Security Policy Tool: Access Control Security Policy Editing, Testing, Verification, and XACML Deployment

(InfoBeyond Technology LLC)

Abstract—This document introduces Security Policy Tool which is a software tool for Policy Authors, Software Developers, IT Access Control Security Managers, Cybersecurity Specialists, or other professionals that specialize in the performance of access control systems. Access control policies are broadly enforced to protect the accessibility of the online resources in networks, IoTs, healthcare systems, financial service systems, enterprise IT and clouds, military systems. There are several issues to build a robust access control model for these systems: (i) how to effectively compose the policies and their rules, (ii) how to test these policies, (iii) how to verify these policies to prevent unintended access control holes that leave online resources unprotected. In addition, manually developing XACML policies is time and cost expensive, and a change of a rule in a policy may affect the access control model. Do we know the effectiveness of the policy or rule change in the entire access control system? Security policy tool targets for solving these issues in an attempt to provide access control policy editing, test, and verification for you to build a robust access control security model to prevent unintended access control holes.

Index Terms—Access control, security policy editing, test, verification, deployment, access control leaks, access control holes, XACML, software tool.

1 Access Control Security

Security policies govern the accessing privilege of all the resources in an access control system. This protects the network, financial, enterprise, organization, healthcare, defense, and various IT resources in an online service system. In order to provide comprehensive security and privacy protection, three steps have to be performed:

- **Build an Access Control Security Model**: This step chooses an access control security model to meet the security and operational requirements. The available access control security models are ABAC (attributed based access control), RBAC (role-based access control), PBAC (proximity-based access control), MLS (Multilevel security, Workflow, and others. Upon the determination of an appropriate access control model, a variety of security policies with rules are composed to specify the accessibility of a request. In general, a policy consists of rules to securely manage (e.g., Permit or Deny) the Subjects, Resource, Actions, Environments, and Conditions.
- **Verify the Access Control Security Model**: The security model has to be verified in different user requesting scenarios to prevent access control errors. NIST has published several types of errors found currently [1], such as Block Privilege, Leak Privilege, Not Protected Resource, Inconsistent Assignment, Privilege Inheritance Loop, Undecided Rules, Separation of Duty Error. A Leak Privilege error is

- *Contact us at: E-mail: Info@Securitypolicytool.com*

Security Policy Tool (www.Securitypolicytool.com) is a commercial version of NIST(National Institute of Standards and Technology)s ACPT (Access Control Policy Tool) . ACPT is developed by NIST for Proof of Concept with some capability restrictions. With tremendous consultant to NIST experts, Security Policy Tool substantially enhances and expands the NISTs ACPT design with advanced features for achieving high security confidence access control levels such that it can be commercialized. The development of Security Policy Tool is financially sponsored by NIST via a SBIR (Small Business Innovation Research) Phase I and II programs. It specifically improves the NISTs ACPT design to provide a robust, unified, professional, and functionally powerful access control policy tool.

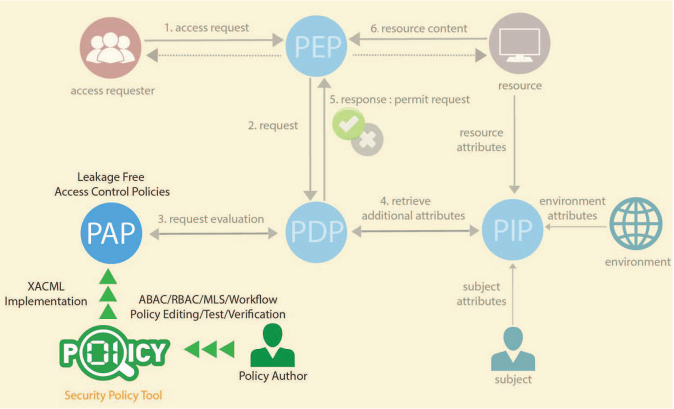


Fig. 1. XACML Framework for Access Control - A flow to authorize an service request to an access control system

an unintended access control privilege that causes data breach, network hijack, service loss, and other security vulnerabilities: *It is common that a system's privacy and security are compromised due to the misconfiguration of access control policies instead of the failure of cryptographic primitives or protocols.* as stated by NIST [2]. Verification of an access control model is the task to inspect and correct the misconfiguration of access control policies potentially hidden in an access control model. The difficulty of verifying an access control security model quickly increases along with the number of policies and rules defined in the model.

- **Implement the Security Model**: An access control security model is implemented in the access control system after it is verified. OASIS defines an XACML framework that includes PAP (Policy Administration Point), PDP (Policy Decision Point), PEP (Policy Enforcement Point), PIP (Policy Information Point), PRP (Policy Retrieval Point) as indicated in Figure 2. Security access control policies are written into the PIP via PAP such that the PDP evaluates a request again the policies when it receives a request from PEP. Currently, more and more access control systems are implemented in such a framework that implements an authorization process to determinate the privilege of a request.

