Postprint: Attribute-Based Access Control Policy Retrieval Method Based on Binary Sequences

Authors: Huang Hengjie, Pan Ruijie, Wang Gaocai, Qin Lizhong

Date: 2022-06-06T00:00:00+00:00

Abstract

In attribute-based access control, the ability to rapidly respond to and retrieve access control requests is crucial; however, retrieval methods that traverse all attribute values in each rule of the policy set to match corresponding rules are inefficient. Therefore, this paper proposes a binary-sequence-based retrieval method for attribute-based access control policies. The method employs binary identifiers and binary encodings to represent attribute-based access control policies and access control requests. By performing logical operations on binary identifiers to select appropriate groups, and subsequently locating suitable rules within each group through matching the binary encoding of the access control request with the binary encodings of all rules, the process of matching rule attributes from the policy set with attributes of the access control request is reduced, thereby improving policy retrieval efficiency. The paper experimentally compares the efficiency of similar retrieval methods across three aspects: policy preprocessing, policy evaluation time, and total policy retrieval time. Experimental results demonstrate that the proposed policy retrieval method achieves higher retrieval efficiency.

Full Text

Preamble

Volume 39, Issue 10
Application Research of Computers
ChinaXiv Partner Journal
Vol. 39 No. 10
Accepted Paper

Retrieval Method of Attribute Access Control Policy Based on Binary Sequence

Huang Hengjie[1], *Pan Ruijie*[2], *Wang Gaocai*[2]†, *Qin Lizhong*[1]
(1. Educational Technology Center, Yulin Normal University, Yulin, Guangxi 537000, China;
2. School of Computer & Electronic Information, Guangxi University, Nanning 530004, China)

Abstract: In Attribute-Based Access Control (ABAC), rapid response to access control requests is critical, yet traversing all attribute values in each rule of a policy set to find matching rules is inefficient. This paper proposes an ABAC retrieval method based on binary sequences, representing both policies and access control requests using binary identification and encoding. Through logical operations on binary identifiers, appropriate groups are selected. Within each group, matching between the binary code of the access request and those of all rules identifies suitable policies, reducing attribute matching processes between rule attributes and request attributes, thereby improving retrieval efficiency. Experimental comparisons with similar methods across three dimensions—policy preprocessing, evaluation time, and total retrieval time—demonstrate that the proposed method achieves higher retrieval efficiency.

Keywords: cloud computing; attribute access control policy; retrieval efficiency; matching; binary sequence

CLC Number: TP393.08
doi: 10.19734/j.issn.1001-3695.2022.03.0126

0 Introduction

The rise of cloud computing has provided modern society with convenient services for data sharing and integrated computation while enabling more efficient utilization of computing and storage resources. However, these resources contain substantial amounts of user privacy data that has not been satisfactorily protected by relevant institutions. Privacy data breaches can cause immeasurable losses to individuals and organizations alike [1]. For instance, the leakage of nearly 100,000 Westpac customers' private information and the exposure of 49 million Instagram users' data both stemmed from unauthorized access by illegal users and over-privileged access by legitimate users. Consequently, access control technology has attracted significant attention from researchers worldwide as an effective means of resource protection. In access control systems, access permissions are regulated and restricted by verifying requester identity information and establishing appropriate policies to protect object resources. These policies ensure that legitimate users can access and utilize resource services in complex network environments while preventing illegal users from stealing resources and blocking unauthorized access attempts by legitimate users. Among various approaches, Attribute-Based Access Control (ABAC) has gained widespread adoption in cloud computing scenarios due to its fine-grained granularity, flexibility, and ability to address large-scale user proliferation.

Although existing ABAC models [2] offer considerable advantages in policy expression and formulation, their application in large-scale policy environments characteristic of cloud computing suffers from low retrieval efficiency. According to ABAC and XACML (eXtensible Access Control Markup Language) theory [3], when a user initiates a request to access an object resource, the Policy Decision Point (PDP) invokes relevant attributes to parse and match against all policies. This means XACML-based policy evaluation requires traversing all policies for determination, with attribute matching involving comparison of all attribute information from the access control request against attributes in each rule of the policy set. This matching process includes string matching and numerical comparisons, and access permission is granted only when all attribute information matches the Attribute Access Request (AAR). Furthermore, during sequential rule matching in the policy set, if inconsistency with the AAR is only discovered at the last attribute of each rule, an excessive number of such rules will increase policy retrieval time. To address these issues, this paper proposes a binary sequence-based attribute access control policy retrieval method to improve retrieval efficiency.

