Continue Policy Test Cases

(InfoBeyond Technology LLC)

Abstract

This document demonstrates access control test cases using Security Policy Tool, a software tool for Access Control Security Managers, Policy Authors, and other IT Security Professionals specializing in the performance of access control systems. Access control policies are designed to protect the accessibility of online resources in networks, IoTs, healthcare systems, financial service systems, enterprise IT and clouds, military systems, and other online environments. There are several challenges in building robust access control models for these systems including (i) effectively composing secure policies and rules, (ii) testing these policies systematically, (iii) verifying these policies to prevent access control leaks. Security Policy Tool solves these issues by providing powerful access control policy modeling, testing, and verification features that empower organizations to close the door to access control leaks.

Index Terms

Access control, attribute-based access control (abac), role-based access control (rbac), security policy editing, test, verification, deployment, access control leaks, XACML, software tool.

1 Introduction to Test Cases

This document and linked Security Policy Tool– Project Files have been designed to help you gain an understanding of what common access control errors look like, how they are created, and how to resolve them. Organizations who leverage Security Policy Tool's systematic modeling, testing and verification features are empowered to efficiently identify errors and close the door to access control leaks.

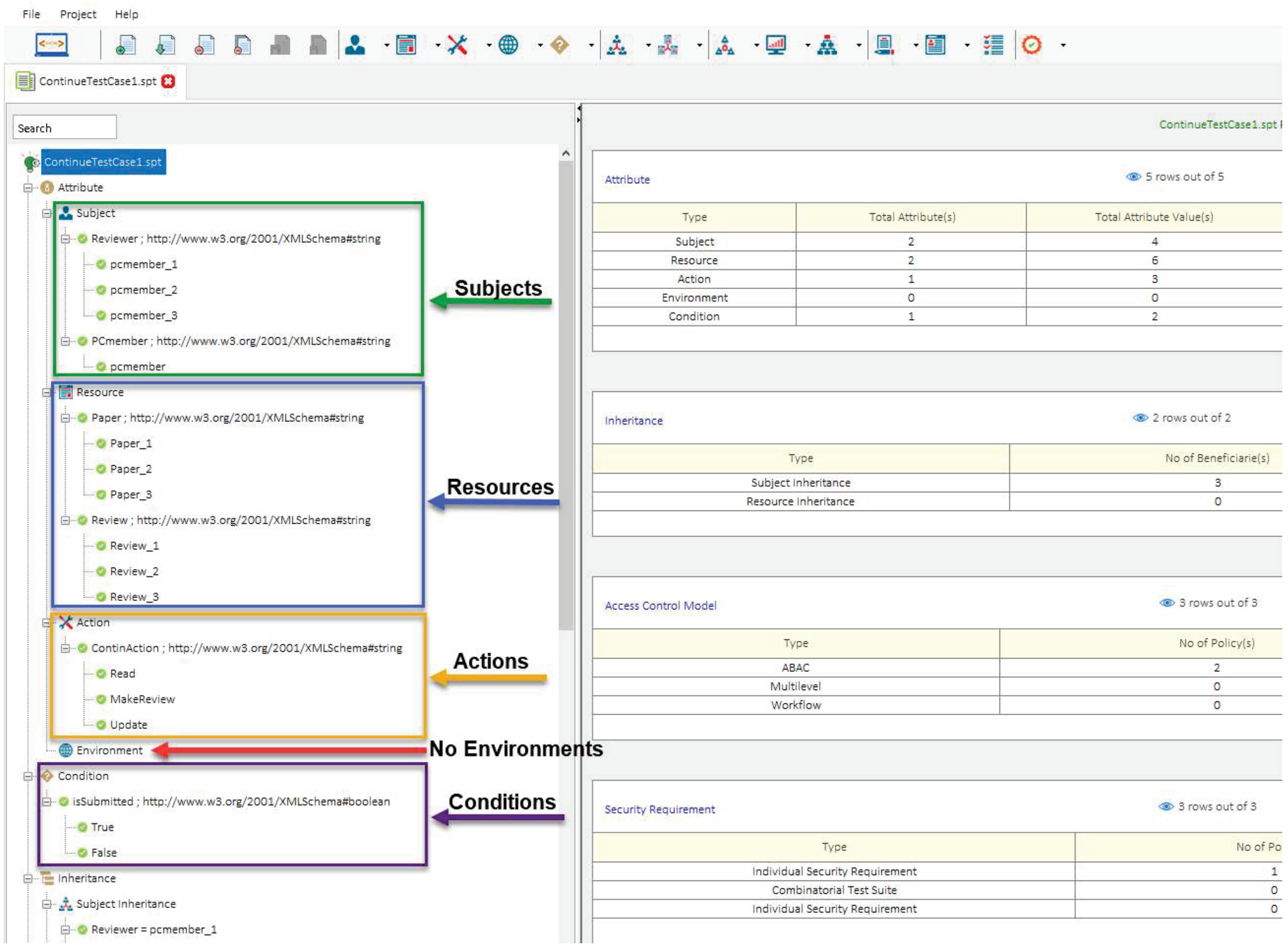
These Continue Policy test cases are based on examples previously created by the National Institute of Standards & Technology (NIST) to demonstrate commonly found errors in access control policy logic similarly. This policy is used for a web-based software called "Continue" which provides tools for organizing processes required to finalize conference papers (e.g., submission, review, discussion, etc.). These test cases consist of policies/rules from NIST's example as well as modifications to better illustrate how Security Policy Tool enhances access control security. The goal of these test cases is to provide a starting point for what to expect as you go on to use Security Policy Tool to analyze your own policy verification results for errors.

2 Setting Up The Policies – Test Case 1 (Rule Conflict)

This continue example contains two policies (PCmember Policy & Reviewer Policy). The Attribute/Attribute Values included in these policies are as shown in Figure 1.

• *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). With tremendous consultation with NIST experts, Security Policy Tool substantially enhances and expands the NIST's 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 NIST's ACPT design to provide a robust, unified, professional, and functionally powerful access control policy tool.



```
ContinueTestCase1.spt
  Attribute
    Subject
      Reviewer ; http://www.w3.org/2001/XMLSchema#string
        pcmember_1
        pcmember_2
        pcmember_3
      PCmember ; http://www.w3.org/2001/XMLSchema#string
        pcmember
    Resource
      Paper ; http://www.w3.org/2001/XMLSchema#string
        Paper_1
        Paper_2
        Paper_3
      Review ; http://www.w3.org/2001/XMLSchema#string
        Review_1
        Review_2
        Review_3
    Action
      ContinAction ; http://www.w3.org/2001/XMLSchema#string
        Read
        MakeReview
        Update
    Environment
  Condition
    isSubmitted ; http://www.w3.org/2001/XMLSchema#boolean
      True
      False
  Inheritance
    Subject Inheritance
      Reviewer = pcmember_1
```

ContinueTestCase1.spt

Attribute — 5 rows out of 5

Type	Total Attribute(s)	Total Attribute Value(s)
Subject	2	4
Resource	2	6
Action	1	3
Environment	0	0
Condition	1	2

Inheritance — 2 rows out of 2

Type	No of Beneficiarie(s)
Subject Inheritance	3
Resource Inheritance	0

Access Control Model — 3 rows out of 3

Type	No of Policy(s)
ABAC	2
Multilevel	0
Workflow	0

Security Requirement — 3 rows out of 3

Type	No of Po
Individual Security Requirement	1
Combinatorial Test Suite	0
Individual Security Requirement	0

Fig. 1. Test Case 1

3 Subject Inheritance – Test Case 1 (Rule Conflict)

Depending on your security needs or organizational makeup you may decide to define Inheritance relationships to help you generate policy Rules more quickly. For this continue policy example we will define (3) Subject Inheritance Relationships as follows:

Subject Inheritance:

Beneficiary Values → Reviewer (pcmember_1, pcmember_2, pcmember_3)

Inherited Values → PCmember

If we have created these Relationships correctly based on the above direction it will look like this in Security Policy Tool:

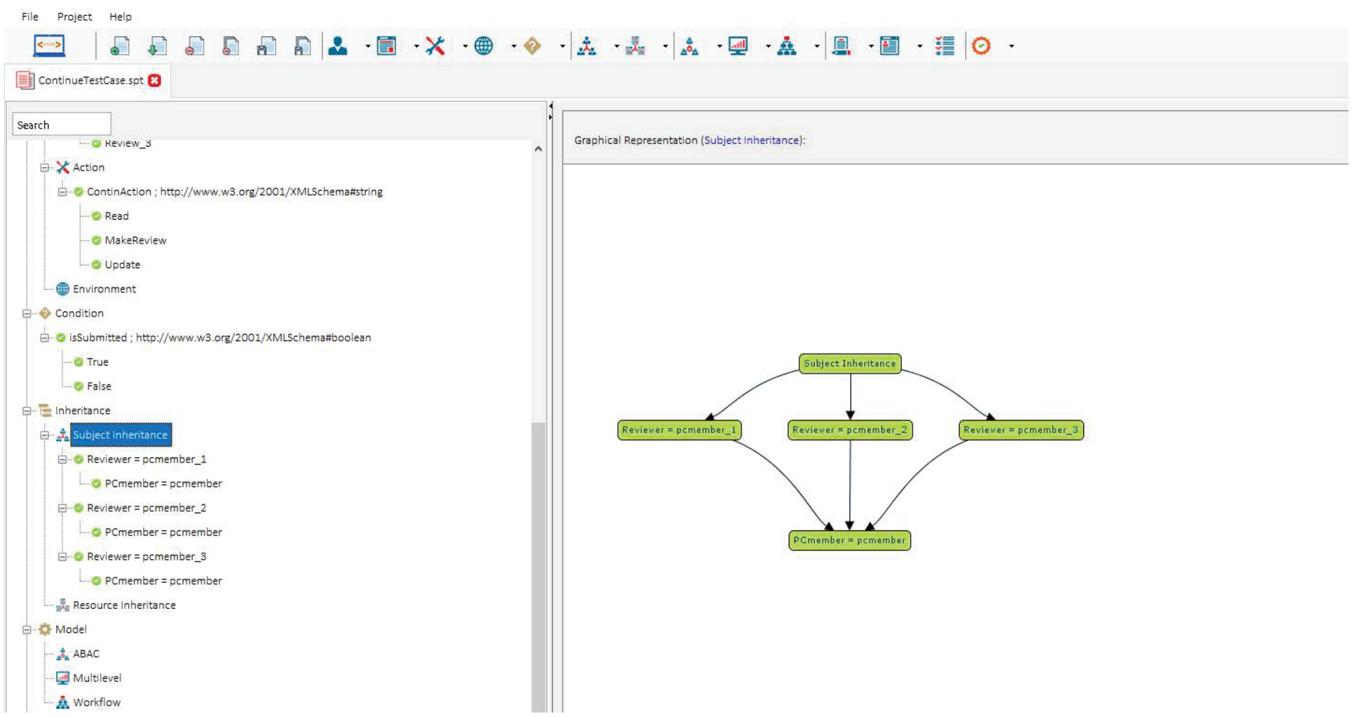


Fig. 2. Subject Inheritance Relationships

By defining these relationships, any (Originated) Rules with Decision = Permit given to PCmember = pcmember will now also be given to pcmember_1, pcmember_2, and pcmember_3 as (Inherited) Rules in our policies. Originated Rules with Decision = Deny are never Inherited. This is because typically Beneficiaries in these relationships are higher ranking/senior roles that by nature will have less restrictions (e.g., denying access) than roles that are providing the Inheritance Value (e.g., pcmember in our example).

Hence, it will authorize the Beneficiaries to obtain all privileges of Inherited Values (e.g., generally lower-level roles) while not obtaining their typically tighter restrictions. If you would like Beneficiaries to be Denied access similar to their Inherited Value you can still do so by manually creating individual rules when you begin modeling.

4 Modeling Your Policy – Test Case 1 (Rule Conflict)

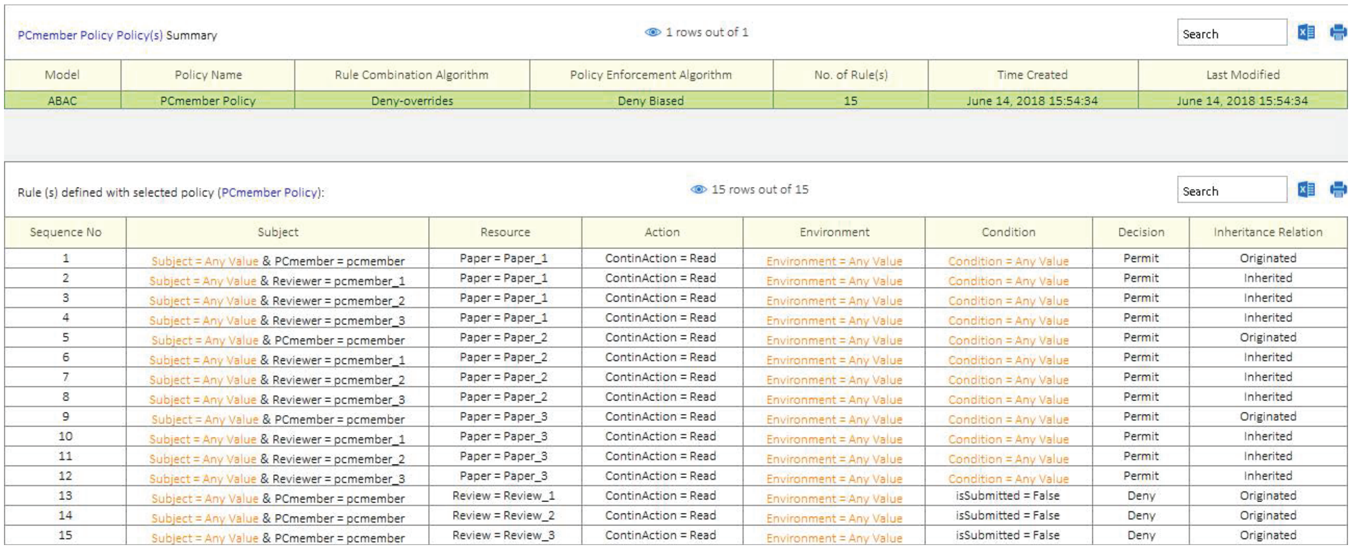
Now that we have entered our attributes we can model our two policies (PCmember Policy & Reviewer Policy). See the list below of the rules contained in each of these policies. You can open a "New (blank) Project" and build these policies by entering the following rules below:

PCmember Policy:
(Subject = Any Value & PCmember = pcmember, Read, Paper_1) → Permit
(Subject = Any Value & PCmember = pcmember, Read, Paper_2) →Permit
(Subject = Any Value & PCmember = pcmember, Read, Paper_3) →Permit
(Subject = Any Value & PCmember = pcmember, isSubmitted: False, Read, Review_1) →Deny
(Subject = Any Value & PCmember = pcmember, isSubmitted: False, Read, Review_2) →Deny
(Subject = Any Value & PCmember = pcmember, isSubmitted: False, Read, Review_3) →Deny
NOTE: You will also generate 9 additional "'Inherited Rules" due to the first 3 Permit Rules

Reviewer Policy:
(Subject = Any Value & Reviewer = pcmember_1, isSubmitted: False, Update, Review_1) →Permit
(Subject = Any Value & Reviewer = pcmember_2, isSubmitted: False, Update, Review_2) →Permit
(Subject = Any Value & Reviewer = pcmember_3, isSubmitted: False, Update, Review_3) →Permit
(Subject = Any Value & Reviewer = pcmember_3, isSubmitted: False, Read, Review_3) →Permit
(Subject = Any Value & Reviewer = pcmember_1, isSubmitted: False, MakeReview, Paper_1) →Permit

(Subject = Any Value & Reviewer = pcmember_2, isSubmitted: False, MakeReview, Paper_2) →Permit
(Subject = Any Value & Reviewer = pcmember_3, isSubmitted: False, MakeReview, Paper_3) →Permit

After entering the rules above your modeled policies should look like the screenshots below. If you did not create your own Project File, you can simply open Security Policy Tool – Project File: ContinueTestCase1 and these policies will have been already created for you.



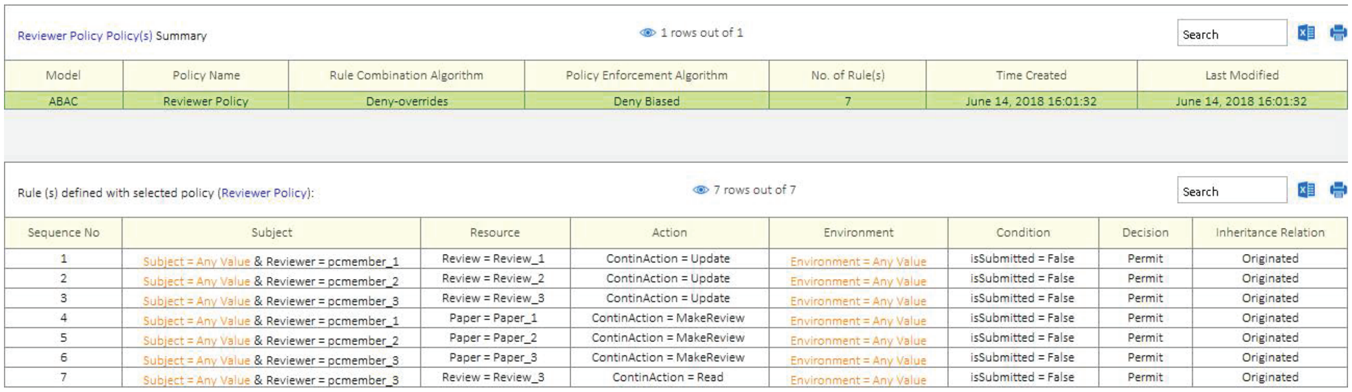
PCmember Policy Policy(s) Summary — 1 rows out of 1

Model	Policy Name	Rule Combination Algorithm	Policy Enforcement Algorithm	No. of Rule(s)	Time Created	Last Modified
ABAC	PCmember Policy	Deny-overrides	Deny Biased	15	June 14, 2018 15:54:34	June 14, 2018 15:54:34