Faulty access control policies, misconfigurations, or flaws in software implementation can result in serious security vulnerabilities. On the other hand, the access control flaws could be hidden from our direct detections until observable damages are caused. As a result, the specification of the policies for an access control security model is often a challenging problem:

1) How to cost-effectively compose policies and their rules?
2) How to test these policies before the policies are deployed into the PIP in an access control system?
3) How to verify these policies to prevent unintended access control holes that leave online resources unprotected?

Such a problem becomes increasingly severe as an access control software system become more and more complex that have a number of access control policies. These policies are deployed to manage a large number of sensitive resources that are organized into sophisticated structures. Identifying discrepancies between policy specifications and their properties (intended function) are crucial because correct implementation and enforcement of policies by applications are based on the premise that the policy specifications are correct. As a result, policy specifications must undergo rigorous verification and validation through systematic testing to ensure that the policy specifications truly encapsulate the desires of the policy authors. In other words, it is indefensible to conduct comprehensive security tests and verifications before the access control policies are deployed into an access control system.

2 Security Policy Tool

Security Policy Tool delivers such a solution for editing, testing, analyzing, and verifying access control policies. Figure 2 illustrates the functionalities in the OASIS-defined XACML framework. This tool allows security specialists to build a robust access control security model while validating and fixing the hidden faulty, unintended, or misconfigured policies. The verified policies enables the intended decisions of the Request Evaluation from PDP. This achieves the goal to instill the confidence that classified assets in the Big data, Cloud, IoT,

Fig. 2. Security Policy Tool - Policy Editing, Testing, and Verification

Cybersecurity, and other access control systems are protected at the level your organization intends them to be. Figure 3 demonstrates the Security Policy Tool to enable powerful, thoughtful, and convenient access control policy editing, testing and analyzing functions such that the policy authors can validate and fix the faulty, unintended, misconfigured policies. At first, it offers the functions to conveniently build an access control security model by composing a large number of rules and policies with ease. For such a security model, test and verification are then conducted. By testing and analyzing the Verification Results ($FALSE$ or $TRUE$ in Figure 3), it minimizes the potential security errors when the policies are deployed in a system. Very importantly, access control policies can be effectively analyzed via user-friendly GUI (graphic user interface) to find unintended accessibility, as indicated in Figure 3. With the identification of the faulty and unintended policies, the policy author can revise the rules in policies to exclude the access control vulnerabilities till all verification results are intended. For the capability of conveniently inspecting an access control error, Security Policy Tool provides the function for the policy author to find/track the correlations among the rules and the access control accessibility.

More specifically, Security Policy Tool has four major capabilities: (i) Policy Development, (ii) Policy Tests and Verifications, (iii) Policy Analysis, Access Control Flaw Inspection and Correction, (iv) XACML Editor, Checker, and Converter, which will be further discussed.

2.1 Policy Development

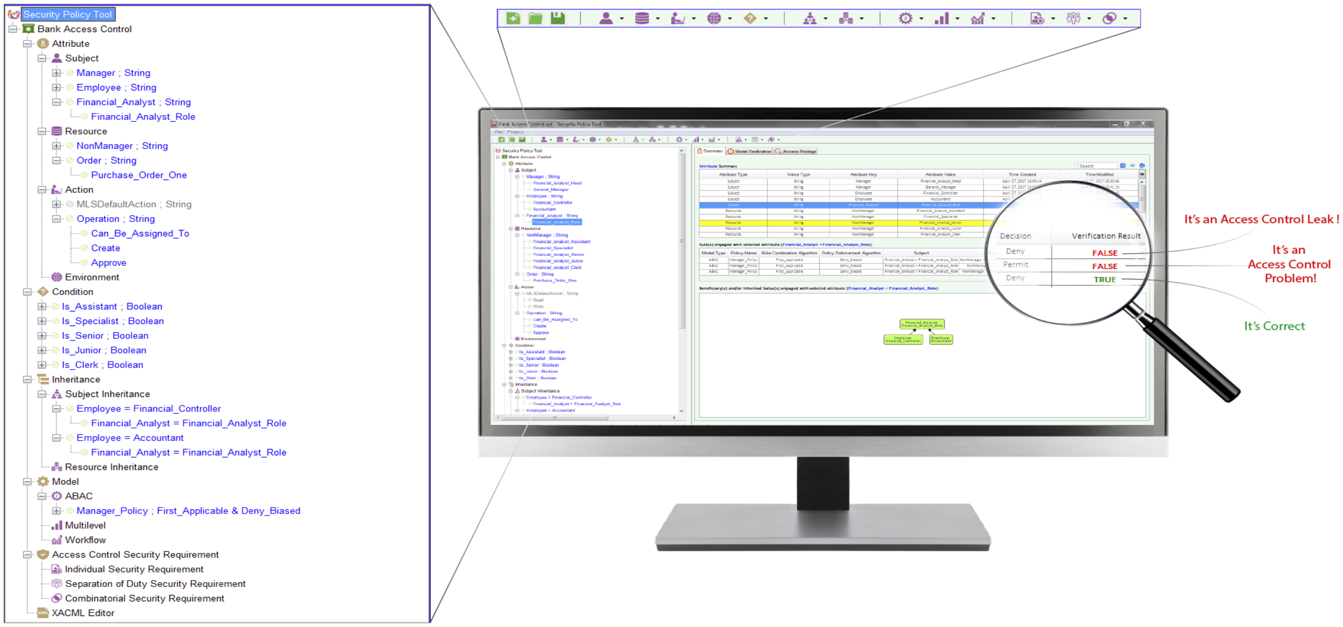
Security Policy Tool saves you valuable time during the stage of "Build an Access Control Security Model". It allows a policy author to conveniently compose a large number of rules and apply them the ABAC, MLS, and Workflow templates. RBAC, PBAC, and other security models can be built as an ABAC form. It enables XACML-compatible policy composition and policy development which in turn makes it easier to use the policy editing functions. Particularly, policy development allows you to quickly and flexibly define attributes, rules, policies, and then build an access control security model such as to:

1) Define, Update, and Edit Subjects, Resources, Actions, Environments, and Conditions via Graphic User Interfaces
2) Build an Access Control Security model by using ABAC, Multilevel Security, and Workflow access control templates
3) Define, Verify, Review, Search, and Manage a number of rules and policies effectively
4) Update and compose inheritance relations for hierarchical organizations to effectively manage a number of Subjects and Resources in a large-scale access control system
5) Automatically detect and prevent an inheritance loop before they occur
6) Export the attributes and policies in Excel tables, giving you full power of your own system

Figure 4 shows the user interface to compose a rule for a policy in an ABAC model. From the interface, a rule for a policy can be easily created by using the predefined Subjects, Resource, Actions, Environments, and Conditions. It is worth mentioning that Security Policy Tool prevents potential errors for defining a security model and presents many functions to manage all the elements of the access control security model.

2.2 Security Policy Test and Verification

Due to the complex nature of designing access control policies, errors commonly come up in the access control security model. Before

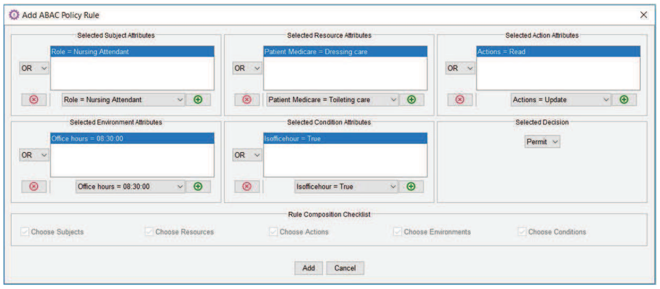


```
Security Policy Tool
  Bank Access Control
    Attribute
      Subject
        Manager ; String
        Employee ; String
        Financial_Analyst ; String
          Financial_Analyst_Role
      Resource
        NonManager ; String
        Order ; String
          Purchase_Order_One
      Action
        MLSDefaultAction ; String
        Operation ; String
          Can_Be_Assigned_To
          Create
          Approve
      Environment
    Condition
      Is_Assistant ; Boolean
      Is_Specialist ; Boolean
      Is_Senior ; Boolean
      Is_Junior ; Boolean
      Is_Clerk ; Boolean
    Inheritance
      Subject Inheritance
        Employee = Financial_Controller
          Financial_Analyst = Financial_Analyst_Role
        Employee = Accountant
          Financial_Analyst = Financial_Analyst_Role
      Resource Inheritance
    Model
      ABAC
        Manager_Policy ; First_Applicable & Deny_Biased
      Multilevel
      Workflow
    Access Control Security Requirement
      Individual Security Requirement
      Separation of Duty Security Requirement
      Combinatorial Security Requirement
    XACML Editor
```

Decision	Verification Result
Deny	FALSE
Permit	FALSE
Deny	TRUE

It's an Access Control Leak !
It's an Access Control Problem!
It's Correct

Fig. 3. Security Policy Tool allows you to edit, test, analyze, verify, correct an access control security model till its access control effectiveness is free of error to your intended access control privileges



Add ABAC Policy Rule

Selected Subject Attributes: OR; Role = Nursing Attendant; Role = Nursing Attendant

Selected Resource Attributes: OR; Patient Medicare = Dressing care; Patient Medicare = Toileting care

Selected Action Attributes: OR; Actions = Read; Actions = Update

Selected Environment Attributes: OR; Office hours = 08:30:00; Office hours = 08:30:00