The remainder of this paper is organized as follows: Section 1 reviews related theories and research work. Section 2 describes the proposed binary sequence-based ABAC policy retrieval method. Section 3 presents experimental results and analysis. Section 4 concludes the paper.

---

1 Related Theory and Research Work

Traditional access control models are no longer suitable for the multi-domain characteristics of cloud computing due to their inherent limitations. ABAC grants access permissions to subjects based on subject attributes, object attributes, environmental attributes, and other information, attracting considerable attention for its dynamic nature, fine granularity, and flexibility. ABAC can be represented as a quadruple (S, O, E, P), where S denotes subject, O denotes object, E denotes environment, and P denotes privilege.

An access control policy comprises subjects, objects, environment, and privileges, formally defined as: (permit, deny) ← (Attr(S), Attr(O), Attr(E), Attr(P)), representing the respective attribute value ranges. An access control policy typically consists of one or more rules, indicating whether subject S is permitted to perform operation P on object O under environmental condition E. The specific form is as follows:

Rule: access(S, O, E, P) ← f(SA, OA, EA, PA)

Here, f is a Boolean function that evaluates attributes of subjects, objects, environment, and privileges according to access control rules. If the result returns true, access is permitted; otherwise, it is denied. In ABAC authorization, four components provide critical support: Policy Decision Point (PDP), Pol-

icy Enforcement Point (PEP), Policy Administration Point (PAP), and Policy Information Point (PIP). The entire process comprises two phases:

(1) Preparation Phase:
PIP collects and manages attribute information for all subjects, objects, privileges, and environment, establishing relationships between attributes and privileges.
PAP obtains necessary attribute information and attribute-privilege relationships from PIP to construct ABAC policies.

(2) Execution Phase:
When PEP receives a Natural Access Request (NAR) from a subject, it requests PIP to obtain subject attributes, object attributes, environmental attributes, and privilege attributes needed to construct an Attribute-Based Access Request (AAR).
PDP verifies the subject' s identity information based on attribute information provided by PIP.
PDP evaluates the AAR forwarded by PEP according to policy query results provided by PAP to determine authorization.
PDP passes the decision result to PEP, which then enforces the decision.

Successful ABAC deployment requires two key components: well-defined policy description languages and effective policy retrieval methods. The former prevents cloud platform resource data leakage and unauthorized access, while the latter ensures timely response to access control requests.

Regarding well-defined policy description languages, researchers have proposed various solutions for different access control models and application scenarios, including XACML, graph-based policy visualization description languages for Web services [4], and RestPL for RESTful interfaces [5]. Among these, XACML is most commonly used and suitable for ABAC models. However, in practical cloud deployments, enterprises tend to prefer their own policy description languages. For large organizations, as ABAC scales up, the number of simultaneous access requests from users increases accordingly. Therefore, efficient ABAC policy retrieval is crucial for real-time response to user AARs and directly impacts user experience.

Research on efficient ABAC policy retrieval has yielded several achievements, broadly categorized into two types: improvements to XACML-standard policy retrieval methods and enhancements based on the Next Generation Access Control (NGAC) [6,7] standard.

Research on improving policy retrieval efficiency and reducing retrieval time remains limited. Traditional methods traverse all policies upon receiving an access control request. When policy redundancy and conflicts exist, XACML supports four combination algorithms: (1) first applicable, (2) only one applicable, (3) permit-override, and (4) deny-override [8]. To improve efficiency, reference [9] proposed a graph-based XACML policy evaluation method that optimizes two policy evaluation data structures—matching trees and merging trees

—and introduces a binary search algorithm for policy matching trees, though it does not support multi-valued attributes. Reference [10] introduced HPEngine, a new statistical analysis-based policy evaluation engine that converts policies from text to numerical form, reducing policy size while employing multi-cache mechanisms to accelerate policy lookup. Reference [11] proposed a prefix-based labeled operation policy retrieval method that adds binary prefixes based on policy attribute values to narrow the retrieval scope. While this method improves efficiency, prefix computation wastes time when attribute names match between policies and requests. To address this, reference [12] introduced a policy decision tree at the attribute-value level after adding binary identifiers to ABAC policies, demonstrating good scalability but with limitations when policies contain numerous attributes. Building on [11], reference [13] proposed an attribute grouping-based access control policy retrieval method that reduces retrieval scope through attribute-based policy grouping, thereby decreasing unnecessary computations and comparisons for efficient retrieval.

