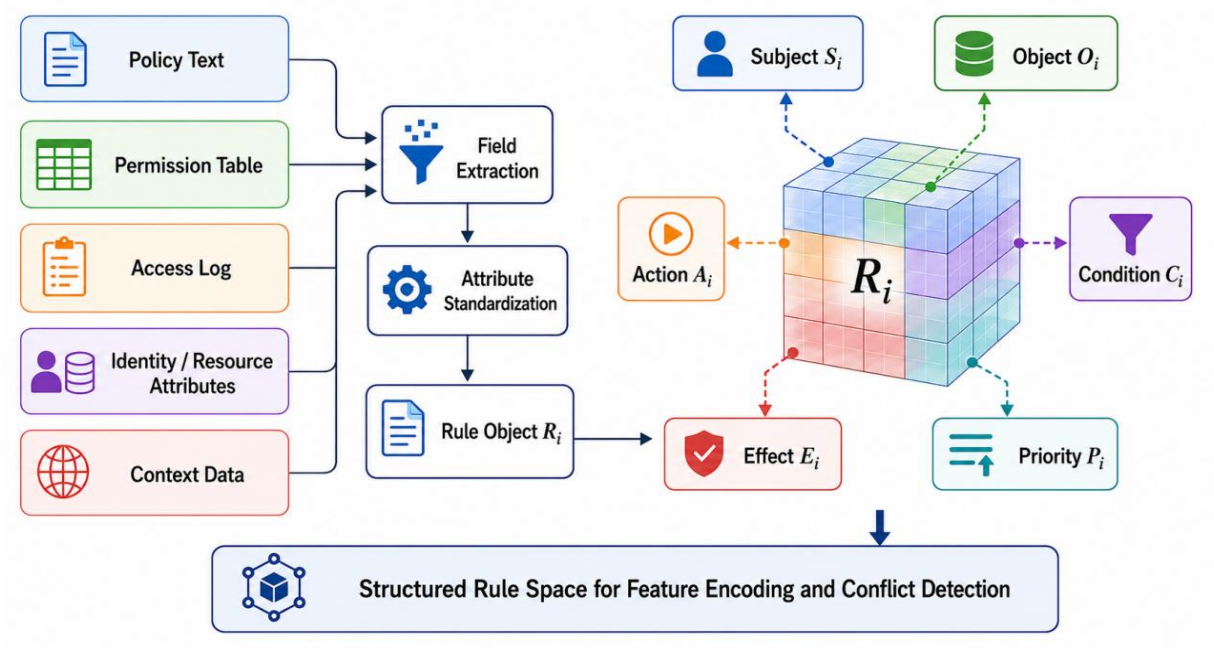

Figure 1: Unified representation model of multimodal access control rules

In this model, policy text, permission configuration, identity and resource attributes, behavior log and environment context are integrated into the unified input. After field extraction and attribute standardization, rule object is formed, which is composed of subject, resource, operation, condition, authorization result and priority. This representation weakens the format differences between different rule sources, and provides a data basis for subsequent feature coding and conflict determination.

3.2 Access control rule feature encoding and semantic matching

After the rules are formally represented, they need to be transformed into feature expressions that can participate in computation. Access control rules contain not only semantic description in policy text, but also structural fields such as subject, resource, operation and condition, and are affected by historical access behavior, device status and network environment. It is easy to ignore the semantic close relationship between "download" and "read and export", "Intranet terminal" and "trusted device" when matching only according to the name of the field. In order to improve the filtering accuracy of conflict candidate rules, text semantics, attribute fields, behavior association and context conditions are uniformly encoded into rule feature vectors:

$$X_i = e_i^T \oplus e_i^A \oplus e_i^B \oplus e_i^C \tag{5}$$

where, Xi represents the comprehensive feature vector of the ith rule; e_i^T represents semantic encoding of policy text; e_i^A represents the encoding of attribute fields such as subject, resource, and operation. e_i^B represents the behavior characteristics extracted from the access log; e_i^C represents the context encoding formed by time, location, device and network environment.

$\oplus$ denotes the vector concatenation or fusion operation.

Text semantic matching is used to identify the close meaning expression and business meaning close relationship in rule description. The text semantic similarity of two rules can be expressed as follows.

$$Sim_T(R_i,R_j)=\frac{e_i^T \cdot e_j^T}{\|e_i^T\|\|e_j^T\|} \tag{6}$$

where, SimT(Ri,Rj) represents the text semantic similarity between rule Ri and Rj; e_i^T and e_j^T denote the text encoding vectors of the two rules, respectively. The closer the value is to 1, the closer the two rules are in semantic expression and more likely to enter the candidate conflict set.

Attribute matching mainly determines whether there is a consistency, inclusion, intersection or equivalence relationship between the subject, resource, operation and condition fields, and its calculation is as follows.

$$M_A(R_i,R_j)=\sum_{k\in K} \omega_k \, m_k(a_{ik},a_{jk}) \tag{7}$$

where, MA(Ri,Rj) represents the attribute matching score; K represents the set of attributes involved in matching; Let ωk denote the KTH attribute weight; aik and ajk denote the values of two rules on attributes of class kkk; mk(·) represents the attribute matching function to distinguish between complete agreement, partial overlap, hierarchical inclusion, and mismatch.

Candidate conflict filtering needs to integrate text semantics, attribute fields, behavior associations and context proximity to form a unified matching score:

$$Score_{ij}=\alpha Sim_T(R_i,R_j)+\beta M_A(R_i,R_j)+\gamma M_B(R_i,R_j)+\delta M_C(R_i,R_j) \tag{8}$$

where, Scoreij represents the comprehensive matching score of rule Ri and Rj; MB(Ri,Rj) represents the behavior association score; MC(Ri,Rj) represents the context proximity score; α, β, γ, δ are the weight coefficients, and satisfies α+β+γ+δ=1. When the comprehensive matching score exceeds the set threshold, the rule pair enters the subsequent conflict constraint determination step.

To illustrate how the rule feature vectors are organized in the unified semantic space and highlight the screening boundaries of candidate conflict rule pairs, a schematic diagram of the access control rule feature embedding space is constructed, as shown in Figure 2.