Selected Condition Attributes: OR; Isofficehour = True; Isofficehour = True

Selected Decision: Permit

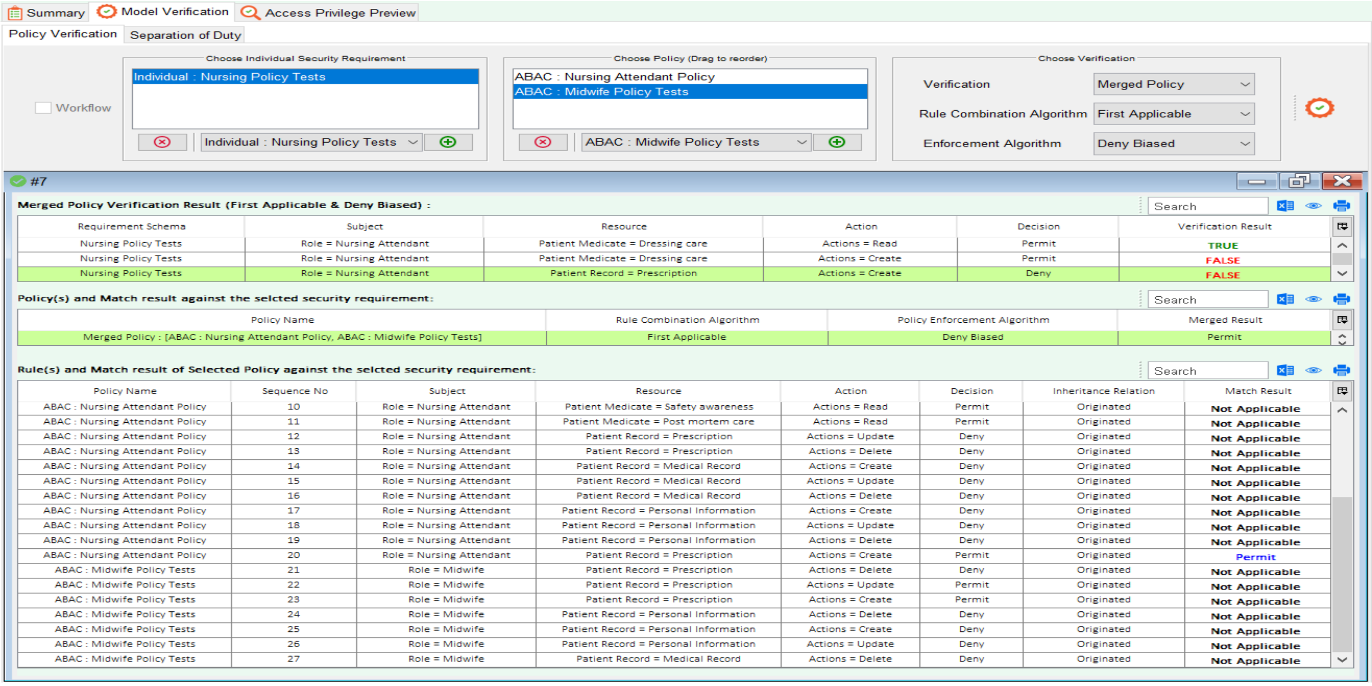
Rule Composition Checklist: ☑ Choose Subjects ☑ Choose Resources ☑ Choose Actions ☑ Choose Environments ☑ Choose Conditions

Fig. 4. Rule composition for a rule from defined Subjects, Resource, Actions, Environments, and Conditions

deploying your access control policy, Security Policy Tool can verify the effectiveness (i.e., Permit or Deny) of your composed policy in comparison to your organizations security requirements. Using our testing engine, a policy author can analyze the results to verify if the access control decisions from the tests are working as planned. If all the results are matched with the intended results, the policies including the rules and the algorithms in the access control models can be deployed into your access control system. In this circumstance, there are access control errors the policy author will need to revise their policies or algorithms and test for new results using Security Policy Tool. Once the expected results are achieved, your organization will have the peace-of-mind knowing that your intended level security is being reached. The policy tests support all XACML 3.0 rule-combining algorithms that include First Applicable, Deny Override, Permit Override, etc. It also supports rule enforcement algorithms that are Deny Based and Permit Based.

Security Policy Tool specifically enables the following policy testing and verification functions for policy error detection:

1) **Policy Verification**: Security Policy Tool can test one or multiple policies against specific security requirements. A security requirement represents an access request, e.g., a case of a Subject requests an Action to a Resource under certain Conditions and Environments. In the case of multiple policies, it can test/verify the security policies through a merging or a combining policy process. The detailed testing results allow you to check if there are access control leaks or not against each rule.
2) **Combinational or Exhaustive Policy Verification**: Security Policy Tool can automatically generate a number of Security



□ Workflow

Choose Individual Security Requirement

Individual : Nursing Policy Tests

Individual : Nursing Policy Tests

Choose Policy (Drag to reorder)

ABAC : Nursing Attendant Policy
ABAC : Midwife Policy Tests

ABAC : Midwife Policy Tests

Choose Verification

Verification: Merged Policy
Rule Combination Algorithm: First Applicable
Enforcement Algorithm: Deny Biased