In the NGAC standard, the Policy Decision Point is responsible for retrieving rules matching access control requests [14]. During retrieval, AAR attributes must be compared against each rule' s attributes until a match is found, which is undoubtedly time-consuming. To this end, reference [15] proposed PolTree, a data structure that stores ABAC policies through two variants (PolTree and N-PolTree) to reduce attribute-value pair comparisons during retrieval, thereby improving efficiency.

Overall, various ABAC retrieval methods possess both advantages and disadvantages. While their efficiency improves upon traditional methods, limitations remain. Inspired by the ability of binary trie trees in routing to quickly locate matching leaf nodes in forwarding tables, this paper proposes a binary sequence-based attribute access control policy retrieval method. This approach uses binary identification for policy grouping, creating several policy set groups, and performs binary encoding of policies within each group. During access, the AAR undergoes binary identification and encoding, then logical operations between the request and policies select appropriate groups, filtering out numerous irrelevant policies. Finally, logical operations between the AAR' s binary code and group policy values avoid individual attribute matching, thereby improving retrieval efficiency.

2.1 Building Binary-Labeled Policies and AARs

To perform binary identification and encoding of ABAC policies and AARs, this work requires enumerating all attributes appearing in the access control policy set and arranging them in a specific order: subject, object, environment, and privilege. Internal attributes within each category must also be ordered to ensure binary identification validity. Attributes are categorized as directory attributes (decision information attribute values including {allow, deny}, with

exactly one decision attribute per rule) and non-directory attributes (information needed for decision-making, including subject, object, environment, and operation attributes). Rule encoding focuses solely on non-directory attributes, using binary digits 0 and 1 to indicate attribute presence (1) or absence (0) in access control requests and policy sets, forming binary identifiers for ABAC policies and AARs.

In cloud computing environments, users request services through their terminal devices. If we construct attribute information as shown in Table 1 and use it to build the ABAC policy set in Table 2 , binary identification applies only to non-directory attributes. The identifier dimension equals the number of non-directory attributes—specifically, the total attribute count across all access control rules determines the binary identifier length. For example, Table 2 shows a maximum of six non-directory attributes in any single rule, yielding a 6-bit binary identifier. If a rule contains a particular attribute, the corresponding bit is set to 1; otherwise, it is 0, meaning each bit indicates attribute presence in that rule.

Table 1 Attribute Combination
SA_{Role} | SA_{trust} | OA_{type} | OA_{trust} | EA_{Network} | PA_{permission}
student | personal | teacher | middle | common | middle
admin | public | delete | write

Table 2 Access Control Policy Set
attr(SA) | Attr(OA) | Attr(EA) | Attr(PA) | Decide | Binary Identifier

ABAC attribute values consist of two types: discrete and continuous attribute values.

1) Discrete Attribute Encoding
For instance, the subject attribute SA_{Role} in the above example has discrete values {student, teacher, admin}. If subject s1' s attribute sa1 is discrete, its values are independent and enumerable. This work employs dummy variable encoding: 1 indicates attribute value presence, 0 indicates absence. When no values exist for an attribute, all bits are set to 0. Assuming sa1' s value range is {sa1v0, sa1v1, sa1v2} with three possible values, three bits encode it. For attr(sa1)≠{sa1v1}, all bits except the second are set to 0, yielding code 101. For attr(sa1)={sa1v1, sa1v2}, the second and third bits are set to 1 (110). Subject s1' s attribute sa2 with range {sa2v0, sa2v1} uses two bits. Thus, five bits encode subject si attributes, as shown in Table 3 .