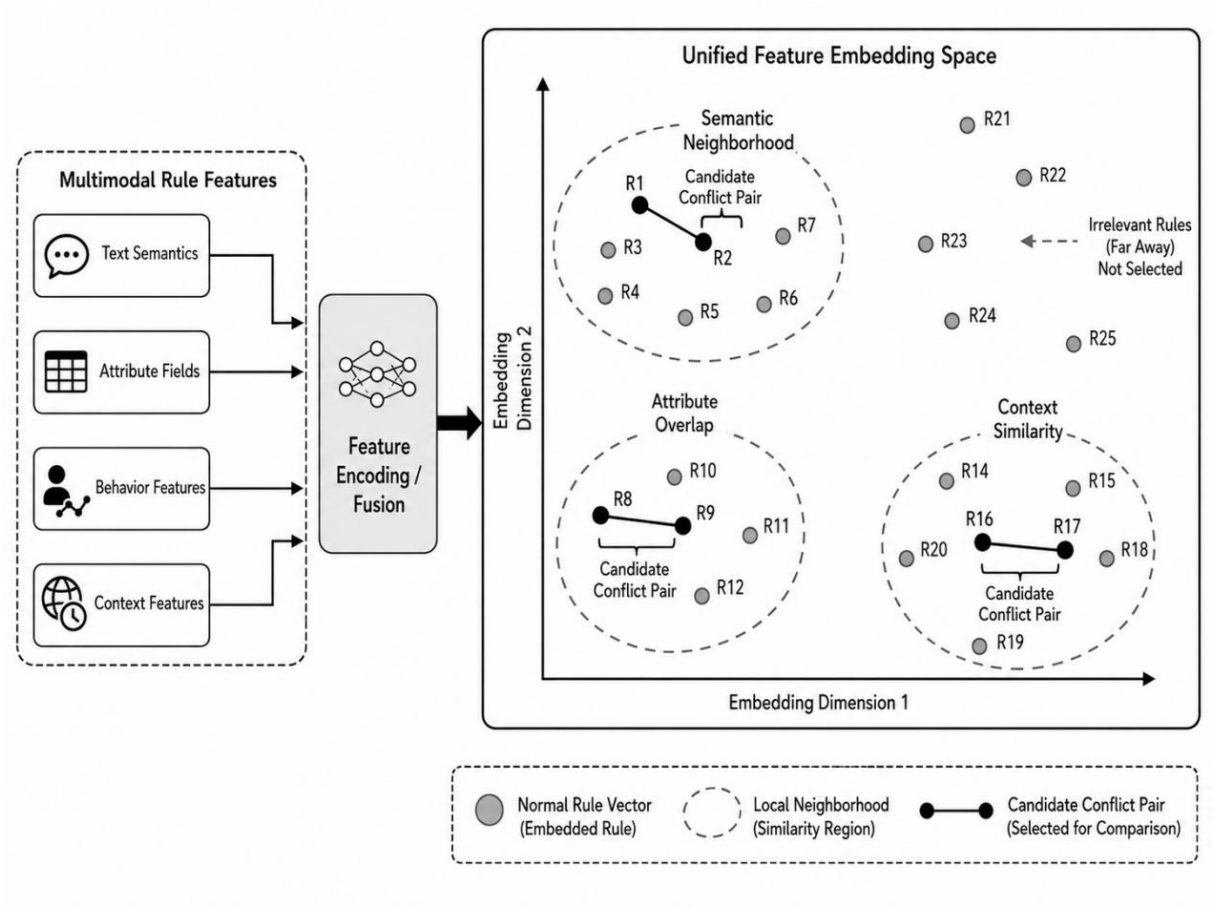

Figure 2: Schematic representation of the feature embedding space for access control rules

The embedding space uses the comprehensive matching score Scoreij as the proximity measure, and maps the rules with similar semantic expression, overlapping attribute fields or similar context conditions to the local neighborhood. For a rule pair satisfying Scoreij≥θs, if there is a need for further determination of its authorization result, resource scope or condition constraint, it will be included in the candidate conflict set. This process can narrow the range of pairwise comparison rules while maintaining the recall ability of potential conflicts, and provide a more centralized input set for subsequent constraint determination.

3.3 Rule conflict type division and constraint determination

The conflict of access control rules needs to be determined on the basis of whether the scope is crossed. Even if the authorization results of two rules are opposite, if the access subject, resource object or operation behavior do not intersect, the actual execution conflict will not be formed. On the contrary, when rules act on similar access scenarios, problems such as different authorization results, overlapping condition boundaries, and fuzzy priorities will affect the final access decision. In order to improve the accuracy of conflict identification, the rule conflicts can be divided into four categories: scope cross conflict, authorization result conflict, condition constraint conflict and priority conflict, and the decision is completed by hierarchical constraints.

Rule-scope crossover is used to measure whether two rules are likely to cover the same class of access requests. Subject, resource and operation are the most basic matching dimensions in an access control rule, and they jointly determine whether the rule has the premise of conflict determination:

$$\Omega_{ij}=w_s J(S_i,S_j)+w_o J(O_i,O_j)+w_a J(A_i,A_j)$$
$$J(X,Y)=\frac{|X\cap Y|}{|X\cup Y|+\varepsilon} \tag{9}$$

where, Ωij represents the scope intersection degree between rule Ri and Rj; Si, Oi and Ai denote the subject set, resource set and operation set of the ith rule, respectively. ws, wo and wa are the corresponding weights; J(X,Y) denotes the set overlap function; Let ε be the smoothing term. The higher the value, the more likely it is that two rules cover the same access scenario.

Authorization result conflict is used to determine whether there is a problem of allowing and denying in similar access scenarios. The determining formula is as follows.

$$D_{ij}=1(\Omega_{ij}\geq\theta_\Omega)1(C_{ij}\geq\theta_c)1(E_i\neq E_j) \tag{10}$$

where, Dij represents the conflict determination value of authorization result between rule Ri and Rj; θΩ represents the scope crossing threshold; Cij is the conditional constraint matching score; Let θc denote the conditional matching threshold; Ei and Ej denote the authorization result of the two rules. If Dij=1, it means that the same access request may be given opposite authorization decisions by two rules.

Conditional constraint conflicts mainly occur when context conditions such as time, location, network, device status and access frequency overlap or are mutually exclusive. The conditional constraint matching score is defined as follows.

$$C_{ij}=\sum_{q\in Q}\lambda_q\,\phi_q(c_{iq},c_{jq})$$
$$\phi_q(a,b)=\begin{cases}1, & a\cap b\neq\emptyset\\ 0, & a\cap b=\emptyset\\ -\rho, & a\perp b\end{cases} \tag{11}$$

where, Cij represents the conditional constraint matching score between rule Ri and Rj; Q={t,l,n,d,f}, corresponding to time, location, network, device state and access frequency respectively; Let λq denote the QTH conditional weight; ciq and cjq denote the values of the two rules on the condition; Let ϕq(·) denote the conditional relation function; a⊥b denotes conditional mutual exclusion; Let ρ denote the mutual exclusion penalty factor.

Priority conflict is used to identify the situation when the conflicting rules cannot be effectively resolved in the order of execution. When the authorization result is in conflict and the priority difference between the two rules is small or there is no clear priority rule, the system needs to mark it as a high concern conflict object:

$$Q_{ij}=1(D_{ij}=1)1(|P_i-P_j|\leq\theta_p) \tag{12}$$

where, Qij represents the priority conflict determination value between rules Ri and Rj; Pi and Pj denote the priority of the two rules; Let θp denote the priority difference threshold. After the continuous determination of scope, authorization result, condition constraint and priority, the candidate rule pair can be further compressed into a conflict set with actual execution risk, which provides input for risk classification detection.

3.4 Conflict detection algorithm based on risk classification

After feature matching and constraint judgment, the system obtains candidate conflict rule pairs. Because different conflicts have different impacts on service security, the detection algorithm can not only output the binary results of "there is a conflict" or "there is no conflict", but also need to combine resource sensitivity level, authorization difference, access frequency, service impact and priority stability to classify the risk. High-risk conflicts need to trigger automatic repair or temporary quarantine. Serious risk conflicts should block rules from taking effect and record audit events.

In order to ensure that the conflict detection results can directly serve the subsequent self-repair processing, the algorithm introduces risk score and level mapping logic on the basis of candidate rule screening and constraint judgment, and forms differentiated processing paths for different conflict rule pairs, as shown in Figure 3.