#7

Merged Policy Verification Result (First Applicable & Deny Biased) :

Requirement Schema	Subject	Resource	Action	Decision	Verification Result
Nursing Policy Tests	Role = Nursing Attendant	Patient Medicate = Dressing care	Actions = Read	Permit	TRUE
Nursing Policy Tests	Role = Nursing Attendant	Patient Medicate = Dressing care	Actions = Create	Permit	FALSE
Nursing Policy Tests	Role = Nursing Attendant	Patient Record = Prescription	Actions = Create	Deny	FALSE

Policy(s) and Match result against the selcted security requirement:

Policy Name	Rule Combination Algorithm	Policy Enforcement Algorithm	Merged Result
Merged Policy : [ABAC : Nursing Attendant Policy, ABAC : Midwife Policy Tests]	First Applicable	Deny Biased	Permit

Rule(s) and Match result of Selected Policy against the selcted security requirement:

Policy Name	Sequence No	Subject	Resource	Action	Decision	Inheritance Relation	Match Result
ABAC : Nursing Attendant Policy	10	Role = Nursing Attendant	Patient Medicate = Safety awareness	Actions = Read	Permit	Originated	Not Applicable
ABAC : Nursing Attendant Policy	11	Role = Nursing Attendant	Patient Medicate = Post mortem care	Actions = Read	Permit	Originated	Not Applicable
ABAC : Nursing Attendant Policy	12	Role = Nursing Attendant	Patient Record = Prescription	Actions = Update	Deny	Originated	Not Applicable
ABAC : Nursing Attendant Policy	13	Role = Nursing Attendant	Patient Record = Prescription	Actions = Delete	Deny	Originated	Not Applicable
ABAC : Nursing Attendant Policy	14	Role = Nursing Attendant	Patient Record = Medical Record	Actions = Create	Deny	Originated	Not Applicable
ABAC : Nursing Attendant Policy	15	Role = Nursing Attendant	Patient Record = Medical Record	Actions = Update	Deny	Originated	Not Applicable
ABAC : Nursing Attendant Policy	16	Role = Nursing Attendant	Patient Record = Medical Record	Actions = Delete	Deny	Originated	Not Applicable
ABAC : Nursing Attendant Policy	17	Role = Nursing Attendant	Patient Record = Personal Information	Actions = Create	Deny	Originated	Not Applicable
ABAC : Nursing Attendant Policy	18	Role = Nursing Attendant	Patient Record = Personal Information	Actions = Update	Deny	Originated	Not Applicable
ABAC : Nursing Attendant Policy	19	Role = Nursing Attendant	Patient Record = Personal Information	Actions = Delete	Deny	Originated	Not Applicable
ABAC : Nursing Attendant Policy	20	Role = Nursing Attendant	Patient Record = Prescription	Actions = Create	Permit	Originated	Permit
ABAC : Midwife Policy Tests	21	Role = Midwife	Patient Record = Prescription	Actions = Delete	Deny	Originated	Not Applicable
ABAC : Midwife Policy Tests	22	Role = Midwife	Patient Record = Prescription	Actions = Update	Permit	Originated	Not Applicable
ABAC : Midwife Policy Tests	23	Role = Midwife	Patient Record = Prescription	Actions = Create	Permit	Originated	Not Applicable
ABAC : Midwife Policy Tests	24	Role = Midwife	Patient Record = Personal Information	Actions = Delete	Deny	Originated	Not Applicable
ABAC : Midwife Policy Tests	25	Role = Midwife	Patient Record = Personal Information	Actions = Create	Deny	Originated	Not Applicable
ABAC : Midwife Policy Tests	26	Role = Midwife	Patient Record = Personal Information	Actions = Update	Deny	Originated	Not Applicable
ABAC : Midwife Policy Tests	27	Role = Midwife	Patient Record = Medical Record	Actions = Delete	Deny	Originated	Not Applicable

Fig. 5. Security Policy Tool allows you to build and choose a test cases and tests policies against these cases to analyze if the access control effectiveness is intended or not. It further allows you to track the rules that yield the effectiveness ($FALSE$ or $TRUE$).

Requirements by pairwise or all-pairs attribute combinatorial algorithms. The collection of these requirements is called a testing suite. Similar to software testing, the testing suite allows a policy author to achieve a certain testing coverage, which reaches an access control flaw detection probability. Pairwise testing is commonly suggested as its testing coverage can find 50% - 90% access control flaw detection probability. 4-way combinatorial testing can discover most complex access control flaws and its testing coverage can discover very close to 100% AC flaw detection probability.

3) **Separation of Duty**: Security Policy Tool can also test Separation of Duty. This tests if there are conflicts among two or more security requirements.