Rule (s) defined with selected policy (PCmember Policy): — 15 rows out of 15

Sequence No	Subject	Resource	Action	Environment	Condition	Decision	Inheritance Relation
1	Subject = Any Value & PCmember = pcmember	Paper = Paper_1	ContinAction = Read	Environment = Any Value	Condition = Any Value	Permit	Originated
2	Subject = Any Value & Reviewer = pcmember_1	Paper = Paper_1	ContinAction = Read	Environment = Any Value	Condition = Any Value	Permit	Inherited
3	Subject = Any Value & Reviewer = pcmember_2	Paper = Paper_1	ContinAction = Read	Environment = Any Value	Condition = Any Value	Permit	Inherited
4	Subject = Any Value & Reviewer = pcmember_3	Paper = Paper_1	ContinAction = Read	Environment = Any Value	Condition = Any Value	Permit	Inherited
5	Subject = Any Value & PCmember = pcmember	Paper = Paper_2	ContinAction = Read	Environment = Any Value	Condition = Any Value	Permit	Originated
6	Subject = Any Value & Reviewer = pcmember_1	Paper = Paper_2	ContinAction = Read	Environment = Any Value	Condition = Any Value	Permit	Inherited
7	Subject = Any Value & Reviewer = pcmember_2	Paper = Paper_2	ContinAction = Read	Environment = Any Value	Condition = Any Value	Permit	Inherited
8	Subject = Any Value & Reviewer = pcmember_3	Paper = Paper_2	ContinAction = Read	Environment = Any Value	Condition = Any Value	Permit	Inherited
9	Subject = Any Value & PCmember = pcmember	Paper = Paper_3	ContinAction = Read	Environment = Any Value	Condition = Any Value	Permit	Originated
10	Subject = Any Value & Reviewer = pcmember_1	Paper = Paper_3	ContinAction = Read	Environment = Any Value	Condition = Any Value	Permit	Inherited
11	Subject = Any Value & Reviewer = pcmember_2	Paper = Paper_3	ContinAction = Read	Environment = Any Value	Condition = Any Value	Permit	Inherited
12	Subject = Any Value & Reviewer = pcmember_3	Paper = Paper_3	ContinAction = Read	Environment = Any Value	Condition = Any Value	Permit	Inherited
13	Subject = Any Value & PCmember = pcmember	Review = Review_1	ContinAction = Read	Environment = Any Value	isSubmitted = False	Deny	Originated
14	Subject = Any Value & PCmember = pcmember	Review = Review_2	ContinAction = Read	Environment = Any Value	isSubmitted = False	Deny	Originated
15	Subject = Any Value & PCmember = pcmember	Review = Review_3	ContinAction = Read	Environment = Any Value	isSubmitted = False	Deny	Originated

Fig. 3. PCmember Policy



Reviewer Policy Policy(s) Summary — 1 rows out of 1

Model	Policy Name	Rule Combination Algorithm	Policy Enforcement Algorithm	No. of Rule(s)	Time Created	Last Modified
ABAC	Reviewer Policy	Deny-overrides	Deny Biased	7	June 14, 2018 16:01:32	June 14, 2018 16:01:32

Rule (s) defined with selected policy (Reviewer Policy): — 7 rows out of 7

Sequence No	Subject	Resource	Action	Environment	Condition	Decision	Inheritance Relation
1	Subject = Any Value & Reviewer = pcmember_1	Review = Review_1	ContinAction = Update	Environment = Any Value	isSubmitted = False	Permit	Originated
2	Subject = Any Value & Reviewer = pcmember_2	Review = Review_2	ContinAction = Update	Environment = Any Value	isSubmitted = False	Permit	Originated
3	Subject = Any Value & Reviewer = pcmember_3	Review = Review_3	ContinAction = Update	Environment = Any Value	isSubmitted = False	Permit	Originated
4	Subject = Any Value & Reviewer = pcmember_1	Paper = Paper_1	ContinAction = MakeReview	Environment = Any Value	isSubmitted = False	Permit	Originated
5	Subject = Any Value & Reviewer = pcmember_2	Paper = Paper_2	ContinAction = MakeReview	Environment = Any Value	isSubmitted = False	Permit	Originated
6	Subject = Any Value & Reviewer = pcmember_3	Paper = Paper_3	ContinAction = MakeReview	Environment = Any Value	isSubmitted = False	Permit	Originated
7	Subject = Any Value & Reviewer = pcmember_3	Review = Review_3	ContinAction = Read	Environment = Any Value	isSubmitted = False	Permit	Originated

Fig. 4. Reviewer Policy

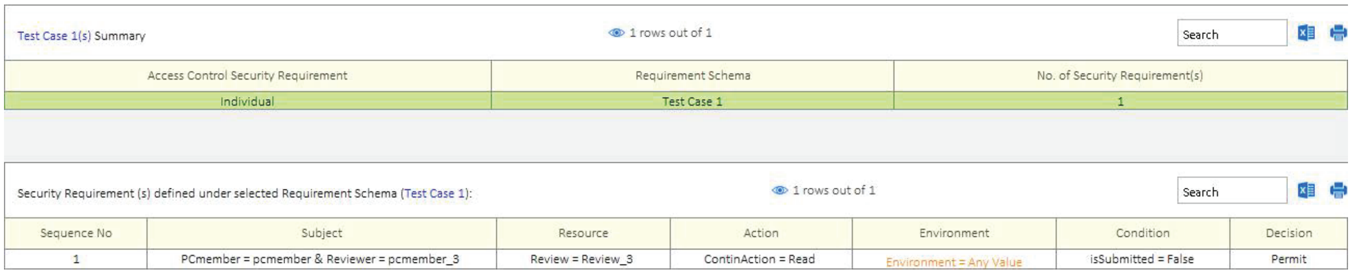
5 Individual Security Requirements - Test Case 1 (Rule Conflict)

The final step before analyzing these policies for errors is to create individual security requirements to use for testing. If you are building a "New (blank) Project" on your own you will enter the following security requirement below:

Individual Security Requirements:

(Pcmember = pcmember & Reviewer = pcmember_3) & (isSubmitted = False) & (Action = Read) & (Review = Review_3) → decision = Permit

After entering the rule above your individual security requirement should look like the screenshot below. If you did not create your own Project File you can simply open Security Policy Tool – Project File: ContinueTestCase1 and this requirement will have been already created for you.



Test Case 1(s) Summary — 1 rows out of 1 — Search

Access Control Security Requirement	Requirement Schema	No. of Security Requirement(s)
Individual	Test Case 1	1

Security Requirement (s) defined under selected Requirement Schema (Test Case 1): — 1 rows out of 1 — Search

Sequence No	Subject	Resource	Action	Environment	Condition	Decision
1	PCmember = pcmember & Reviewer = pcmember_3	Review = Review_3	ContinAction = Read	Environment = Any Value	isSubmitted = False	Permit

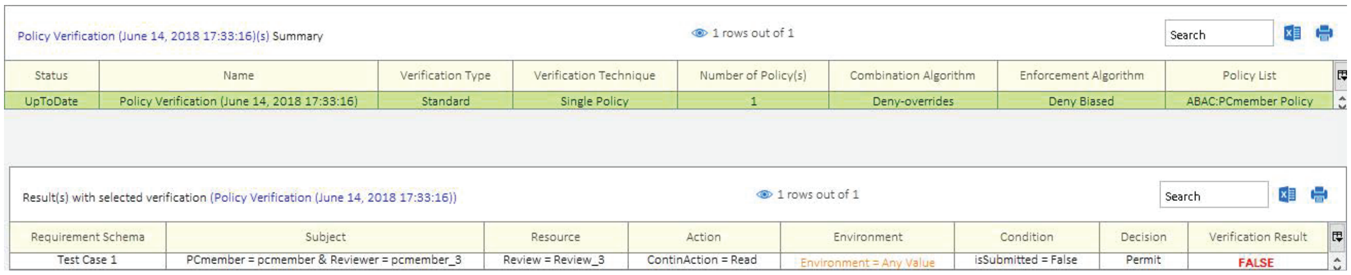
Fig. 5. Individual Security Requirements

6 Policy Verification/Analyzing Results - Test Case 1 (Rule Conflict)

Now that we are ready to test our policies let's discuss the error we will be looking at in this first example. When policies are designed, there is potential for a "Rule Conflict" being created. A Rule Conflict occurs when two or more rules are defining opposite authorization in an access control policy.

In our example, an individual using this "'Continue"' software has a role of both PCmember and Reviewer in the system. Due to this, the individual is assigned both (PCmember: pcmember and Reviewer: pcmember_3) attribute values by the system during access evaluation. In the PCmember Policy it defines that pcmembers cannot read Review_3. However, in the Reviewer Policy it defines pcmember_3 can read Review_3 resulting in a Rule Conflict (e.g., Permitted to Read in Reviewer Policy; Denied to Read in PCmember Policy).

Next, we will run two "Single Policy" Verifications to reveal the Rule Conflict that is present in our policies. To do this, we will select PCmember Policy and Test Case 1 (security requirement) as a Single Policy Verification and also choose Reviewer Policy and Test Case 1 (security requirement) as a Single Policy Verification and analyze our two verification results. Again, this will have already been done for you if you open Project File: ContinueTestCase1.