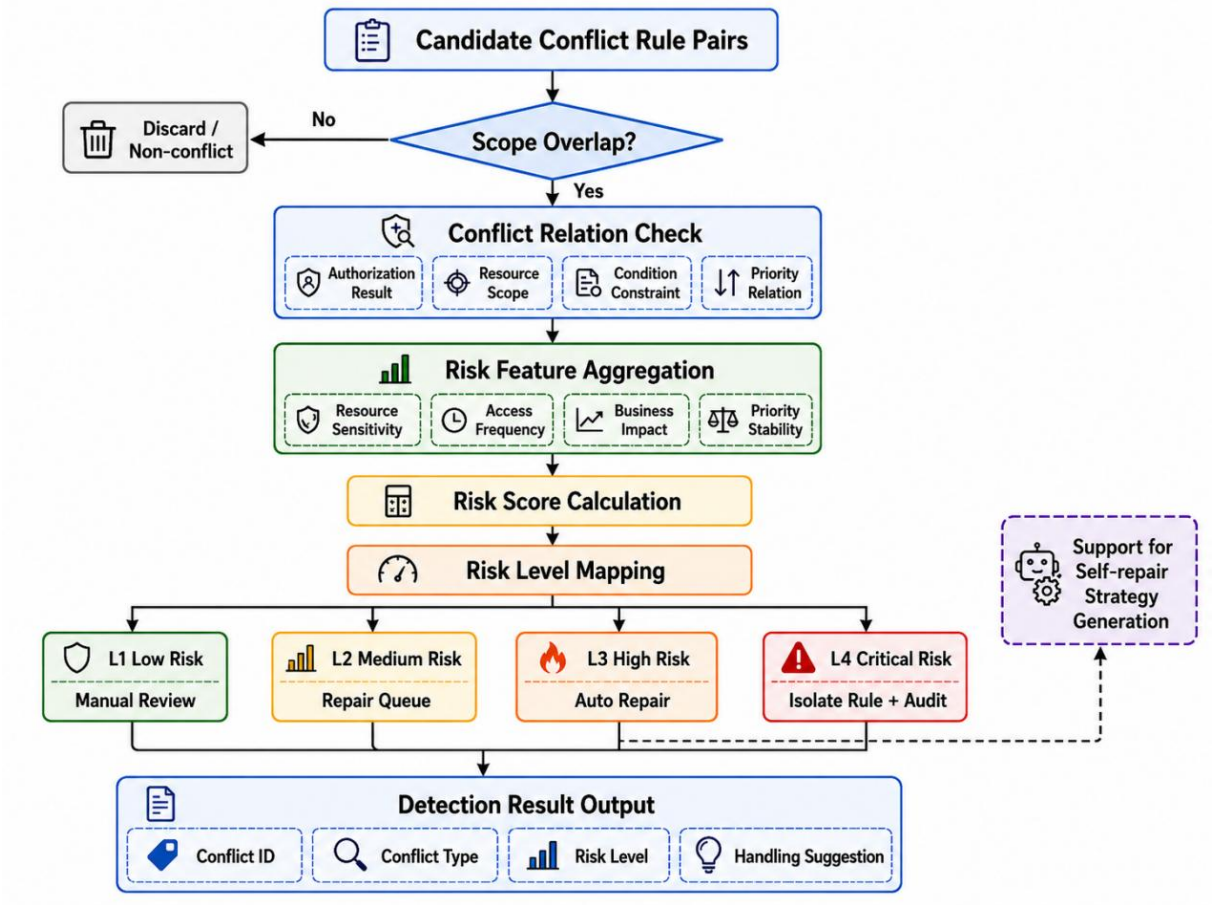

Figure 3: Logic diagram of conflict risk classification decision

The framework takes candidate conflict rule pairs as input, and completes conflict confirmation according to authorization results, scope crossing, conditional constraints and priority relationships. Then, the risk score is calculated by combining resource sensitivity, historical access frequency and business influence degree. After the level mapping, the system outputs the conflict number, conflict type, risk level and treatment suggestion, so that the detection results can be connected with the repair strategy generation link.

The conflict risk score is defined as follows.

$$R_{ij}=\eta_1 D_{ij}+\eta_2 Q_{ij}+\eta_3 S_o+\eta_4 F_{ij}+\eta_5 I_{ij} \tag{13}$$

where, Rij represents the conflict risk score of rules Ri and Rj; Dij represents the conflict determination value of authorization result; Qij represents the priority conflict determination value; So denotes resource sensitivity level; Fij represents the historical access frequency covered by the conflict rule. Iij represents business influence degree; η1 to η5 are the weight coefficients and satisfy the weight sum to one. The scoring puts rule logic conflicts and business impact factors into the same calculation scale, avoiding high-sensitive resource conflicts being overwhelmed by ordinary rule conflicts.

According to the risk score results, the conflict rule pairs are divided into different levels:

$$L_{ij}=\begin{cases} L_1, 0\le R_{ij}<\tau_1 \\ L_2, \tau_1\le R_{ij}<\tau_2 \\ L_3, \tau_2\le R_{ij}<\tau_3 \\ L_4, R_{ij}\ge\tau_3 \end{cases} \tag{14}$$

where, Lij represents the conflict level; L1, L2, L3 and L4 correspond to low risk, medium risk, high risk and severe risk respectively. τ1, τ2, and τ3 are the threshold values for classification. The detection algorithm runs in the sequence of "candidate rule screening -- constraint conflict judgment -- risk score calculation -- level result output", and finally forms a detection result containing conflict rule number, conflict type, risk level and treatment suggestion, which provides input for subsequent self-healing strategy generation.

4 Development of access control rule conflict self-healing system

4.1 Overall system architecture and functional module design

The access control rule conflict self-healing system is located in the policy governance layer outside the existing access control platform, and mainly takes on the tasks of rule access, rule parsing, conflict reception, repair decision, execution verification and log tracking. The system does not directly replace the original permission management module, but establishes a unified processing channel among rule base, permission configuration, access log and environment context, so that the conflict rule pair output in Chapter 3 can continue to enter the repair process. In order to present the data flow relationship between each functional layer of the system, the overall architecture design of the access control rule conflict self-healing system is shown in Figure 4.

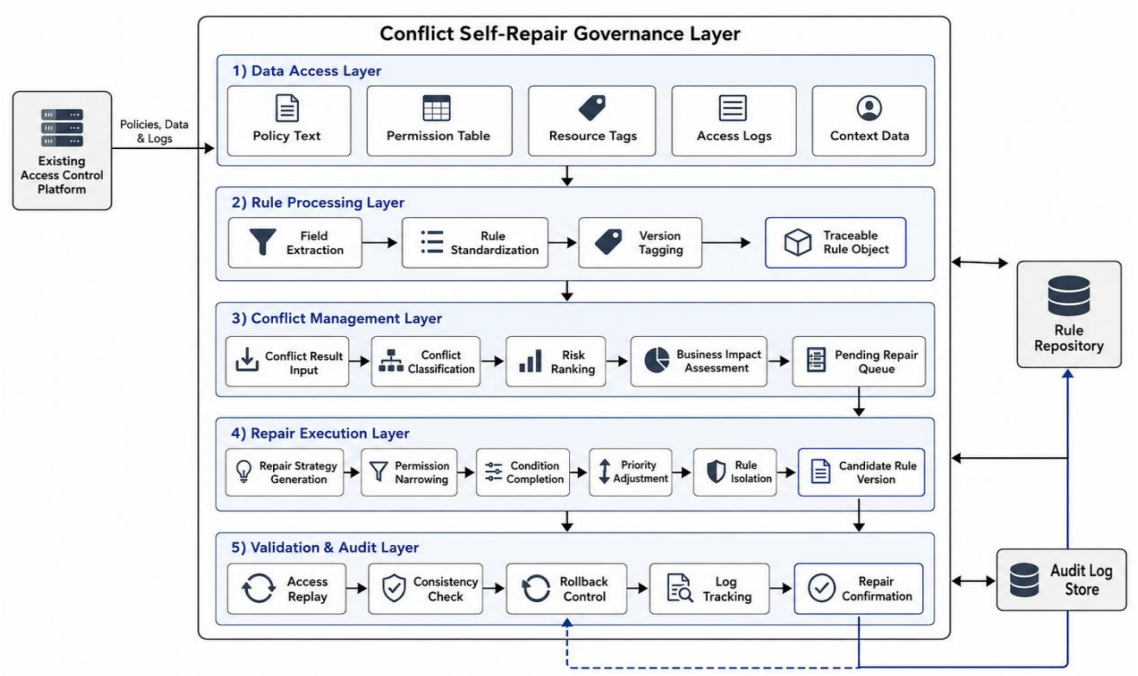

Figure 4: Overall architecture diagram of access control rule conflict self-healing system

The system consists of data access layer, rule processing layer, conflict management layer, repair execution layer and verification and audit layer. The data access level collected and recognized the format of policy text, permission table, resource label, access log and environment context. The rule processing layer is responsible for field extraction, rule standardization and version identification, and provides traceable rule objects for subsequent repair. The conflict management layer receives the conflict results output by the detection algorithm, and forms a queue to be processed according to the conflict type, risk level and business impact. According to the repair strategy, the repair execution layer completes the operations such as permission contraction, condition completion, priority adjustment, and rule isolation. The verification audit layer performs closed-loop confirmation of repair results through access request replay, consistency check, exception rollback and log recording.

During the operation of the system, the detection results will not directly cover the production rules, but first enter the repair sandbox to generate candidate rule versions. The candidate version is written into the rule base after verification, and the original rule is retained and the rollback record is triggered when the verification fails. This design can reduce the disturbance of automatic repair to the business system, and keep a clear boundary between automatic execution, manual review and audit trail.