Security Policy Tool allows the policy author to test and verify different security requirements and the results are presented in tables in order for inspecting source that causes the results. Figure 5 shows an example of the test of two policies with the configuration of Rule Combination Algorithm and policy Enforcement Algorithm. From the testing results, you can verify if the effectiveness is intentional or not for meeting the access control security requirements. If there are any errors in any test cases, the policy author needs to revise the policy rule or algorithm to fix the error. Tests can be conducted with the revised access control security model.

2.3 Policy Analysis, Access Control Flaw Inspection and Correction

Security Policy Tool presents the testing results to you by tables for further analysis, inspection, and correction. Tests are repeatable for you to analyze the effectiveness of different rules, policies, and algorithms. Analysis and inspection can be explained as the process for you to compare the testing effectiveness ($FALSE$ or $TRUE$) with the intended one in an access control system. Error occurs when the testing effectiveness is not matched to the intended one. With the analysis of the testing results, you can further update and revise the access control security models until the access control results are satisfied for the intended access control

Error Types	Error Name	Error Description
Error type 1	Block Privilege	Suppose you know A should be able to access resource B. However, A is unable to access the resource B.
Error type 2	Leak Privilege	Suppose you know A shouldn't be able to access B. However, A is able to access B.
Error type 3	Not Protected Resource	This is an error that a Resource (e.g., B) is not protected by any rules and policies. For example, a document is not covered by any policy.
Error type 4	Rule Conflict	This is an error that two or more rules are conflicted in a policy, e.g., a rule allows access while the other rule declines access.
Error type 5	Inconsistent Assignment	Suppose a policy author edits a number of XACML policy documents separately. Attributes, conditions, rules or other policy variables/values could be inconsistently assigned in different policies. For example, Attribute Nurse could be inconsistently termed as, Hospital Nurse, in different policy documents. Integrity verification is necessary for detecting this type of error.
Error type 6	Privilege Inheritance Loop	A policy author may specify an error inheritance relation. An access control inheritance loop is caused by recursive and subsequent privilege inheritance, e.g., $A \rightarrow B \rightarrow C \rightarrow A$. It is able to automatically detect and prevent an inheritance loop.
Error type 7	Undecided Rules	Undecided rule error is that a Resource is unassigned with necessary actions in the workflow access control model. For example, a purchase order is assigned for a person to create but there is no rule assigned to approve the order. The workflow is unable to move forward due to this error.
Error type 8	Separation of Duty Error	This is an error that two or more rules cause competing interests among subjects, resources, or actions. For example, a person with Role A and Role B is a conflict for accessing a resource.

TABLE 1: **Access Control Error Types: All these access control errors could occur in an access control system**

security. During the test and analyzing, you are allowed to verify:

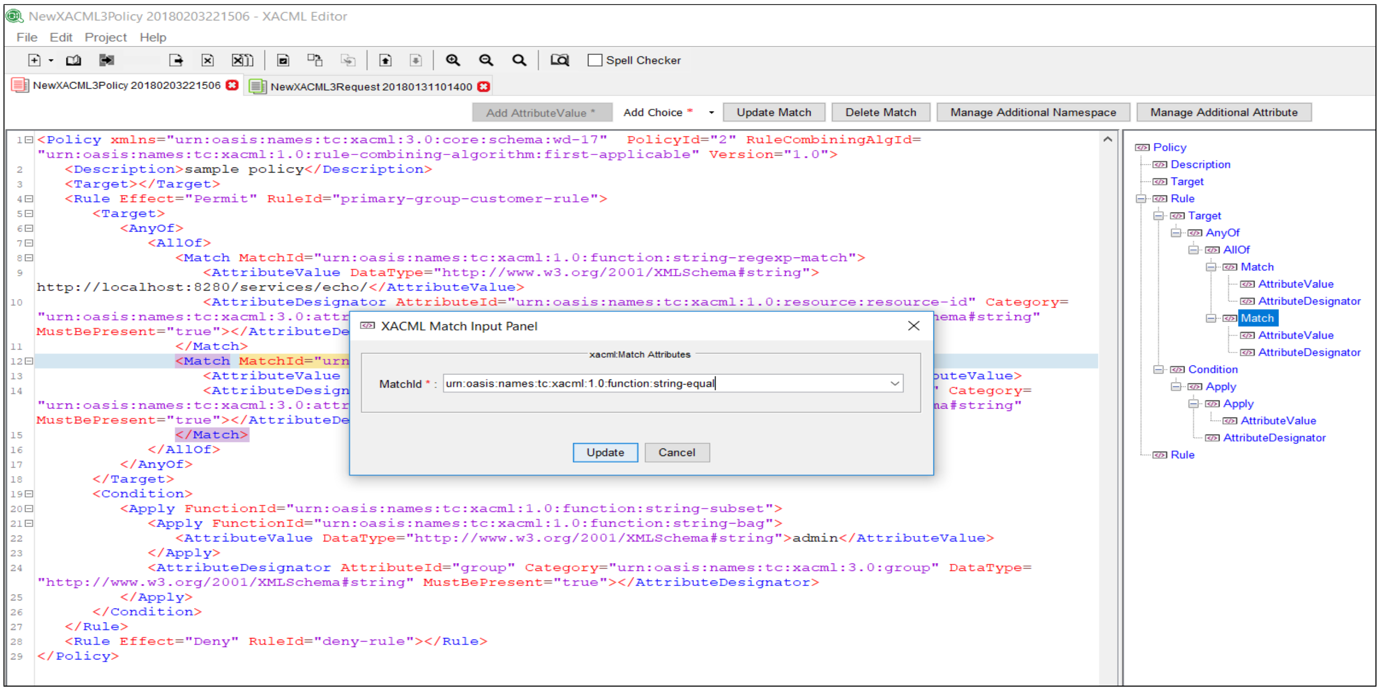
1) Individual security requirements vs. actual policy rules, which is to check the effectiveness of each rule in a policy or policyset.
2) Separation of Duty to verify no conflicts of interest among rules, which is to inspect the access control error of Separation of Duty in an access control security model.
3) Resource accessibility for a given Subject and the rule-combining algorithms, which allows you to detect if any unprotected resource.
4) Subject accessibility for a given Resource and the rule-combining algorithms, which allows you to review the accessibility of a given subject (e.g., a doctor) again all resources.