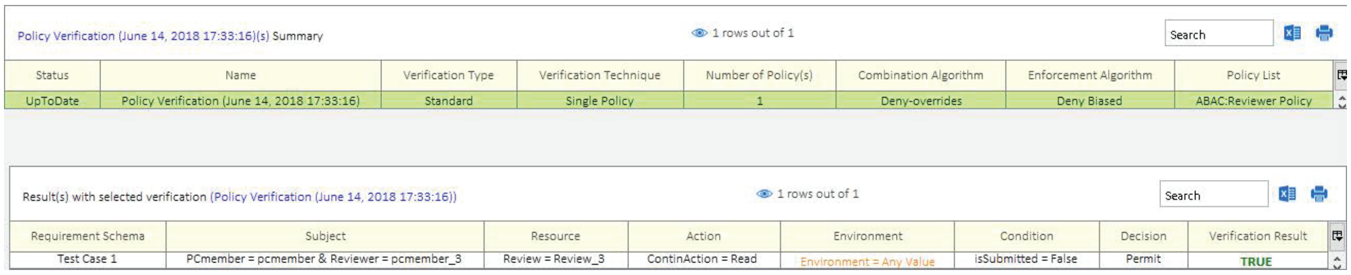
Policy Verification (June 14, 2018 17:33:16)(s) Summary — 1 rows out of 1 — Search

Status	Name	Verification Type	Verification Technique	Number of Policy(s)	Combination Algorithm	Enforcement Algorithm	Policy List
UpToDate	Policy Verification (June 14, 2018 17:33:16)	Standard	Single Policy	1	Deny-overrides	Deny Biased	ABAC:PCmember Policy

Result(s) with selected verification (Policy Verification (June 14, 2018 17:33:16)) — 1 rows out of 1 — Search

Requirement Schema	Subject	Resource	Action	Environment	Condition	Decision	Verification Result
Test Case 1	PCmember = pcmember & Reviewer = pcmember_3	Review = Review_3	ContinAction = Read	Environment = Any Value	isSubmitted = False	Permit	FALSE

Fig. 6. PCmember Policy x Test Case 1



Policy Verification (June 14, 2018 17:33:16)(s) Summary — 1 rows out of 1 — Search

Status	Name	Verification Type	Verification Technique	Number of Policy(s)	Combination Algorithm	Enforcement Algorithm	Policy List
UpToDate	Policy Verification (June 14, 2018 17:33:16)	Standard	Single Policy	1	Deny-overrides	Deny Biased	ABAC:Reviewer Policy

Result(s) with selected verification (Policy Verification (June 14, 2018 17:33:16)) — 1 rows out of 1 — Search

Requirement Schema	Subject	Resource	Action	Environment	Condition	Decision	Verification Result
Test Case 1	PCmember = pcmember & Reviewer = pcmember_3	Review = Review_3	ContinAction = Read	Environment = Any Value	isSubmitted = False	Permit	TRUE

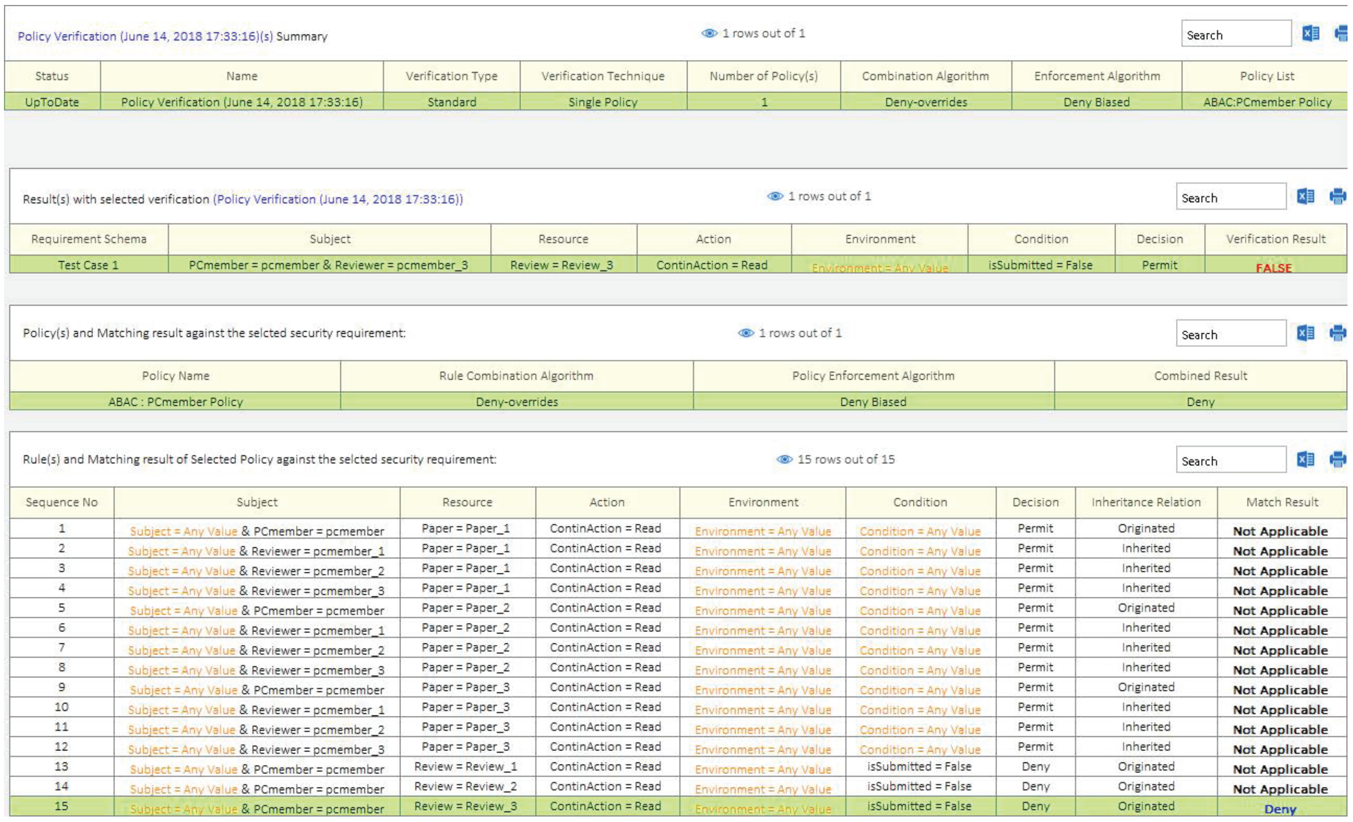
Fig. 7. Reviewer Policy x Test Case 1

As you can see from our verification results our policies are both Permitting and Denying the individual (PCmember = pcmember/Reviewer = pcmember_3) from reading Unsubmitted resource Review_3 which is known as a Rule Conflict error.

7 Resolving This Error - Test Case 1 (Rule Conflict)

To solve a Rule Conflict the policy author would need to go back and either update or delete the related rules to the error. To view which specific Rules are resulting in these Verification Results we can click on our 2 Results (PCmember PolicyxTestCase1: False & Reviewer PolicyxTestCase1: True) and see which Rules have "Match Results".

See the screen shots below of our two Policy's Match Results to discover which specific rules are related to our Verification Results (e.g., False, True).



Policy Verification (June 14, 2018 17:33:16)(s) Summary — 1 rows out of 1

Status	Name	Verification Type	Verification Technique	Number of Policy(s)	Combination Algorithm	Enforcement Algorithm	Policy List
UpToDate	Policy Verification (June 14, 2018 17:33:16)	Standard	Single Policy	1	Deny-overrides	Deny Biased	ABAC:PCmember Policy

Result(s) with selected verification (Policy Verification (June 14, 2018 17:33:16)) — 1 rows out of 1

Requirement Schema	Subject	Resource	Action	Environment	Condition	Decision	Verification Result
Test Case 1	PCmember = pcmember & Reviewer = pcmember_3	Review = Review_3	ContinAction = Read	Environment = Any Value	isSubmitted = False	Permit	FALSE

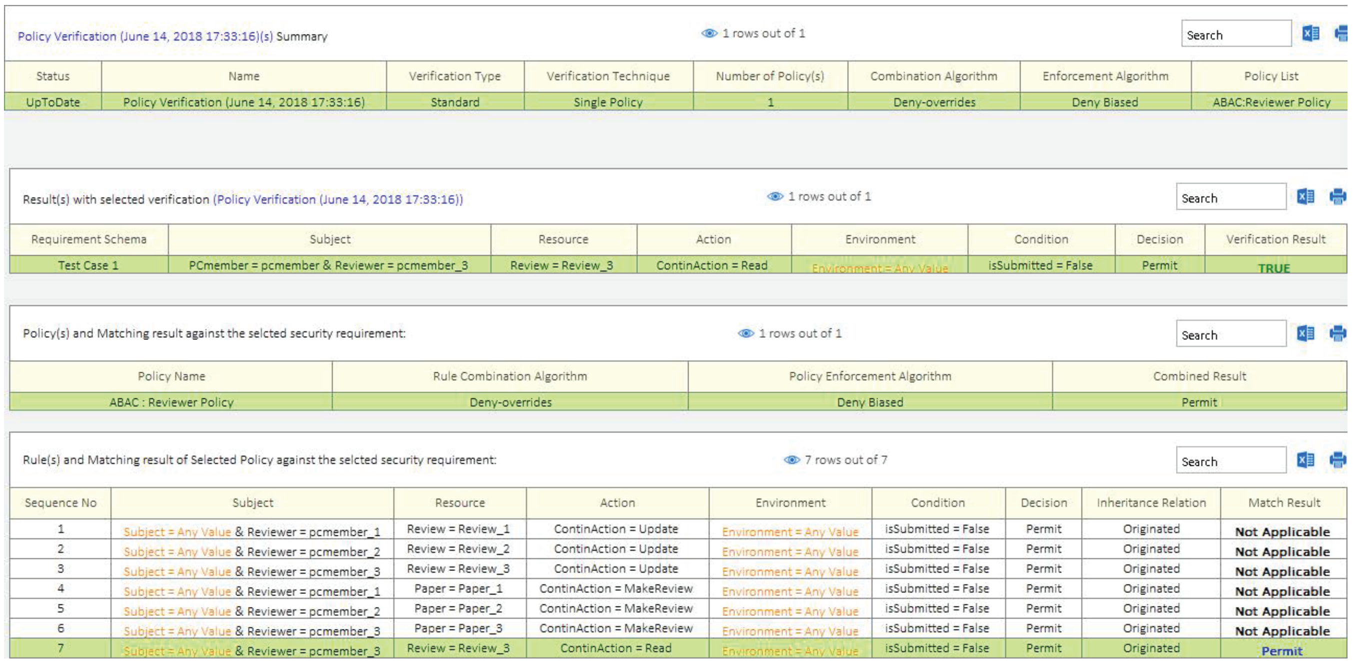
Policy(s) and Matching result against the selcted security requirement: — 1 rows out of 1