4.2 Conflict repair strategy generation

The core task of conflict repair policy generation is to select repair actions with smaller impact scope, obvious risk reduction and strong verifiability without destroying the existing business permission boundaries. After receiving the rule pairs output by the conflict detection module, the system first calculates the repair priority according to the risk level, resource sensitivity, access frequency and business impact degree, so as to avoid low-impact conflicts from taking up high-priority processing resources.

$$H_{ij}=\mu_1 L_{ij}+\mu_2 S_o+\mu_3 F_{ij}+\mu_4 I_{ij} \tag{15}$$

where, Hij represents the repair priority score of rules Ri and Rj; Lij represents the conflict risk level; So denotes resource sensitivity. Fij denotes the historical access frequency; Iij represents business influence degree; μ1 to μ4 are the weight coefficients, and the weight sum is 1. The higher the score, the more the conflict needs to be prioritized into the repair queue.

The repair action cannot be done simply by deleting rules or overwriting rules. For the candidate actions such as permission contraction, condition completion, priority adjustment, rule splitting and rule isolation, the modification cost needs to be calculated:

$$Cost_m=\xi_1 U_m+\xi_2 B_m+\xi_3 V_m+\xi_4 A_m \tag{16}$$

where, Costm represents the modification cost of the m-th repair action. Um represents the proportion of affected users. Bm represents the affected business scope; Vm represents the magnitude of rule version change. Am represents the manual review cost; The values ξ1 to ξ4 are the cost weights. This function is used to control the influence boundary of repair actions and reduce the probability of introducing new permission exceptions by automatic repair.

The strategy selection stage makes a trade-off between risk reduction benefit and modification cost, and the repair strategy is defined as follows.

$$M_{ij}^{*}=\arg\max_{m\in M}\ (G_m-Cost_m) \tag{17}$$

where, M_{ij}^{*}denotes the optimal repair strategy of the rule for Ri and Rj; M represents the set of candidate repair actions. Gm represents the risk reduction benefit from the m-th action. According to the result, the system generates the repair scheme, and marks the high-sensitive resources, administrator permissions and cross-domain sharing rules as the objects for manual review.

4.3 Rule self-healing execution mechanism

The execution of rule self-repair unfolds along the main line of "candidate generation - sandbox verification - rule base update - exception rollback". After receiving the repair policy, the system does not overwrite the original rule directly, but locks the version of the conflict rule first, and generates the candidate rules according to the selected policy. After the candidate rules enter the verification environment, whether they meet the authorization boundary requirements is judged by the access request replay and consistency check, and then they are written into the production rule base after verification. If the verification fails, the original rule is retained and the failure reason is recorded to avoid the new permission exception caused by automatic repair.

The rule version update relationship is expressed as follows.

$$R_i^{v+1}=T_m(R_i^{v},\Delta_m) \tag{18}$$

where, R_i^{v} denotes the current version of the ith rule; R_i^{v+1} denotes the fixed candidate version. Tm(·) denotes the m-th repair transformation function; Δm represents the parameter change corresponding to the repair action, such as permission range contraction, condition completion, or priority adjustment.

The basic access boundaries need to be kept intact after the rules are updated, and the consistency constraint is defined as follows.

$$Cons(R^{v+1})=1(B_a\subseteq B_s)1(C_r=0) \tag{19}$$

where, Cons(Rv+1) represents the consistency judgment value of the repaired rule version; Ba represents the actual authorization boundary after repair; Bs denotes the allowable boundary of security policy; Cr represents the number of critical conflicts remaining after repair. Only when the actual authorization boundary does not exceed the security boundary and the critical conflict is eliminated, the candidate rule has the validity condition.

The rule self-repair execution involves the interaction between multiple modules, and the detection results, repair strategies, rule versions, verification results and log records need to be connected continuously, and the closed-loop timing relationship is shown in Figure 5.

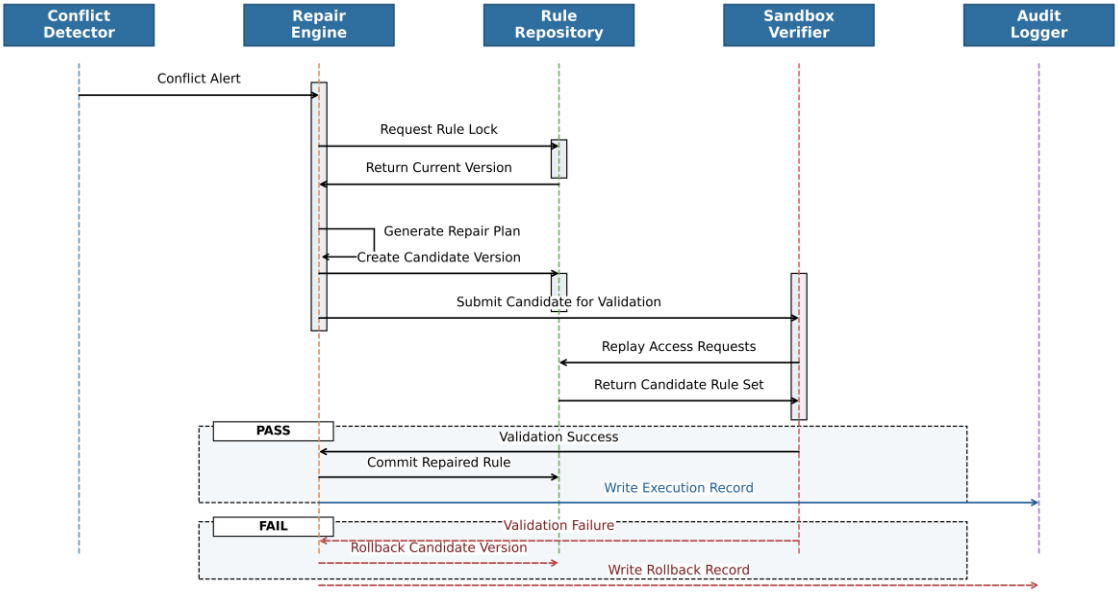

Figure 5: Self-healing closed-loop timing diagram of access control rules

This temporal relation is used to render the order of calls between the detector, the fixer, the rule base, the verifier, and the log base. When the conflict rule pair enters the fixer, the candidate version is generated, the rule base completes the version lock, the verifier performs the access replay and consistency check, and the log library records the policy source, execution status and rollback information.

Successful repair execution is determined as follows.

$$Z_{ij}=1(\mathrm{Cons}(R^{v+1})=1)1(V_{ij}\geq\theta_v) \tag{20}$$

where, Zij represents the repair execution result of the rule on Ri and Rj; Vij denotes the access replay verification score; Let θv denote the verification pass threshold. If Zij=1, candidate rules are written into the rule base and audit records are generated. If Zij=0, the system performs a version rollback and marks this conflict as a human review object.

4.4 Fixing result validation, rollback, and log tracking

After the execution of the repair strategy, the system needs to verify the results of the candidate rules, and the validity of the candidate rules cannot be judged only by whether the repair action is completed. The verification process takes the replay of access requests as the core, and the history normal requests, abnormal requests, boundary requests and high-sensitive resource requests are formed into a test set, and the repaired candidate rule versions are input respectively to compare whether the actual authorization results are

consistent with the expected authorization results. For rules involving administrator permissions, cross-domain shared resources and high-security data, it is also necessary to improve the validation weight to avoid a large number of low-risk requests passing and masking anomalies in critical access scenarios.

The access replay verification score is defined as follows.

$$V_{ij}=\frac{\sum_{r\in T_{ij}} \omega_r \, 1(\hat{E}_r=E_r^*)}{\sum_{r\in T_{ij}} \omega_r +\varepsilon} \tag{21}$$

where, Vij represents the access replay verification score of the rule after repairing Ri and Rj; Tij denotes the set of playback requests associated with this conflicting rule pair; r represents a single access request in the set. Let ωr denote the request weight; $\hat{E}_r$ represents the actual authorization result given by the fixed rule; E_r^* denotes the expected authorization result; $1(\cdot)$ denotes the indicator function; Let ε be the smoothing term. The higher this score is, the closer the fixed rule is to the expected access control state.

When the verification fails, the system does not write the candidate rules into the production rule base, but reverts to the pre-repair state according to the version identification. The rollback process should retain the original rule version, candidate version, failure reason, trigger request and execution time, so as to facilitate subsequent manual review. For the candidate rules that pass the verification, the system still needs to detect whether there is a residual conflict in the rule base, so as to avoid the local repair to eliminate a conflict, but introduce a new condition conflict or priority exception.