Security Policy Tool has powerful analyzing functions to help policy authors to inspect and correct the access control errors. Table 1 shows eight types of access control errors that cover all the current types of error discovered by NIST [3]. Understanding these errors help you to fully test and analyze your access control security model.

2.4 XACML Editor, Checker, and Converter

Security Policy Tool includes an XACML 3.0/2.0 policy editor that provide powerful policy editing, syntax check, and inputting/outputting functions. The XACML editor included in Security Policy Tool is **free** and is the most user-friendly and powerful tool on the market. OASIS XACML specification (1.0/2.0/3.0) defines a number of XML elements. These elements are very verbose in order to provide flexibility and on the other hand, it imposes a high usage threshold for users, especially for those who are not familiar with it or other XML based languages. Writing these policies could be difficult (e.g., errors, typos, syntax mistakes) for complex policies. Security Policy Tool offers a most comprehensive XACML editor that supports all elements for both XACML 2.0 and 3.0 specifications. Security Policy Tool offers XACML editing in two ways:

1) **GUI-based Composition**: This allows you to compose an XACML element by a GUI-based interface. The GUI-based interface hides the complexity of the



```
<Policy xmlns="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17"  PolicyId="2" RuleCombiningAlgId=
"urn:oasis:names:tc:xacml:1.0:rule-combining-algorithm:first-applicable" Version="1.0">
   <Description>sample policy</Description>
   <Target></Target>
   <Rule Effect="Permit" RuleId="primary-group-customer-rule">
      <Target>
         <AnyOf>
            <AllOf>
               <Match MatchId="urn:oasis:names:tc:xacml:1.0:function:string-regexp-match">
                  <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
http://localhost:8280/services/echo/</AttributeValue>
                  <AttributeDesignator AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id" Category=
"urn:oasis:names:tc:xacml:3.0:attr                                          hema#string"
MustBePresent="true"></AttributeDe
               </Match>
               <Match MatchId="urn
                  <AttributeValue                                             buteValue>
                  <AttributeDesign                                            " Category=
"urn:oasis:names:tc:xacml:3.0:attr                                          ma#string"
MustBePresent="true"></AttributeDe
               </Match>
            </AllOf>
         </AnyOf>
      </Target>
      <Condition>
         <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-subset">
            <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-bag">
               <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">admin</AttributeValue>
            </Apply>
            <AttributeDesignator AttributeId="group" Category="urn:oasis:names:tc:xacml:3.0:group" DataType=
"http://www.w3.org/2001/XMLSchema#string" MustBePresent="true"></AttributeDesignator>
         </Apply>
      </Condition>
   </Rule>
   <Rule Effect="Deny" RuleId="deny-rule"></Rule>
</Policy>
```

MatchId * : urn:oasis:names:tc:xacml:1.0:function:string-equal

Fig. 6. XACML editor offers both graphic and text-based XACML editing functions to minimize tedious XACML statement TYPING: A tree structure shows you the policy structure; It indicates the potential editing functions for a given XACML element; Syntax checking indicates errors and typos.

XACML statements. From the GUI, an XACML statement can be composed by choosing the elements from the indicated options with an automatic matching.

2) **Text-based Editing**: It allows you to manually editing (adding, deleting, updating) XACML elements during policy editing. Considering the XACML complexity, Security Policy Tool adds intelligence to help you to easily and quickly edit the XACML statement.