Policy Name	Rule Combination Algorithm	Policy Enforcement Algorithm	Combined Result
ABAC : PCmember Policy	Deny-overrides	Deny Biased	Deny

Rule(s) and Matching result of Selected Policy against the selcted security requirement: — 15 rows out of 15

Sequence No	Subject	Resource	Action	Environment	Condition	Decision	Inheritance Relation	Match Result
1	Subject = Any Value & PCmember = pcmember	Paper = Paper_1	ContinAction = Read	Environment = Any Value	Condition = Any Value	Permit	Originated	Not Applicable
2	Subject = Any Value & Reviewer = pcmember_1	Paper = Paper_1	ContinAction = Read	Environment = Any Value	Condition = Any Value	Permit	Inherited	Not Applicable
3	Subject = Any Value & Reviewer = pcmember_2	Paper = Paper_1	ContinAction = Read	Environment = Any Value	Condition = Any Value	Permit	Inherited	Not Applicable
4	Subject = Any Value & Reviewer = pcmember_3	Paper = Paper_1	ContinAction = Read	Environment = Any Value	Condition = Any Value	Permit	Inherited	Not Applicable
5	Subject = Any Value & PCmember = pcmember	Paper = Paper_2	ContinAction = Read	Environment = Any Value	Condition = Any Value	Permit	Originated	Not Applicable
6	Subject = Any Value & Reviewer = pcmember_1	Paper = Paper_2	ContinAction = Read	Environment = Any Value	Condition = Any Value	Permit	Inherited	Not Applicable
7	Subject = Any Value & Reviewer = pcmember_2	Paper = Paper_2	ContinAction = Read	Environment = Any Value	Condition = Any Value	Permit	Inherited	Not Applicable
8	Subject = Any Value & Reviewer = pcmember_3	Paper = Paper_2	ContinAction = Read	Environment = Any Value	Condition = Any Value	Permit	Inherited	Not Applicable
9	Subject = Any Value & PCmember = pcmember	Paper = Paper_3	ContinAction = Read	Environment = Any Value	Condition = Any Value	Permit	Originated	Not Applicable
10	Subject = Any Value & Reviewer = pcmember_1	Paper = Paper_3	ContinAction = Read	Environment = Any Value	Condition = Any Value	Permit	Inherited	Not Applicable
11	Subject = Any Value & Reviewer = pcmember_2	Paper = Paper_3	ContinAction = Read	Environment = Any Value	Condition = Any Value	Permit	Inherited	Not Applicable
12	Subject = Any Value & Reviewer = pcmember_3	Paper = Paper_3	ContinAction = Read	Environment = Any Value	Condition = Any Value	Permit	Inherited	Not Applicable
13	Subject = Any Value & PCmember = pcmember	Review = Review_1	ContinAction = Read	Environment = Any Value	isSubmitted = False	Deny	Originated	Not Applicable
14	Subject = Any Value & PCmember = pcmember	Review = Review_2	ContinAction = Read	Environment = Any Value	isSubmitted = False	Deny	Originated	Not Applicable
15	Subject = Any Value & PCmember = pcmember	Review = Review_3	ContinAction = Read	Environment = Any Value	isSubmitted = False	Deny	Originated	Deny

Fig. 8. PCmember Policy: Match Results



Policy Verification (June 14, 2018 17:33:16)(s) Summary — 1 rows out of 1

Status	Name	Verification Type	Verification Technique	Number of Policy(s)	Combination Algorithm	Enforcement Algorithm	Policy List
UpToDate	Policy Verification (June 14, 2018 17:33:16)	Standard	Single Policy	1	Deny-overrides	Deny Biased	ABAC:Reviewer Policy

Result(s) with selected verification (Policy Verification (June 14, 2018 17:33:16)) — 1 rows out of 1

Requirement Schema	Subject	Resource	Action	Environment	Condition	Decision	Verification Result
Test Case 1	PCmember = pcmember & Reviewer = pcmember_3	Review = Review_3	ContinAction = Read	Environment = Any Value	isSubmitted = False	Permit	TRUE

Policy(s) and Matching result against the selcted security requirement: — 1 rows out of 1

Policy Name	Rule Combination Algorithm	Policy Enforcement Algorithm	Combined Result
ABAC : Reviewer Policy	Deny-overrides	Deny Biased	Permit

Rule(s) and Matching result of Selected Policy against the selcted security requirement: — 7 rows out of 7

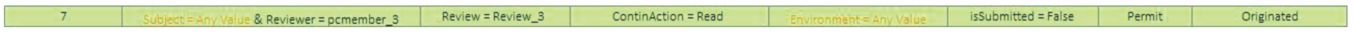
Sequence No	Subject	Resource	Action	Environment	Condition	Decision	Inheritance Relation	Match Result
1	Subject = Any Value & Reviewer = pcmember_1	Review = Review_1	ContinAction = Update	Environment = Any Value	isSubmitted = False	Permit	Originated	Not Applicable
2	Subject = Any Value & Reviewer = pcmember_2	Review = Review_2	ContinAction = Update	Environment = Any Value	isSubmitted = False	Permit	Originated	Not Applicable
3	Subject = Any Value & Reviewer = pcmember_3	Review = Review_3	ContinAction = Update	Environment = Any Value	isSubmitted = False	Permit	Originated	Not Applicable
4	Subject = Any Value & Reviewer = pcmember_1	Paper = Paper_1	ContinAction = MakeReview	Environment = Any Value	isSubmitted = False	Permit	Originated	Not Applicable
5	Subject = Any Value & Reviewer = pcmember_2	Paper = Paper_2	ContinAction = MakeReview	Environment = Any Value	isSubmitted = False	Permit	Originated	Not Applicable
6	Subject = Any Value & Reviewer = pcmember_3	Paper = Paper_3	ContinAction = MakeReview	Environment = Any Value	isSubmitted = False	Permit	Originated	Not Applicable
7	Subject = Any Value & Reviewer = pcmember_3	Review = Review_3	ContinAction = Read	Environment = Any Value	isSubmitted = False	Permit	Originated	Permit

Fig. 9. Reviewer Policy: Match Results

Now that we have pinpointed our (2) Rules related to our Rule Conflict Error we can go back and make changes or possibly remove these rules. Depending on your organizational structure the policy author or access control administrator would need to decide what is the most appropriate action to take to resolve the error. There is no “right” or “wrong” solution for this, you would need to determine what is most appropriate based on your organizational needs.

For our example, if we look at all other Rules in the Reviewer Policy no other Subjects are assigned to Read any included Resources. Thus, this individual assigned both (pcmember and pcmember_3) should not be allowed to Read Review_3. Similar to the rest of the Subjects in the Reviewer Policy they can still Update and MakeReview to Review_3 because it is required for their duties. To resolve this, we will simply delete this unneeded Rule 7 which will in turn resolve the Rule Conflict.

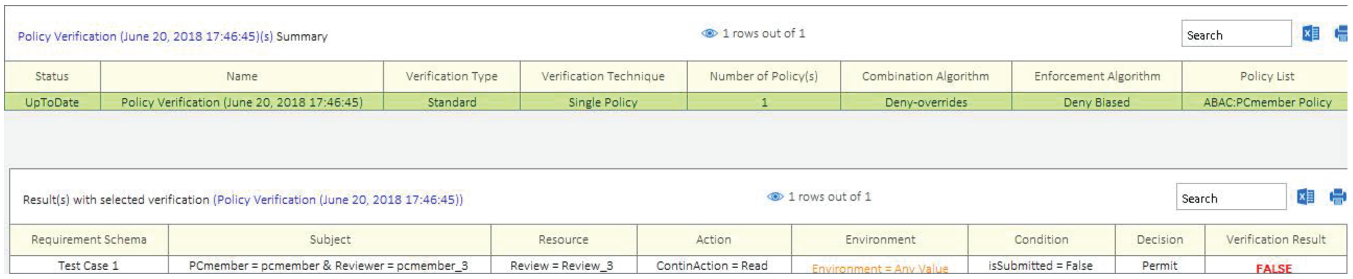
Reviewer Policy: Delete (1) Current Rule:
(Rule No. = 7) → (Subject = Any Value & Reviewer = pcmember_3) → (Action = Read) → (isSubmitted = False) → (Resource = Review_3) → decision = Permit



7	Subject = Any Value & Reviewer = pcmember_3	Review = Review_3	ContinAction = Read	Environment = Any Value	isSubmitted = False	Permit	Originated

Fig. 10. Reviewer Policy: Delete Rule (7)

After we “Refresh” our previous Verification Results we no longer have a Rule Conflict occurring. . . .