The residual conflict rate after repair is defined as follows.

$$\Gamma_{ij}=\frac{N_{ij}^{res}+\lambda N_{ij}^{fail}}{N_{ij}^{ori}+\varepsilon} \tag{22}$$

where, Γij represents the residual rate of conflict after rule repairing Ri and Rj. N_{ij}^{ori} represents the original number of conflicts detected before repair; N_{ij}^{res} indicates the number of conflicts that remain after the fix; N_{ij}^{fail} indicates the number of fixes that failed validation or triggered a rollback. Let λ denote the failure penalty coefficient. ε is used to avoid the denominator being zero. The lower the residual rate is, the more stable the repair result is. If the value exceeds the set threshold, the system will stop the automatic repair and turn to manual review.

The log trace runs through the whole process of verification and rollback, recording rule number, conflict type, repair action, candidate version, verification score, residual rate and operator information. The complete log chain can not only be used for security audit, but also reverse the subsequent repair policy, so that the system can gradually accumulate reusable conflict handling experience in continuous operation.

5 System experiment and performance evaluation

5.1 Experimental data set construction and experimental environment

The experimental data set is built around a hybrid access control scenario, and the rule sources include RBAC role permission tables, ABAC attribute rules, XACML policy fragments, interface access control lists, and manually curated policy text rules. In order to be close to the heterogeneous state of policy in real business, the rule object covers the fields of

user, role, department, terminal, resource label, interface path, operation behavior, time condition, network environment and authorization result. The access log selects user request, interface call, resource access, exception rejection and approval records and other data. After desensitization, it is used for behavior association and access playback verification. The conflict samples are marked by both automatic rule scanning and manual review, and the types include authorization result conflict, resource scope conflict, condition constraint conflict, priority conflict and inheritance relationship conflict.

The dataset contains a total of 12000 access control rules, 185000 access logs, 4200 resource objects and 3600 subject objects. After filtering and labeling rule pairs, 2860 groups of conflict rule pairs are obtained, including 812 groups of authorization result conflicts, 646 groups of resource scope conflicts, 584 groups of condition constraints conflicts, 438 groups of priority conflicts and 380 groups of inheritance conflicts. The data were divided into parameter optimization set, validation set and test set according to 7∶1∶2. The parameter optimization set is used to set the semantic matching weight, scope crossing threshold and risk classification threshold, the validation set is used to adjust the candidate conflict screening boundary, and the test set is only used for the final performance evaluation to avoid bias caused by parameter selection.

To ensure the reproducibility of the experimental process, the operating environment, data scale and main parameter Settings are shown in Table 3.

Table 3: Experimental data set and system environment configuration table

Item Category	Experimental Configuration	Description
Rule scale	12,000 access control rules	Covers RBAC, ABAC, XACML, ACL, and institutional text rules
Access logs	185,000 records	Includes user requests, API calls, resource access, abnormal rejection, and approval records
Resource objects	4,200 objects	Includes data tables, APIs, file directories, and business resource labels
Subject objects	3,600 objects	Includes users, roles, departments, terminals, and device identities
Conflict rule pairs	2,860 groups	Jointly labeled by automatic scanning and manual review
Conflict types	5 types	Authorization result, resource scope, condition constraint, priority, and inheritance relationship conflicts
Data split	70% / 10% / 20%	Used for parameter optimization, validation, and testing respectively
Semantic matching threshold	0.72	Used to screen candidate conflict rule pairs
Scope intersection threshold	0.60	Used to determine whether subjects, resources, and operations have effective intersections
Condition matching threshold	0.55	Used to determine whether time, location, network, and device conditions are close
Risk level thresholds	0.35, 0.60, 0.80	Used to classify low, medium, high, and critical risks
Operating system	Ubuntu 22.04 LTS	Used as the experimental running platform
Processor and memory	Intel Xeon Silver 4214R, 64 GB	Supports rule matching, conflict detection, and concurrency testing
Database environment	MySQL 8.0, Redis 7.0	Used for rule storage and cache acceleration respectively
Development environment	Python 3.10, Java 17	Used for algorithm implementation and system service development
Evaluation metrics	Precision, Recall, F1, false positive rate, false negative rate, detection time, repair success rate, residual rate	Covers detection performance, repair effectiveness, and system stability

5.2 Conflict detection performance evaluation

The performance evaluation of conflict detection is carried out using 572 labeled conflict rule pairs and 1800 non-conflict rule pairs in the test set. The comparison objects include field matching method, single XACML static detection method and heterogeneous strategy mapping method. The ablation experiment is set to remove semantic matching module, remove context constraint module and remove risk classification module to observe the influence of key components on detection results. The evaluation indicators are Precision, Recall, F1, false detection rate, miss detection rate and single batch detection time. Precision is used to measure the false alarm control ability, Recall is used to measure the conflict recall ability, F1 reflects the comprehensive detection level, and single batch detection time is used to evaluate the running cost of the algorithm. For a unified comparison of detection accuracy, recall capability, false detection control, and computational overhead, the baseline method, the ablation version, and the full method were evaluated under the same test set, and the results are shown in Table 4.

Table 4: Conflict detection performance comparison table of different methods

Method	Precision/%	Recall/%	F1/%	False Positive Rate/%	False Negative Rate/%	Single-batch Detection Time/ms
Field matching method	78.6	70.4	74.3	8.9	29.6	41.8
Single XACML static detection method	84.2	76.8	80.3	6.7	23.2	56.5
Heterogeneous policy mapping method	87.5	82.1	84.7	5.4	17.9	64.2
Without semantic matching module	90.8	85.2	87.9	4.6	14.8	70.1
Without context constraint module	92.1	86.7	89.3	4.1	13.3	73.4
Without risk grading module	93.5	91.0	92.2	3.4	9.0	76.6
Complete method	95.4	93.1	94.2	2.7	6.9	83.2

The complete method achieves high results in Precision, Recall and F1, and F1 reaches 94.2%. Compared with the field matching method, F1 is increased by 19.9 percentage points, and the missed detection rate is reduced from 29.6% to 6.9%, indicating that it is difficult to cover the implicit conflicts between system text, interface permissions and attribute rules by relying solely on field consistency. Compared with the heterogeneous policy mapping method, the F1 of the complete method is increased by 9.5 percentage points. The main reasons are that semantic matching can identify rules with different expressions but similar authorization meanings, and context constraints can supplement dynamic conditions such as time, device, and network environment. Ablation results show that Recall decreases by 7.9 percentage points after removing the semantic matching module, and F1 decreases by 4.9 percentage points after removing the context constraint module, indicating that these two modules have obvious effects on improving the ability of implicit conflict recognition. After removing the risk grading module, the overall F1 decreases slightly, but the ranking stability of high-risk

conflicts becomes weaker.

To further present the differences of different methods in core detection indicators, the results of Precision, Recall and F1 are compared visually, as shown in Figure 6.

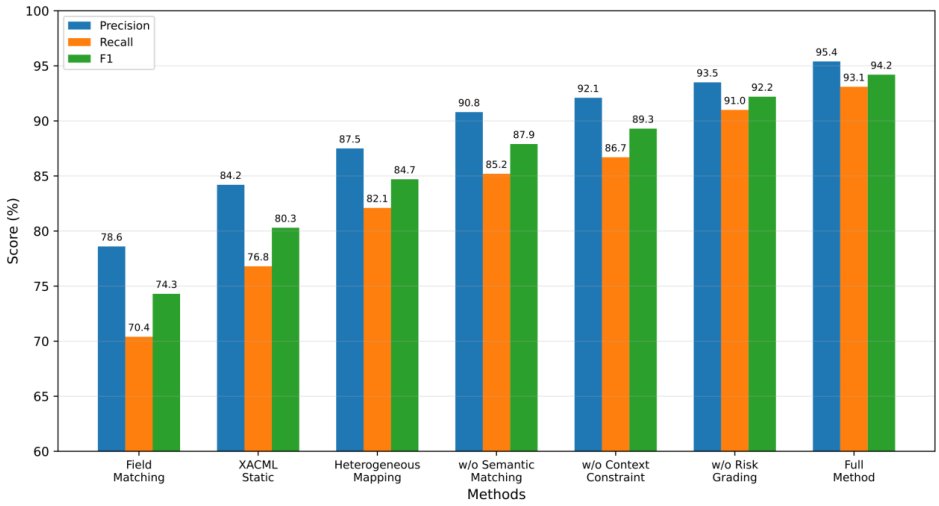

Figure 6: Comparison of conflict detection indicators of different methods

From the changes of the three indicators, it can be seen that the Precision, Recall and F1 of the complete method remain above 93%, while the Recall of the field matching method is only 70.4%, indicating that its ability to recognize non-standard fields, synonymy operations and conditional implicit relations is insufficient. The metrics of the ablation version are between those of the baseline method and the full method, indicating that semantic matching, contextual constraints, and risk grading are not isolated decorative modules, but jointly participate in the conflict identification process.