Table 3 Binary Encoding of Attributes
Subject Attribute Value Range | Binary Encoding
attr(sa1)={sa1=sa1v1} attr(sa2)={sa2=sa2v0} | ···
attr(sa1)={sa1≠sa1v1} attr(sa2)={sa2=sa2v0} | ···
attr(sa1)={sa1={sa1v1, sa1v2}} attr(sa2)={sa2=sa2v0} | ···
attr(sa1)={sa1=sa1v1} | ···
attr(sa2)={sa2=sa2v0} | ···

As previously described, ABAC access control comprises preparation and execution phases. During preparation, PAP performs binary identification on each rule in the policy set. During execution, only the AAR requires binary identification, which then undergoes logical AND operations with policy binary identifiers. When the operation result matches a rule' s binary identifier, other rule attributes are examined to verify AAR compliance.

To further reduce retrieval time, policies are grouped by binary identifier, constructing a structure similar to routing binary trie trees. In this tree, each node (except the root) represents a group, with rules sharing identical binary identifiers in the same group and different identifiers in separate groups.

Access control rule attributes represent requirements for AARs—successful matching requires the AAR to possess required attributes and values. This means matching is not strictly identical. During retrieval, only whether AAR attributes satisfy rule requirements is checked; extra AAR attributes do not affect matching success. For example, an AAR (SA_{Role}= "student" , SA_{trust}= "low" , OA_{trust}= "low" , OA_{type}= "personal" , EA_{Network}= "public" , PA_{permission}= "Denny") has binary identifier 111111, yet groups with identifiers 011111, 101111, ···, 111110 may also contain matching rules. When searching Table 2' s policy set, rule R2 in the 011111 group also matches this AAR.

2.2 Binary Encoding of Attribute Information

Binary identification and grouping of ABAC policies effectively narrow retrieval scope, limiting it to groups matching the binary identifier and filtering numerous irrelevant rules to reduce retrieval time. However, within-group retrieval requires matching each rule' s attribute values against the AAR, with worst-case scenarios discovering mismatches only at the last attribute, causing time waste. If many such rules exist, retrieval time increases significantly. To address this, this paper proposes binary encoding for each group' s access control policies. This enables authorization decisions through logical operations between the AAR's binary code and policy binary codes, eliminating individual attribute value judgments for each policy rule and saving retrieval time.

2) Continuous Attribute Encoding

Specific access control environments may require resource access within certain time periods or trust levels between values, necessitating continuous attributes. Non-enumerable continuous attribute values must be discretized. Common continuous attributes in access control systems include trust and temporal attributes. Using trust attributes as an example (temporal attributes follow similar encoding and discretization), the subject' s SA_{trust} and object' s OA_{trust} attributes represent trust values calculated using methods such as that proposed in reference [16].

2.3 Analysis of Binary Sequence-Based Policy Retrieval Method

Access control policy attribute value ranges represent requirements for AAR attribute combinations—specific attributes and values needed to obtain particular permissions. This means policy-request matching is not strictly identical. Policy matching involves string comparisons that only check whether the request possesses relevant attribute information; extra request attributes do not affect matching success. For example, a policy permitting subject access to object resources during working hours will allow access even if the request contains additional role information.

Using Table 2' s policy set, consider an attribute-based access control request built from Table 1' s attributes: {SA_{role}=student, SA_{trust}=low, OA_{trust}=low, EA_{Network}=work, PA_{permission}=delete}. Applying the proposed binary sequence-based retrieval method, Table 2' s rules encode as: (R1, 10010010100100100), (R2, 00010010100001100), (R3, 10000000100010100), (R4, 10000000100100100). The AAR' s binary identifier is 110111 (all attributes except OA_{type} have values). The AAR encodes as (AAR, 10010000100010100). The logical AND of request and policy binary identifiers (110111 & 111111 = 110111 ≠ 111111) indicates R1' s group fails attribute requirements and is skipped. Similarly, 110111 & 011111 = 010111 ≠ 011111 filters out R2' s group. Querying the third group with identifier 100111 yields 110111 & 100111 = 100111, indicating R3' s group requires further searching. The logical AND of R3' s binary code (10000000100010100) with the AAR' s code equals R3' s code (10000000100010100 & 10010000100010100 = 10000000100010100), identifying R3 as the matching rule. Since its decision attribute is Deny, PDP' s decision rejects the access request, concluding retrieval.