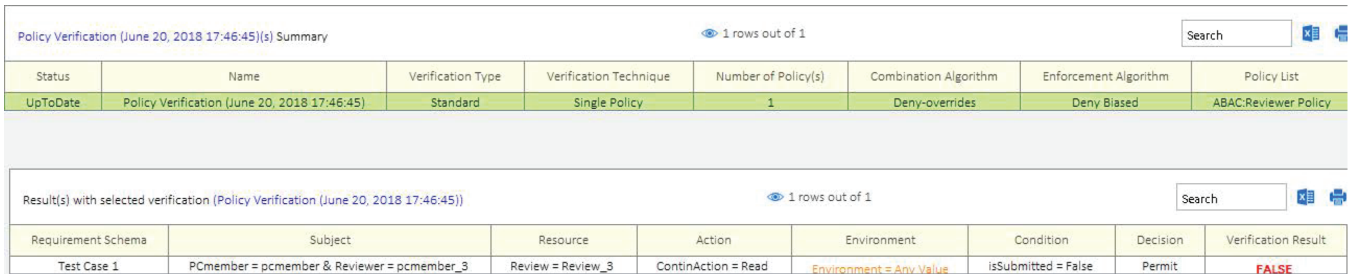
Policy Verification (June 20, 2018 17:46:45)(s) Summary — 1 rows out of 1 — Search

Status	Name	Verification Type	Verification Technique	Number of Policy(s)	Combination Algorithm	Enforcement Algorithm	Policy List
UpToDate	Policy Verification (June 20, 2018 17:46:45)	Standard	Single Policy	1	Deny-overrides	Deny Biased	ABAC:PCmember Policy

Result(s) with selected verification (Policy Verification (June 20, 2018 17:46:45)) — 1 rows out of 1 — Search

Requirement Schema	Subject	Resource	Action	Environment	Condition	Decision	Verification Result
Test Case 1	PCmember = pcmember & Reviewer = pcmember_3	Review = Review_3	ContinAction = Read	Environment = Any Value	isSubmitted = False	Permit	FALSE

Fig. 11. Updated Results: PCmember Policy (No Rule Conflict)



Policy Verification (June 20, 2018 17:46:45)(s) Summary — 1 rows out of 1 — Search

Status	Name	Verification Type	Verification Technique	Number of Policy(s)	Combination Algorithm	Enforcement Algorithm	Policy List
UpToDate	Policy Verification (June 20, 2018 17:46:45)	Standard	Single Policy	1	Deny-overrides	Deny Biased	ABAC:Reviewer Policy

Result(s) with selected verification (Policy Verification (June 20, 2018 17:46:45)) — 1 rows out of 1 — Search

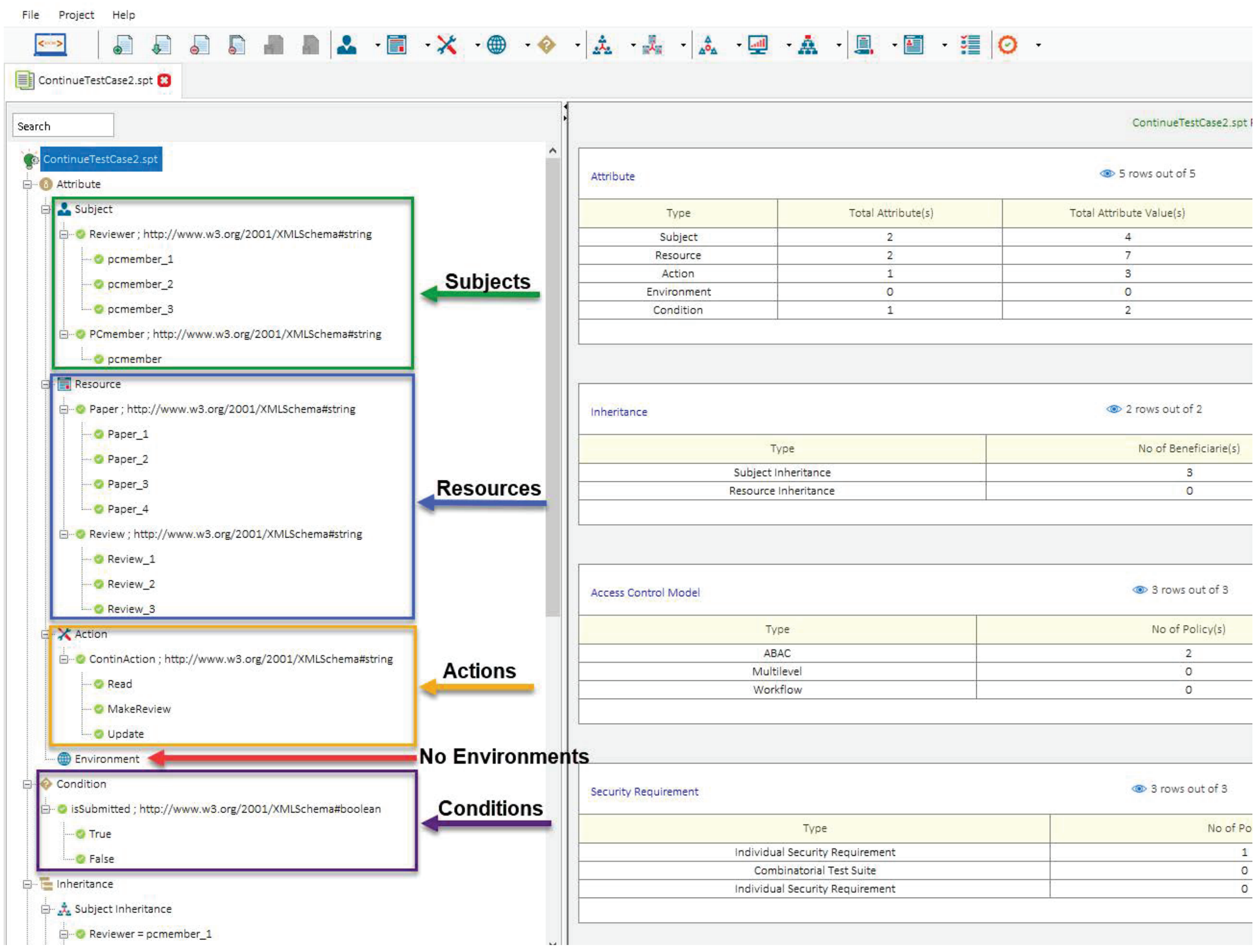
Requirement Schema	Subject	Resource	Action	Environment	Condition	Decision	Verification Result
Test Case 1	PCmember = pcmember & Reviewer = pcmember_3	Review = Review_3	ContinAction = Read	Environment = Any Value	isSubmitted = False	Permit	FALSE

Fig. 12. Updated Results: Reviewer Policy (No Rule Conflict)

Now the Rule Conflict is gone. Both policies are agreeing in their Access Decision = pcmember & pcmember_3 should not be Permitted to Read resource Review_3.

8 Setting Up The Policies – Test Case 2 (Not Protected Resource)

This continue example contains two policies (PCmember Policy & Reviewer Policy). The attributes in this example have been changed slightly from previous Test Case 1. Paper has gained a new attribute valued called "'Paper_4"'.The Attribute/Attribute Values include in these policies are as shown in Figure 13.



```
ContinueTestCase2.spt
  Attribute
    Subject
      Reviewer ; http://www.w3.org/2001/XMLSchema#string
        pcmember_1
        pcmember_2
        pcmember_3
      PCmember ; http://www.w3.org/2001/XMLSchema#string
        pcmember
    Resource
      Paper ; http://www.w3.org/2001/XMLSchema#string
        Paper_1
        Paper_2
        Paper_3
        Paper_4
      Review ; http://www.w3.org/2001/XMLSchema#string
        Review_1
        Review_2
        Review_3
    Action
      ContinAction ; http://www.w3.org/2001/XMLSchema#string
        Read
        MakeReview
        Update
    Environment
  Condition
    isSubmitted ; http://www.w3.org/2001/XMLSchema#boolean
      True
      False
  Inheritance
    Subject Inheritance
      Reviewer = pcmember_1
```

ContinueTestCase2.spt

Attribute — 5 rows out of 5

Type	Total Attribute(s)	Total Attribute Value(s)
Subject	2	4
Resource	2	7
Action	1	3
Environment	0	0
Condition	1	2

Inheritance — 2 rows out of 2

Type	No of Beneficiarie(s)
Subject Inheritance	3
Resource Inheritance	0

Access Control Model — 3 rows out of 3

Type	No of Policy(s)
ABAC	2
Multilevel	0
Workflow	0

Security Requirement — 3 rows out of 3

Type	No of Po
Individual Security Requirement	1
Combinatorial Test Suite	0
Individual Security Requirement	0

Fig. 13. Test Case 2

9 Subject Inheritance – Test Case 2 (Not Protected Resource)

To help us generate policy Rules more quickly we will define some Subject Inheritance relationships. Please note these are the same definitions from Test Case 1. For this continue policy example we will define (3) Subject Inheritance Relationships as follows:

Subject Inheritance:

Beneficiary Values → Reviewer (pcmember_1, pcmember_2, pcmember_3)

Inherited Values → PCmember

If we have created these Relationships correctly based on the above direction it will look like this in Security Policy Tool:

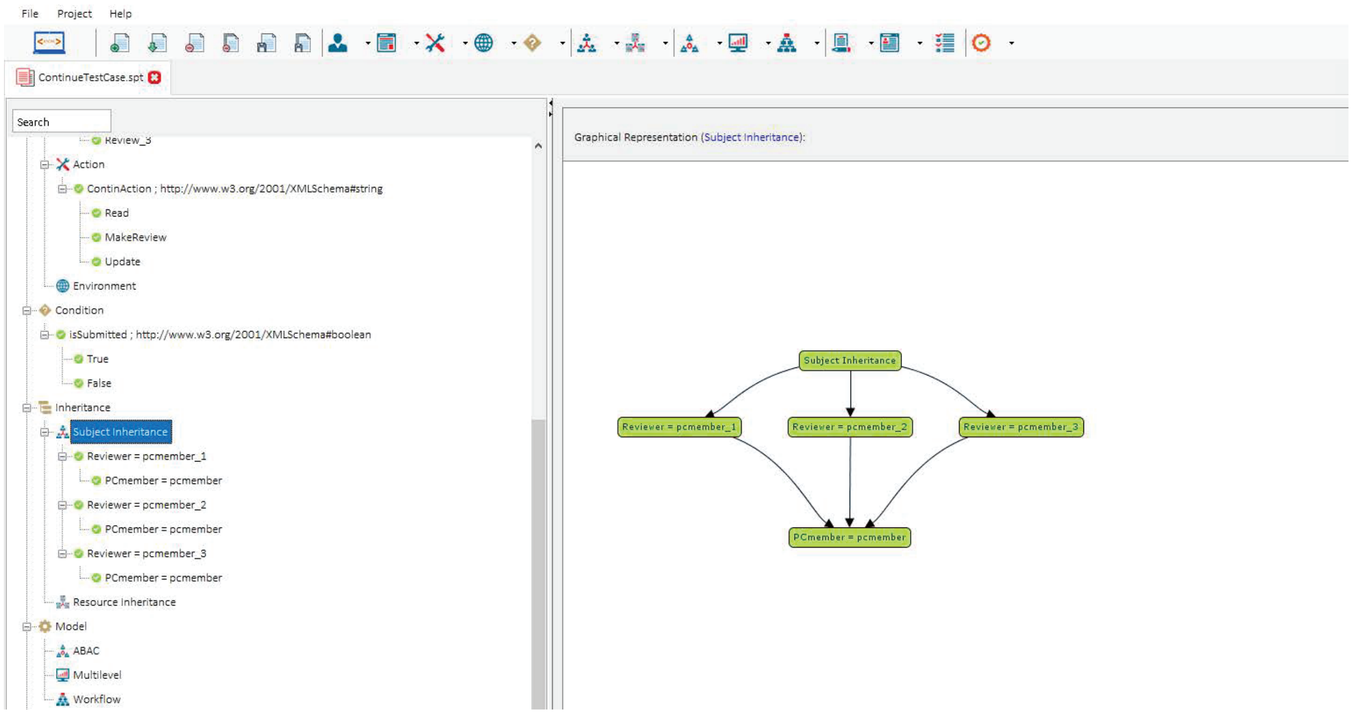


Fig. 14. Subject Inheritance Relationships

By defining these relationships, any (Originated) Rules with Decision = Permit given to PCmember = pcmember will now also be given to pcmember_1, pcmember_2, and pcmember_3 as (Inherited) Rules in our policies. Originated Rules with Decision = Deny are never Inherited. This is because typically Beneficiaries in these relationships are higher ranking/senior roles that by nature will have less restrictions (e.g., denying access) than roles that are providing the Inheritance Value (e.g., pcmember in our example).

Hence, it will authorize the Beneficiaries to obtain all privileges of Inherited Values (e.g., generally lower-level roles) while not obtaining their typically tighter restrictions. If you would like Beneficiaries to be Denied access similar to their Inherited Value you can still do so by manually creating individual rules when you begin modeling.

10 Modeling Your Policy – Test Case 2 (Not Protected Resource)

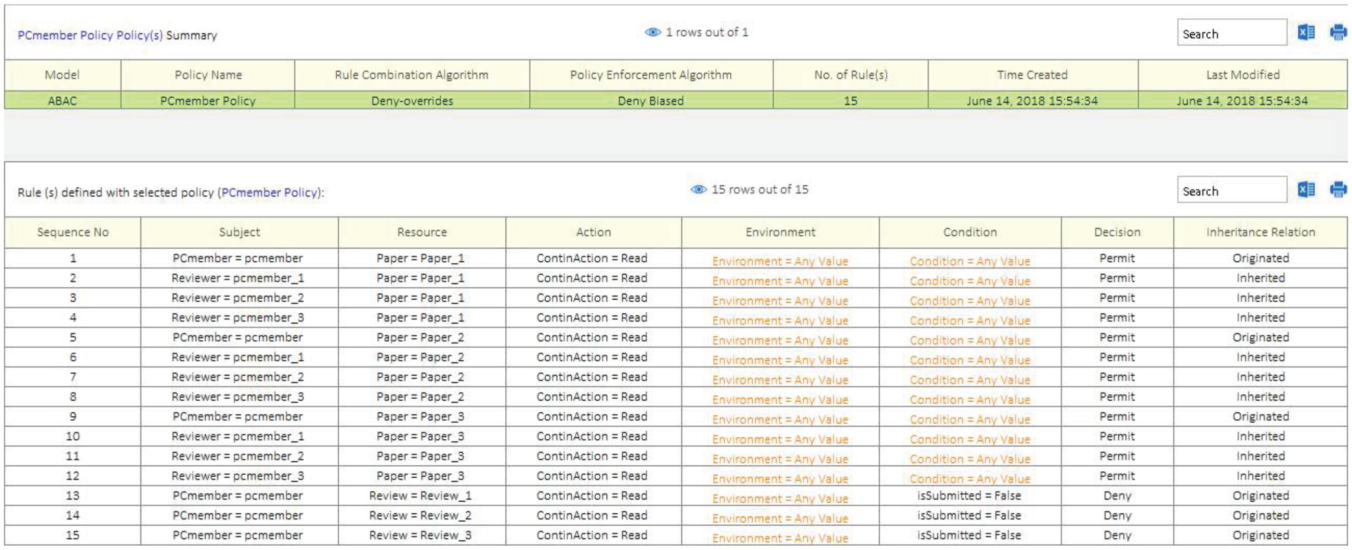
Now that we have entered our attributes we can model our two policies (PCmember Policy & Reviewer Policy). See the list below of the rules contained in each of these policies. You can open a "New (blank) Project" and build these policies by entering the following rules below:

PCmember Policy:
(PCmember = pcmember, Read, Paper_1) → Permit
(PCmember = pcmember, Read, Paper_2) →Permit
(PCmember = pcmember, Read, Paper_3) →Permit
(PCmember = pcmember, isSubmitted: False, Read, Review_1) →Deny
(PCmember = pcmember, isSubmitted: False, Read, Review_2) →Deny
(PCmember = pcmember, isSubmitted: False, Read, Review_3) →Deny
NOTE: You will also generate 9 additional "'Inherited Rules" due to the first 3 Permit Rules

Reviewer Policy:
(Reviewer = pcmember_1, isSubmitted: False, Update, Review_1) →Permit
(Reviewer = pcmember_2, isSubmitted: False, Update, Review_2) →Permit
(Reviewer = pcmember_3, isSubmitted: False, Update, Review_3) →Permit
(Reviewer = pcmember_1, isSubmitted: False, MakeReview, Paper_1) →Permit
(Reviewer = pcmember_2, isSubmitted: False, MakeReview, Paper_2) →Permit

(Reviewer = pcmember_3, isSubmitted: False, MakeReview, Paper_3) →Permit

After entering the rules above your modeled policies should look like the screenshots below. If you did not create your own Project File, you can simply open Security Policy Tool – Project File: ContinueTestCase2 and these policies will have been already created for you.