GUI-based and Text-based XACML editing methods should be both utilized to save your efforts and ensure the correctness of the XACML statement. Figure 6 demonstrates the XACML editor that has the following user-friendly features:

1) **Access Control Security Model Translation**: All the policies in an access control security model defined in the Security Policy Tool can be automatically translated to XACML 3.0 format, just by one click. This saves tremendous of labor cost to build an access control security model and makes it executable into an access control system.
2) **XACML 2.0, 3.0, Policyset, Policy, Request**: It supports all these elements in the legacy and state-of-art XACML specifications. Further, it can automatically recognize the XACML versions.
3) **Editing Intelligence**: Intelligence follows the XACML standard to instruct you to edit XACML statement in the next step while minimizing your TYPING of XACML elements.
4) **XACML Syntax Checker and Indicator**: Syntax checker is performed on the background and it indicates any syntax errors at which statement in real-time.
5) **XACML Structure**: A tree architecture shows your composed XACML.
6) **Multiple XACML Files**: It allows you to create, edit, and manage multiple XACML files in one project.
7) **Import, Export, and Management**: Multiple XACML documents can be imported, exported, and managed at the same time.

The above functions make the XACML editing easily, simply, correctly, and enjoyable. The

XACML editor in the Security Policy Tool follows the current popular design with line indicator to provide the flexibility during editing.

3 Versions

Security Policy Tool software currently provides Windows and Mac versions. Please visit www.Securitypolicytool.com to register and download a version for free. Linux version will be available soon.

4 Usability and Feedback

Security Policy Tool is considered to be used for security specialists as a tool for:

1) **Build a Robust Access Control System**: Security Policy Tool helps you to build a robust access control system with rigorous tests and verification to achieve high-security confidence.
2) **Verify if Errors in the Existing System**: If your access control system was not tested and verified before, you should apply the access control configuration to Security Policy Too in order to verify if there are security vulnerabilities caused by access control errors.
3) **Verify an Access Control System under New Configuration**: When your access control system is reconfigured (e.g., new attributes or other modifications), you are strongly recommended to use Security Policy Tool to verify if security vulnerabilities are introduced on the new configuration.

Security Policy Tool has been quickly recognized by IT industry, BFSI (Banking, Financial Services, and Insurance), healthcare, government, and other organizations all the world. In addition, IT consultant companies use it to ensure the security confidence of their customers. NIST states that: "*SPT (Security Policy Tool) has rich policy analysis functions such that the policy author can use them to user-friendly analyze if there are access control leaks and then fix these leaks caused by unintended or faulty security policies. It offers Subject/Resource Privilege Access Preview functions to find unintended accessibility, such as: (i) who has the accessibility for a giving resource, and (ii) what resource can access for a given subject. All these functions help a policy author to identify and correct AC flaws, such as block privilege, leak privilege, unprotected objects, Separation of Duty error, etc.*".

Acknowledgments

Thanks Dr. Vincent Hu and the NIST team for all their tremendous efforts that helps us during the product development. Appreciations are also expressed to users who gave us feedback to improve the design of the Security Policy Tool.

References

[1] Vincent C. Hu, Rick Kuhn, Dylan Yaga, "NIST SP 800-192: Verification and Test Methods for Access Control Policies/Models", Computer Security Division, National Institute of Standards and Technology, 2017, Alliable at: https://beta.csrc.nist.gov/News/2017/NIST-Release-SP-800-192.

[2] Vincent C. Hu, and Karen Scarfone "NIST IR 800-7874: Guidelines for Access Control System Evaluation Metrics", Computer Security Division, National Institute of Standards and Technology, 2012, Alliable at: http://nvlpubs.nist.gov/nistpubs/ir/2012/NIST.IR.7874.pdf.

[3] Vincent C. Hu, David Ferraiolo, Rick Kuhn, Adam Schnitzer, Kenneth Sandlin, Robert Miller, Karen Scarfone "NIST SP 800-162: Guide to Attribute Based Access Control (ABAC) Definition and Considerationss", Computer Security Division, National Institute of Standards and Technology, 2014, Alliable at: http://nvlpubs.nist.gov/nistpubs/specialpublications/NIST.SP.800-162.pdf.

[4] Vincent C. Hu, D. F. Ferraiolo, and D. R. Kuhn, "Assessment of Access Control Systems," NIST IR 7316, Computer Security Division, National Institute of Standards and Technology, 2006.

InfoBeyond Technology, LLC is an innovative company specializing in Network, Machine Learning and Data Security within the Information Technology industry. The mission of InfoBeyond is to research, develop, and deliver viable software products for network communication and security. Some of our research is sponsored by Department of Defense, Department of Energy, Missile Defense Agency, Department of Transportation, NIST (National Institute of Standards and Technology), etc. Security Policy Tool is Awarded the 2017 Innovative Security Solution Award at the 2017 Big Data and SDN/NFV Summit. NXdrive is a fragment-based cybersecurity storage system and more information can be found at www.NXdrive.com. The company is featured as one of 50 fast growth IT small businesses in 2017 by The Silicon Review.