Traditional retrieval would require matching each attribute of every rule until finding the appropriate one, executing 18 matches in this case. Binary identifier grouping reduces this to 10 matches through logical AND operations between AAR and policy identifiers. The proposed binary sequence-based method requires only binary identifier and code matching, executing merely 4 matches. The retrieval process is illustrated in Figure 1 [Figure 1: see original paper].

Figure 1 Process of Policy Retrieval Mechanism Based on Binary Sequence

The core algorithm for the binary sequence-based ABAC policy retrieval method is presented as Algorithm 1.

Algorithm 1 Binary Identification-Based Policy Retrieval Algorithm
Input: policy_{set}, AAR */policy set, access control request to evaluate/*
Output: Decision */decision result/*

1. Step 1: Identify and encode attribute-based access control requests.
2. Init(AAR.policy_{code}); */initialize AAR binary identifier and policy code bits/*
3. for (i=0; i < AAR.size(); i++) do
4. if (AAR[i]) exists then
5. `AAR.policy_{groupbinary}.setposition(p)=1 /*perform binary identification on AAR*/`
6. end if
7. end for
8. for (j=0; j < AAR.policy_{code}.size(); j++) do */perform binary encoding on AAR/*
9. if (AAR[j].attrvalue) exists then
10. a=getposition(AAR[j].attrvalue);
11. AAR.policycode.setposition(a)=1;
12. end if
13. end for
14. Step 2: Retrieve policy table.
15. for (i=0; i < Group.size; i++) do */perform policy retrieval/*
16. if (Group[i].groupbinary & AAR.groupbinary) == Group[i].groupbinary then
17. for (g=0; g < Group[i].size; g++) do
18. `if (Group[i].ruleset[g].policycode & AAR.policy_{code} == Group[i].ruleset[g].policycc`

```
    decide(); /*make decision*/
  end if
```

end for

end if

end for

From the AAR binary identification and encoding algorithm, the maximum processing operations equal the number of AAR attributes, yielding O(n) time complexity. Policy retrieval examines each rule in every group, resulting in $O(n^2)$ complexity. Typically, n is small (e.g., n=6 in our experiments).

---

3 Experimental Results and Analysis

To evaluate the proposed method' s efficiency, we implemented test code in C++ using Qt Creator on a Windows 10 platform, with Matlab for data analysis. Evaluating retrieval methods requires substantial ABAC policies and AARs, but real industry data remains confidential. Router access control policies typically number 300-500 entries; reference [13] experimented with 105, 210, 315, 433, and 525 policies. To demonstrate algorithm efficiency and address large-scale scenarios, this work examines larger policy counts: 2000, 2500, and 3000. We implemented a policy and request generator producing 4000 ABAC policies and 2000 AARs, with each policy containing one rule. To ensure binary identifier validity, we applied data classification standardization from reference [17]. Experiments evaluate retrieval efficiency from three perspectives: policy preprocessing, evaluation time, and total retrieval time. All results represent averages across 10 trials for generality. The experimental environment: CPU: 11th Gen Intel(R) Core(TM) i5-1135G7 @ 2.40GHz, RAM: 16.00GB; Qt Creator 3.3.0 (opensource) Based on Qt 5.4.0 (MSVC 2010, 32 bit); Matlab Version: 8.6.0.267246 (R2015b).

In the following experiments, we compare our binary identification and encoding method with similar grouping and prefix-based approaches from references [11] and [13]. SG-PRM denotes the prefix-based policy retrieval method from [11], AG-PRM denotes the attribute grouping-based method from [13], and B-S-PRM represents our proposed method. All comparisons use identical policies and requests. Policy complexity refers to the number of attribute key-value pairs per rule.