The rule scale expansion experiment increases step by step according to 1000, 3000, 5000, 8000 and 12000 rules, and records the full time-consuming of candidate screening and conflict determination, which is used to evaluate the computational stability of the method in the case of rule base expansion. The results are shown in Figure 7.

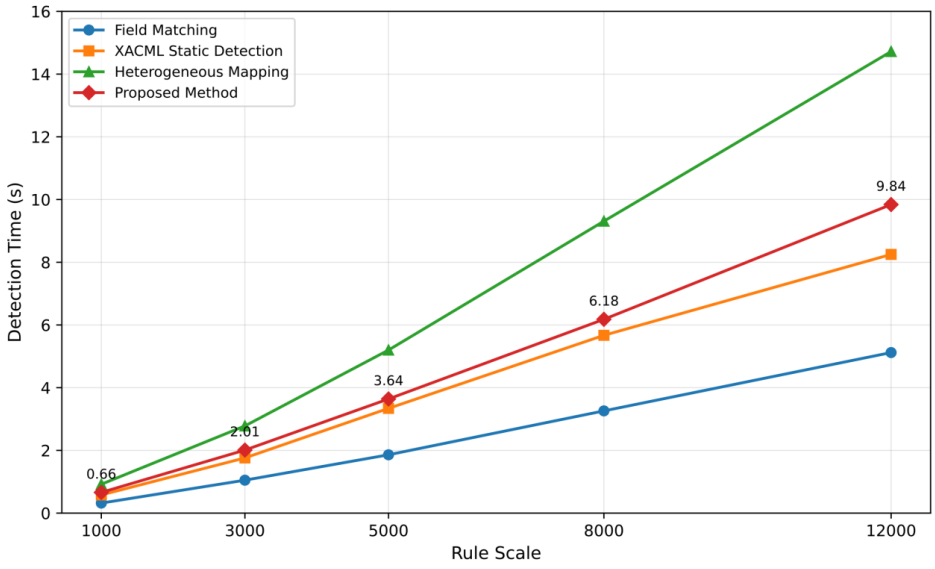

Figure 7: Change diagram of detection time under different rule scales

The detection time increases with the scale of rules. The complete method takes 9.84 s to complete the detection under 12000 rules, which is higher than the field matching method, but lower than the heterogeneous policy mapping method without candidate screening. This result shows that multi-modal feature coding will increase the front-end computational overhead, but the candidate rule screening can significantly compress the subsequent constraint determination range and keep the overall time consumption in an acceptable interval.

To observe the identification differences of different conflict types, the classification results of authorization result conflict, resource scope conflict, condition constraint conflict, priority conflict and inheritance relationship conflict in the test set are counted to form a confusion matrix, as shown in Figure 8.

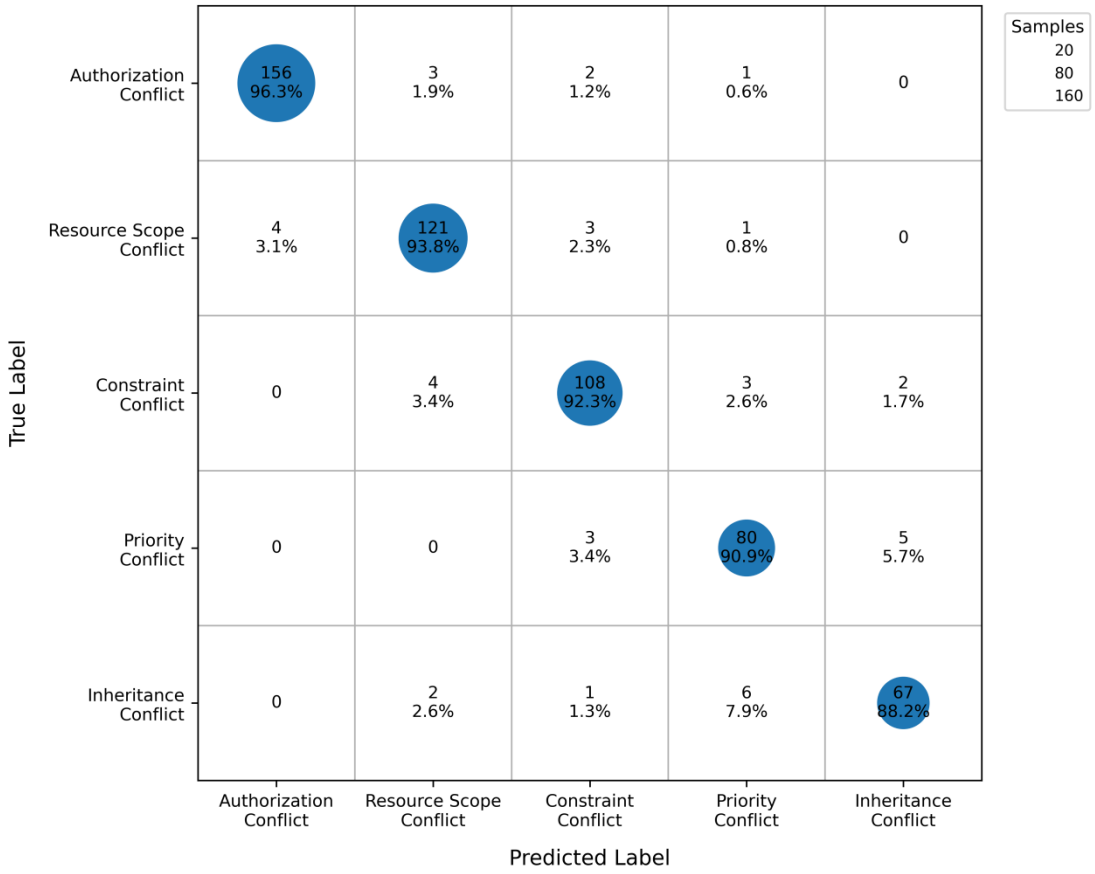

Figure 8: Confusion matrix diagram for conflict detection

The highest conflict recognition rate is 96.1% for authorization results, because the difference between permit and deny has strong dominant characteristics. The recognition rates of resource scope conflict and condition constraint conflict were 93.4% and 92.6%, respectively, indicating that the ability to identify implicit conflicts was enhanced after the inclusion of resource labels, path scope and context conditions. There is a small amount of confusion between priority conflicts and inheritance relationship conflicts, mainly due to the fact that authorization order and priority changes in the role inheritance chain often occur synchronously. The overall results show that the conflict detection algorithm can not only improve the detection accuracy, but also distinguish different conflict types more stably, which provides reliable input for subsequent repair strategy matching.

5.3 Evaluation of self-repairing effect

The evaluation of self-repair effect took 572 groups of conflict rule pairs in the test set as objects, including 162 groups of authorization result conflicts, 129 groups of resource scope conflicts, 117 groups of condition constraint conflicts, 88 groups of priority conflicts, and 76 groups of inheritance relationship conflicts. The system is executed according to the process of "repair policy generation-candidate version writing sandbox-access request replay-consistency verification-result submission or rollback". The criterion of successful repair is that the candidate rules pass replay verification and no new high-risk conflicts are introduced. To compare the repair adaptation ability under different conflict types, the comprehensive repair score is formed after normalizing the repair success rate, average repair time, manual review rate and residual conflict rate, as shown in Figure 9.