3.1 Policy Preprocessing Time

Policy preprocessing occurs during the access control preparation phase, primarily deploying policy attributes. For our method, preprocessing time includes

PAP' s binary identification, grouping, and encoding of each rule. SG-PRM performs binary identification during preprocessing, while AG-PRM performs both identification and grouping. As Figure 2 [Figure 2: see original paper] shows, with policy counts of 500, 1000, 1500, 2000, and 2500, preprocessing times for B-S-PRM, SG-PRM, and AG-PRM increase linearly with policy count. AG-PRM and SG-PRM exhibit similar preprocessing times, both lower than B-S-PRM, because our method adds binary encoding to the processes of the other two methods. The binary sequence-based method (B-S-PRM) indeed requires more time for policy analysis. However, ABAC policy preprocessing occurs during system preparation and does not affect retrieval time, making this overhead acceptable.

Figure 2 Policy Preprocessing

3.2 Policy Evaluation Time

Policy evaluation occurs during the execution phase at the PDP, primarily retrieving matching rules and making decisions. Figures 3(a), 3(b), and 3(c) show evaluation time changes as access request counts increase under different policy rule counts (2000, 2500, 3000). All three methods' retrieval times increase with request count. SG-PRM performs logical OR operations between request prefixes and policy prefixes, matching attribute information only when results align with policy prefixes. AG-PRM reduces retrieval scope through attribute-based policy grouping.

Figure 3 The Policy Evaluation Time Under Different Policy Rules

Across different policy conditions, all three methods' evaluation times increase with request count. However, B-S-PRM' s evaluation time decreases as policy count increases. With 3000 policies, B-S-PRM' s evaluation time is approximately half that with 2000 policies, with fluctuation range gradually narrowing, while the other methods show no significant change. SG-PRM evaluation time fluctuates between 4-32ms, AG-PRM between 4-27ms, and B-S-PRM between 1-11ms, demonstrating our method' s greater stability and approximately 3× efficiency improvement over SG-PRM and AG-PRM. This confirms the method' s applicability in high-policy environments.

3.3 Total Policy Retrieval Time

Total ABAC policy retrieval time includes PEP's AAR processing time and PDP' s policy evaluation time. As Figure 4 [Figure 4: see original paper] shows, with 3000 total policies, all three methods' request processing times increase nearly linearly with request count. SG-PRM and AG-PRM show similar processing times, while B-S-PRM requires approximately double their processing time due to its more complex request handling (binary encoding and identification versus simple prefix addition for SG-PRM or grouping identification for AG-PRM). However, Figure 5 [Figure 5: see original paper] reveals that despite longer PEP processing time, B-S-PRM' s total retrieval time decreases substantially—

approximately 4× faster than SG-PRM and 3.5× faster than AG-PRM. As request counts increase, our method' s advantage becomes more pronounced, with more stable retrieval times. Unlike the other methods, B-S-PRM avoids traversing attribute-value pairs within rules, instead selecting appropriate groups and traversing only rules within those groups, dramatically reducing retrieval scope and improving efficiency.

Figure 4 The Processing Time of Access Control Request
Figure 5 The Total Policy Retrieval Time

In summary, although our method requires longer preprocessing time, it significantly reduces retrieval time with clear efficiency advantages. As policy scale increases, retrieval time remains relatively stable. When deployed in cloud computing-based ABAC systems, this method can shorten user waiting times and respond to access control requests in real-time.

---

4 Conclusion

With the rise of cloud computing and IoT, modern network environments increasingly exhibit massive scale and dynamic characteristics, featuring enormous user and resource populations with constantly changing access contexts. As an effective technology for protecting object resources, ABAC has been widely adopted. Users expect to obtain resource access permissions within short timeframes, necessitating reduced policy retrieval and response times. However, existing ABAC policy retrieval methods exhibit limitations in large-scale scenarios. To address this issue, this paper proposes a binary sequence-based attribute access control policy retrieval method that performs binary identification and encoding on both ABAC policies and requests, then groups policies by binary identifier. During retrieval, logical AND operations between request and group binary identifiers select appropriate groups, filtering numerous irrelevant policies. Subsequently, logical operations between request binary codes and rule binary codes within groups identify suitable rules, reducing attribute matching processes. Experimental results validate that the proposed method achieves lower retrieval time and higher efficiency.

Future research will focus on combining this retrieval method with attribute encryption to further enhance attribute security. Additionally, as the current method remains theoretical, optimizing it for practical deployment and real-world applications represents another future direction.

References

[1] PanJun Sun. Security and privacy protection in Cloud Computing: discussions and challenges [J]. Journal of Network and Computer Applications. 2020,

160 (102642): 1-22
[2] Servos D, Osborn S L. Current research and open problems in attribute-based access control [J]. ACM Computing Surveys, 2017, 49 (4): 1-45.
[3] OASIS XACML. eXtensible access control Markup language XACML version 3.0 [S]. OASIS Standard, Munich, 2011.
[4] D. Nabil, H. Slimani and H. Nacer, et al. ABAC conceptual graph model for composite Web services [C]// The 5th IEEE International Congress on Information Science and Technology, Marrakech, Morocco, IEEE 2018: 36-41
[5] 罗杨, 沈晴霓, 吴中海. 一种新的访问控制策略描述语言及其权限划分方法 [J]. 计算机学报, 2018, 41 (6): 969-986 (Luo Yang, Shen Qingni, Wu Zhonghai. A novel access control policy specification language and its permission classification method [J]. Chinese Journal of Computer, 2018, 41 (6): 969-986)
[6] Jun Ho Huh, Rakesh B. Bobba, Tom Markham, et. al. Next-Generation access control for distributed control systems. IEEE Internet computing, 2016, 20 (5): 28-37
[7] Working DRAFT Information technology-Next Generation Access Control – Generic Operations and Data Structures (NGAC-GOADS), INCITS 499-2013, American National Standard for Information Technology, American National Standards Institute, 2014.
[8] Mohamed Mejri, Hamdi Yahyaoui, Azzam Mouradc, et al. A rewriting system for the assessment of XACML policies relationship [J]. Computers & Security, 97 (2020), 2020, 1-12.
[9] Santiago Pina Ros, Mario Lischka, Félix Gómez Mármol. Graph-based XACML evaluation [C]// In Proceedings of 17th ACM symposium on Access Control Models and Technologies (SACMT), New York, NY, 2012, 83-92.
[10] D. H. Niu, J. F. Ma, Z. Ma, C. N. Li, L. Wang. HPEngine: High performance XACML policy evaluation engine based on statistical analysis [J]. Journal of Communication. 2014, 35 (8): 206-215.
[11] 邹佳顺, 张永胜. ABAC 中基于前缀标记运算的策略检索方法 [J]. 计算机工程与设计, 2015, 36 (11): 2943-2947. (Zou Jiashun, Zhang Yongsheng. Policy retrieval method based on prefix sign operation in ABAC model [J]. Computer Engineering and Design, 2015, 36 (11): 2943-2947)
[12] Liu M P, Cheng Y, Hao Li, et al. An efficient attribute-based access control (ABAC) policy retrieval method based on attribute and value levels in multimedia networks [J], Sensors, 2020, 20 (6), 1-15.
[13] 黄美蓉, 欧博. 基于属性分组的访问控制策略检索方法 [J]. 计算机应用研究, 2020, 37 (10): 3096-3100+3106. (Huang Meirong, Ou Bo. Attribute and RBAC based hybrid access control model [J]. Application Research of Computers, 2020, 37 (10): 3096-3100+3106)
[14] Vincent Hu, David F. Ferraiolo, D. Richard Kuhn, et al. Implementing and managing policy rules in attribute based access control [C]// 2015 IEEE 16th International Conference on Information Reuse and Integration, San Francisco, California, IEEE Press, 2015, 518-525
[15] Ronit Nath, Saptarshi Das, Shamik Sural. PolTree: A data structure for making efficient access decisions in ABAC [C]// In The 24th ACM Symposium on Access Control Models and Technologies (SACMAT' 19), Toronto, ON, ACM

Press, 2019, 25-35.
[16] Zhao Z. Y., Sun L. Attribute-based access control with dynamic trust in a hybrid cloud computing environment [C]// Proceedings of the 2017 International Conference on Cryptography, Security and Privacy. New York, ACM Press, 2017: 112-118
[17] Shaikh R A, Adi K, Logrippo L. A data classification method for inconsistency and incompleteness detection in access control policy sets [J]. International Journal of Information Security, 2017, 16 (1): 91-113

Note: Figure translations are in progress. See original paper for figures.