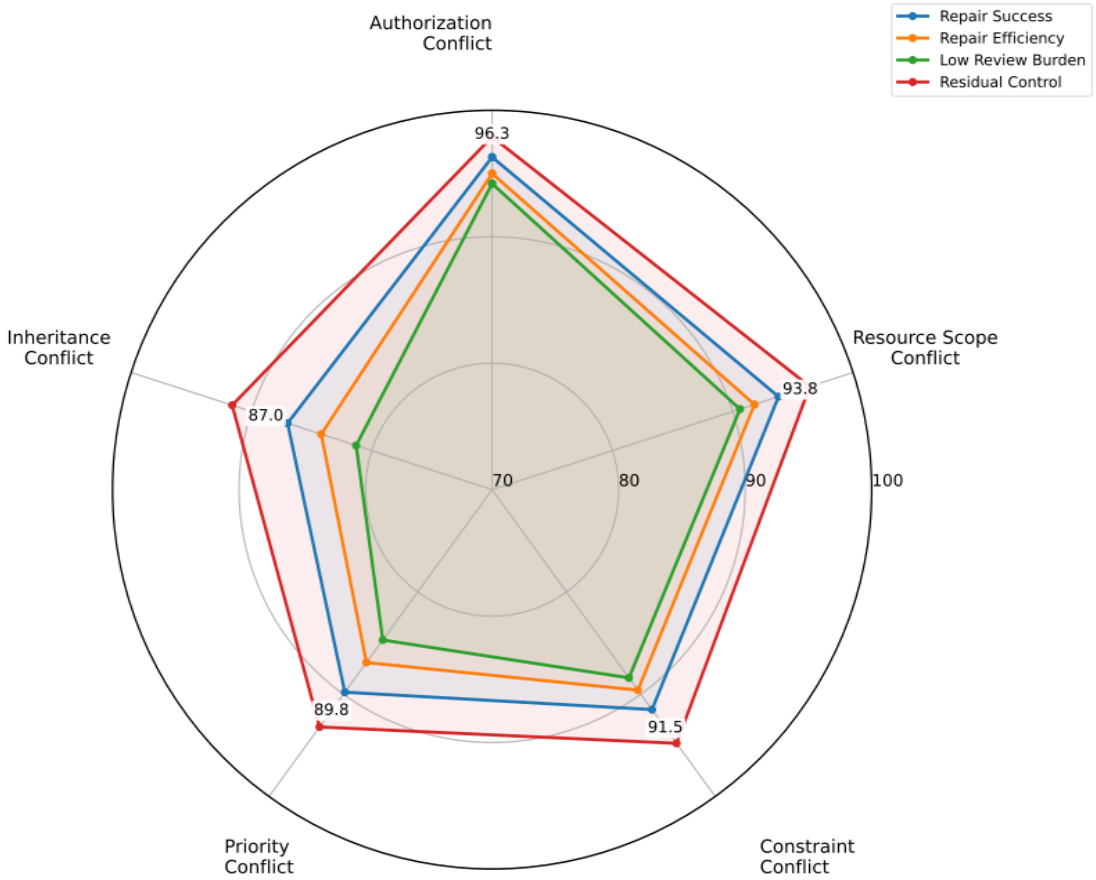

Figure 9: Radar chart of repair effect for different conflict types

The success rate is 96.3%, and the average repair time is 118 ms. The main reason is that the conflict boundary between permit and deny is clear, and the system can deal with the conflict through priority adjustment or permission contraction. The success rate of resource scope conflict is 93.8%, and the success rate of condition constraint conflict is 91.5%, which indicates that resource label refinement, path splitting and time condition completion can better eliminate hidden conflicts. It is relatively difficult to repair priority conflicts and inheritance conflicts, the success rate is 89.8% and 87.0% respectively, and the manual review rate is also higher. The reason is that there is often a coupling relationship between role inheritance chain and rule execution order, and automatic repair needs to avoid affecting the superior role or administrator permissions.

In order to further observe the residual situation of conflicts before and after repair, the residual rate of five types of conflicts is counted, and the results are shown in Figure 10.

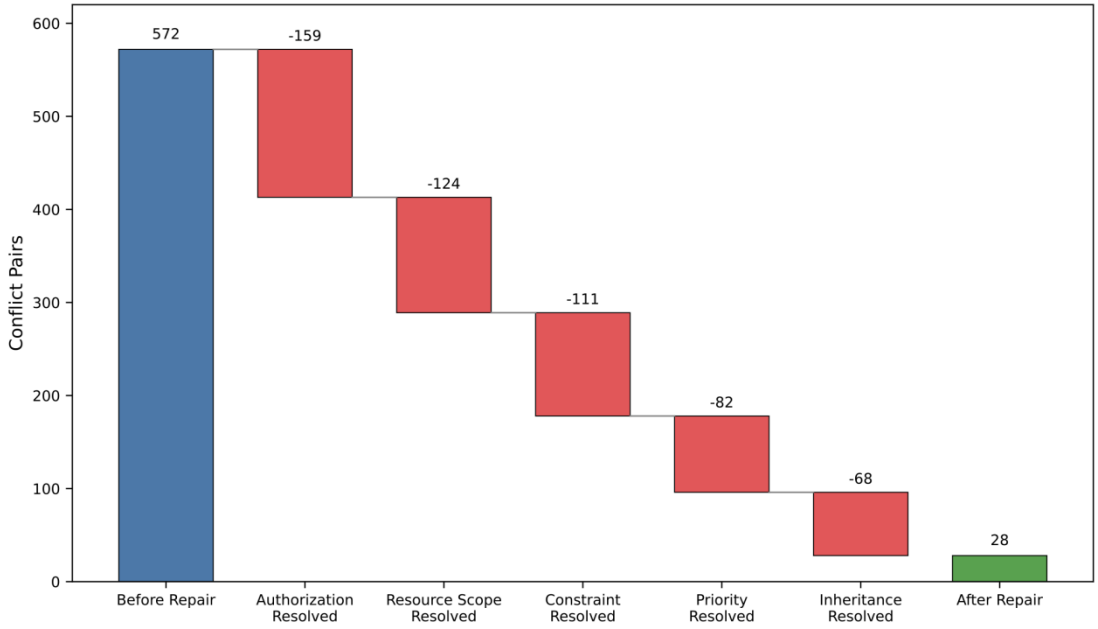

Figure 10: Comparison of conflict residual rate before and after repair

Before repair, all the conflict rule pairs in the test set were in the state to be processed. After automatic repair and verification, the overall number of residual conflicts decreased from 572 groups to 28 groups, and the comprehensive residual rate was 4.9%. The residual conflict rate of authorization result is the lowest, which is 2.1%. The residual rates of resource scope conflict and condition constraint conflict were 3.6% and 5.2%, respectively. The residual rate of priority conflict and inheritance conflict is relatively high, 6.8% and 8.4% respectively. The results show that the self-repair mechanism is stable in dealing with explicit authorization conflicts and resource boundary conflicts, and it still needs to retain the manual review channel for complex conflicts involving inheritance chain and execution order to ensure the security and traceability of repair results.

5.4 System stability and comparative analysis

The system stability evaluation is carried out for two scenarios: concurrent access and dynamic rule update. The concurrent pressure is set to 100, 300, 500, 800 and 1000 requests, and the rule size is set to 3000, 5000, 8000 and 12,000. Each set of experiments was run continuously for 30 min, and the average response time, P95 response time, request failure rate, and rule update blocking time were counted. In the dynamic update scenario, 200 new rules are incrementally written to the rule base every 5 minutes, and candidate conflict detection and repair queue update are synchronously triggered to observe the running state of the system when the rules continue to change.

The test results show that the average response time of the system stays between 42.6 ms and 86.4 ms, and the request failure rate is less than 0.20% under the scale of 3000-8000 rules. When the number of rules expands to 12000 and concurrent requests reaches 1000, the average response time increases to 138.7 ms, the P95 response time is 214.3 ms, and the request failure rate is 0.84%, which is still in the acceptable range. Compared with the direct update method without setting cache queue and repair sandbox, the P95 response time of the

complete system under high concurrency is reduced by about 27.6%, and the blocking time of rule update is reduced from 1.82 s to 0.73 s, which indicates that the rule version locking, candidate rule caching and asynchronous verification mechanisms can alleviate the impact of rule update on access request processing.

In order to present the response changes under the joint action of rule size and concurrent pressure, the system performance heatmap is constructed, as shown in Figure 11.

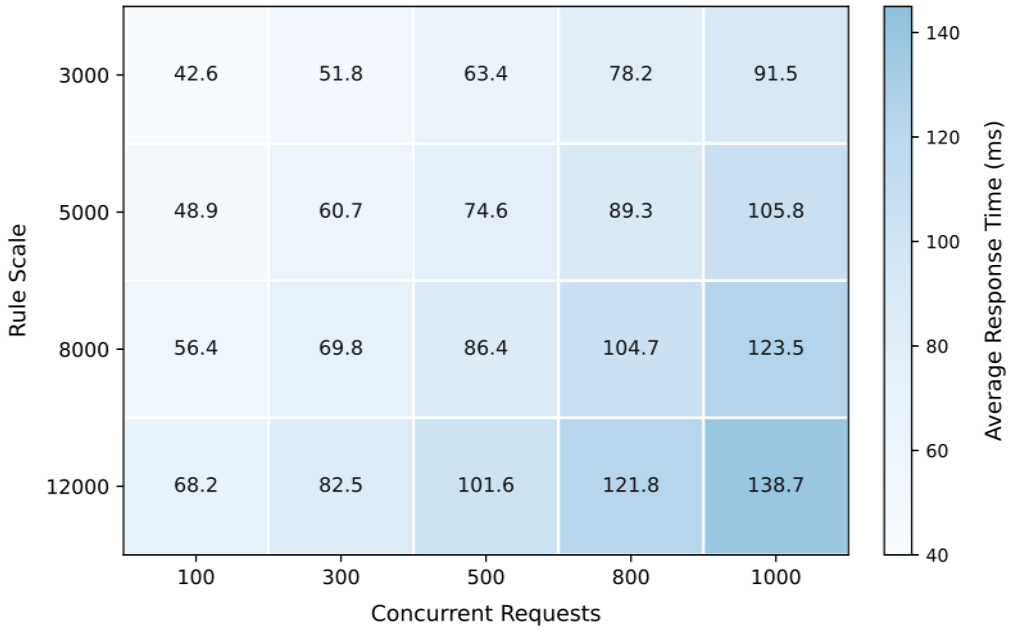

Figure 11: Heat map of system performance under concurrent pressure

The heatmap shows that the response time increases with the rule size and the number of concurrent requests, but the growth does not get out of control. The color change of the low concurrency region is gentle, which indicates that the rule size has limited influence on the system response. At high concurrency and the scale of rules reaches 12000, the response time increases significantly, and the main overhead comes from candidate conflict screening, rule version verification and log writing. In the dynamic update experiment, there is no long-term lock of the rule base or repair queue blocking, and the average completion time of the rollback operation is 96.5 ms. In general, the system has good stability in large-scale rule base and continuous update scenario, and can support the continuous operation of access control rule conflict detection, autonomous repair and audit trail.

6 Conclusions

Aiming at the requirements of conflict detection and self-repair for multi-modal access control rules, this paper constructs a processing framework linking rule formal modeling, feature coding, constraint judgment, risk classification and repair execution, and integrates the detection results into the candidate verification, exception rollback and log tracking processes. Comprehensive test data show that the full detection time of the complete method is 9.84s under the scale of 12000 rules, which can maintain a relatively stable computational efficiency when the scale of rules is expanded. The 572 groups of conflicts in the test set are processed by self-repair, and the residual conflicts are reduced to 28 groups, and the residual rate of authorization conflicts is 2.1%, which indicates that the repair strategy has a good

ability to deal with explicit authorization conflicts. In the concurrent stress test, the system P95 response time is 214.3ms under 1000 concurrent requests, and the rule update blocking time is reduced from 1.82s to 0.73s, which shows that the rule version locking, sandbox verification and asynchronous update mechanism can reduce the impact of repair process on business access. On the whole, this method can enhance the ability of conflict identification, repair verification and audit trail in heterogeneous rule environment.

References

[1] SYED N F, SHAH S W, SHAGHAGHI A, et al. Zero trust architecture (ZTA): A comprehensive survey[J]. IEEE Access, 2022, 10: 57143-57179.

[2] YU M J, LI F H, YU N H, et al. Detecting conflict of heterogeneous access control policies[J]. Digital Communications and Networks, 2022, 8(5): 664-679.

[3] CASERIO C, DELL'AMICO M, RODOLÀ G, et al. A formal validation approach for XACML 3.0 access control policies[J]. Sensors, 2022, 22(8): 2984.

[4] GUO F F, SHEN G H, HUANG Z Q, et al. DABAC: Smart contract-based spatio-temporal domain access control for the Internet of Things[J]. IEEE Access, 2023, 11: 36452-36463.

[5] AYASUNDARA S H, ARACHCHILAGE N A G, RUSSELLO G. SoK: Access control policy generation from high-level natural language requirements[J]. ACM Computing Surveys, 2024, 57(4): 1-37.

[6] ARSHAD H, HORNE R, JOHANSEN C, et al. Process algebra can save lives: Static analysis of XACML access control policies using mCRL2[C]// Formal Techniques for Distributed Objects, Components, and Systems. Lecture Notes in Computer Science. Cham: Springer, 2022, 13273: 11-30.

[7] AMEER S, GUPTA M, BHATT S, et al. BlueSky: Towards convergence of zero trust principles and score-based authorization for IoT enabled smart systems[C]// Proceedings of the 27th ACM Symposium on Access Control Models and Technologies. New York: ACM, 2022: 235-244.

[8] ALOHALY M, BALOGUN O, TAKABI D. Integrating cyber deception into attribute-based access control (ABAC) for insider threat detection[J]. IEEE Access, 2022, 10: 108965-108978.

[9] XIAO S Y, YE Y, KANWAL N, et al. SoK: Context and risk aware access control for zero trust systems[J]. Security and Communication Networks, 2022, 2022(1): 1-20.

[10] OLIVEIRA M T D, REIS L H A, VERGINADIS Y, et al. SmartAccess: Attribute-based access control system for medical records based on smart contracts[J]. IEEE Access, 2022, 10: 117836-117854.

[11] LIAO C H, GUAN X Q, CHENG J H, et al. Blockchain-based identity management and access control framework for open banking ecosystem[J]. Future Generation Computer Systems, 2022, 135: 450-466.

[12] KHOUMSI A. Automata-based study of dynamic access control policies[C]// Proceedings of the 9th International Conference on Information Systems Security and Privacy. Setúbal: SCITEPRESS, 2023: 218-227.

[13] MOHAMED A, SINGELEE D, JOOSEN W. XACML extension for graphs: Flexible authorization policy specification and enforcement for graph-structured data[C]// Proceedings of the 20th International Conference on Security and Cryptography. Setúbal: SCITEPRESS, 2023, 1: 442-449.

[14] MITANI S, KWON J, GHATE N, et al. Qualitative intention-aware attribute-based access control policy refinement[C]// Proceedings of the 28th ACM Symposium on Access Control Models and Technologies. New York: ACM, 2023: 201-208.

[15] ZHONGHUA C, GOYAL S B, RAJAWAT A S, et al. Smart contracts attribute-based access control model for security & privacy of IoT system using blockchain and edge computing[J]. The Journal of Supercomputing, 2024, 80(2): 1396-1425.

[16] SHAN F F, WANG Z Y, LIU M Y, et al. Automatic generation of attribute-based access control policies from natural language documents[J]. Computers, Materials & Continua, 2024, 80(3): 3881-3902.

[17] RUBIO-MEDRANO C E, KOTAK A, WANG W L, et al. Pairing human and artificial intelligence: Enforcing access control policies with LLMs and formal specifications[C]// Proceedings of the 29th ACM Symposium on Access Control Models and Technologies. New York: ACM, 2024: 105-116.

[18] VARSHITH H O S, GUPTA M, SANDHU R. Efficiently supporting attribute-based access control in Linux[J]. IEEE Transactions on Dependable and Secure Computing, 2024, 21(4): 2012-2026.

[19] ADHIKARI D, JIANG W, ZHAN J Y, et al. Recent advances in anomaly detection in Internet of Things: Status, challenges, and perspectives[J]. Computer Science Review, 2024, 54: 100665.

[20] AZAD M A, ABDULLAH S, ARSHAD J, et al. Verify and trust: A multidimensional survey of zero-trust security in the age of IoT[J]. Internet of Things, 2024, 27: 101227.