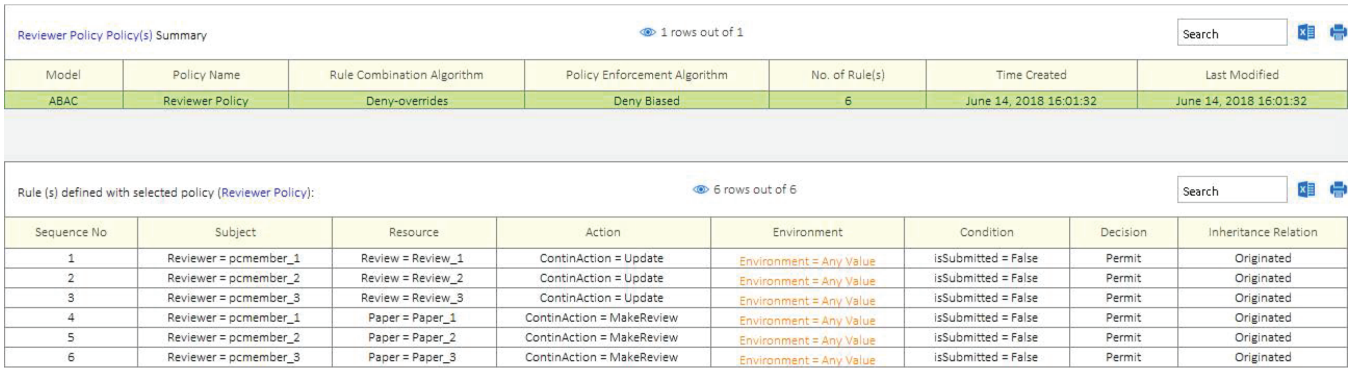
PCmember Policy Policy(s) Summary — 1 rows out of 1 — Search

Model	Policy Name	Rule Combination Algorithm	Policy Enforcement Algorithm	No. of Rule(s)	Time Created	Last Modified
ABAC	PCmember Policy	Deny-overrides	Deny Biased	15	June 14, 2018 15:54:34	June 14, 2018 15:54:34

Rule (s) defined with selected policy (PCmember Policy): — 15 rows out of 15 — Search

Sequence No	Subject	Resource	Action	Environment	Condition	Decision	Inheritance Relation
1	PCmember = pcmember	Paper = Paper_1	ContinAction = Read	Environment = Any Value	Condition = Any Value	Permit	Originated
2	Reviewer = pcmember_1	Paper = Paper_1	ContinAction = Read	Environment = Any Value	Condition = Any Value	Permit	Inherited
3	Reviewer = pcmember_2	Paper = Paper_1	ContinAction = Read	Environment = Any Value	Condition = Any Value	Permit	Inherited
4	Reviewer = pcmember_3	Paper = Paper_1	ContinAction = Read	Environment = Any Value	Condition = Any Value	Permit	Inherited
5	PCmember = pcmember	Paper = Paper_2	ContinAction = Read	Environment = Any Value	Condition = Any Value	Permit	Originated
6	Reviewer = pcmember_1	Paper = Paper_2	ContinAction = Read	Environment = Any Value	Condition = Any Value	Permit	Inherited
7	Reviewer = pcmember_2	Paper = Paper_2	ContinAction = Read	Environment = Any Value	Condition = Any Value	Permit	Inherited
8	Reviewer = pcmember_3	Paper = Paper_2	ContinAction = Read	Environment = Any Value	Condition = Any Value	Permit	Inherited
9	PCmember = pcmember	Paper = Paper_3	ContinAction = Read	Environment = Any Value	Condition = Any Value	Permit	Originated
10	Reviewer = pcmember_1	Paper = Paper_3	ContinAction = Read	Environment = Any Value	Condition = Any Value	Permit	Inherited
11	Reviewer = pcmember_2	Paper = Paper_3	ContinAction = Read	Environment = Any Value	Condition = Any Value	Permit	Inherited
12	Reviewer = pcmember_3	Paper = Paper_3	ContinAction = Read	Environment = Any Value	Condition = Any Value	Permit	Inherited
13	PCmember = pcmember	Review = Review_1	ContinAction = Read	Environment = Any Value	isSubmitted = False	Deny	Originated
14	PCmember = pcmember	Review = Review_2	ContinAction = Read	Environment = Any Value	isSubmitted = False	Deny	Originated
15	PCmember = pcmember	Review = Review_3	ContinAction = Read	Environment = Any Value	isSubmitted = False	Deny	Originated

Fig. 15. PCmember Policy



Reviewer Policy Policy(s) Summary — 1 rows out of 1 — Search

Model	Policy Name	Rule Combination Algorithm	Policy Enforcement Algorithm	No. of Rule(s)	Time Created	Last Modified
ABAC	Reviewer Policy	Deny-overrides	Deny Biased	6	June 14, 2018 16:01:32	June 14, 2018 16:01:32

Rule (s) defined with selected policy (Reviewer Policy): — 6 rows out of 6 — Search

Sequence No	Subject	Resource	Action	Environment	Condition	Decision	Inheritance Relation
1	Reviewer = pcmember_1	Review = Review_1	ContinAction = Update	Environment = Any Value	isSubmitted = False	Permit	Originated
2	Reviewer = pcmember_2	Review = Review_2	ContinAction = Update	Environment = Any Value	isSubmitted = False	Permit	Originated
3	Reviewer = pcmember_3	Review = Review_3	ContinAction = Update	Environment = Any Value	isSubmitted = False	Permit	Originated
4	Reviewer = pcmember_1	Paper = Paper_1	ContinAction = MakeReview	Environment = Any Value	isSubmitted = False	Permit	Originated
5	Reviewer = pcmember_2	Paper = Paper_2	ContinAction = MakeReview	Environment = Any Value	isSubmitted = False	Permit	Originated
6	Reviewer = pcmember_3	Paper = Paper_3	ContinAction = MakeReview	Environment = Any Value	isSubmitted = False	Permit	Originated

Fig. 16. Reviewer Policy

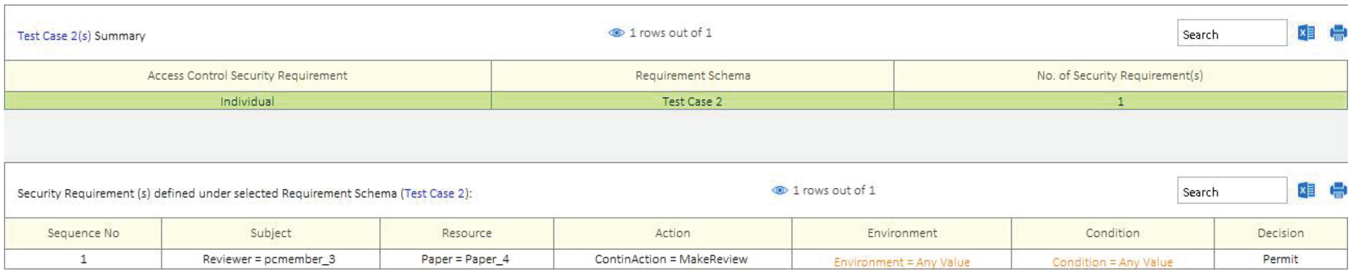
11 Individual Security Requirement - Test Case 2 (Not Protected Resource)

The final step before analyzing these policies for errors is to create individual security requirements to use for testing. If you are building a "New (blank) Project" on your own you will enter the following security requirement below:

Individual Security Requirement:

(Reviewer = pcmember_3) & (Action = MakeReview) & (Paper = Paper_4) → decision = Permit

After entering the rule above your individual security requirement should look like the screenshot below. If you did not create your own Project File you can simply open Security Policy Tool – Project File: ContinueTestCase2 and this requirement will have been already created for you.



Test Case 2(s) Summary · 1 rows out of 1 · Search

Access Control Security Requirement	Requirement Schema	No. of Security Requirement(s)
Individual	Test Case 2	1

Security Requirement (s) defined under selected Requirement Schema (Test Case 2): · 1 rows out of 1 · Search

Sequence No	Subject	Resource	Action	Environment	Condition	Decision
1	Reviewer = pcmember_3	Paper = Paper_4	ContinAction = MakeReview	Environment = Any Value	Condition = Any Value	Permit

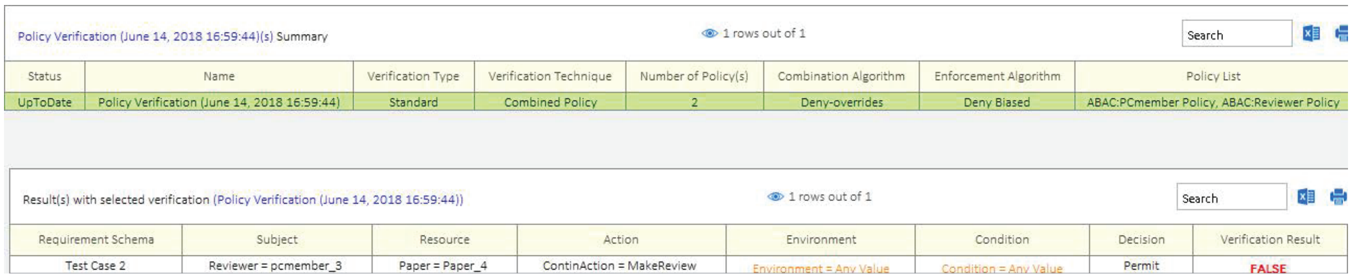
Fig. 17. Individual Security Requirements

12 Policy Verification/Analyzing Results - Test Case 2 (Not Protected Resource)

Now that we are ready to test our policies let's discuss the error we will be looking at in this second example. When policies are designed there is potential for a "Not Protected Resource" error being created by mistake. A Not Protected Resource error occurs when a resource is created but without protection from any rules.

For example, when the policy author was designing the logic for these continue policies; the author created a resource "Paper_4" with no protections. This means there are not currently any rules defined that are giving a decision for an access request to the resource. This Not Protected Resource error is not caused by any specific rules in either of our policies; it is caused due to a lack of rules created to cover this resource.

Next, we will run one "Combined Policy" Verification to reveal the Not Protected Resource error that is present in our policies. To do this, we will select Test Case 2 (security requirement) and PCmember Policy & Reviewer Policy as a Combined Policy Verification and analyze our verification result. Again, this will have already been done for you if you open Project File: ContinueTestCase2.



Policy Verification (June 14, 2018 16:59:44)(s) Summary · 1 rows out of 1 · Search

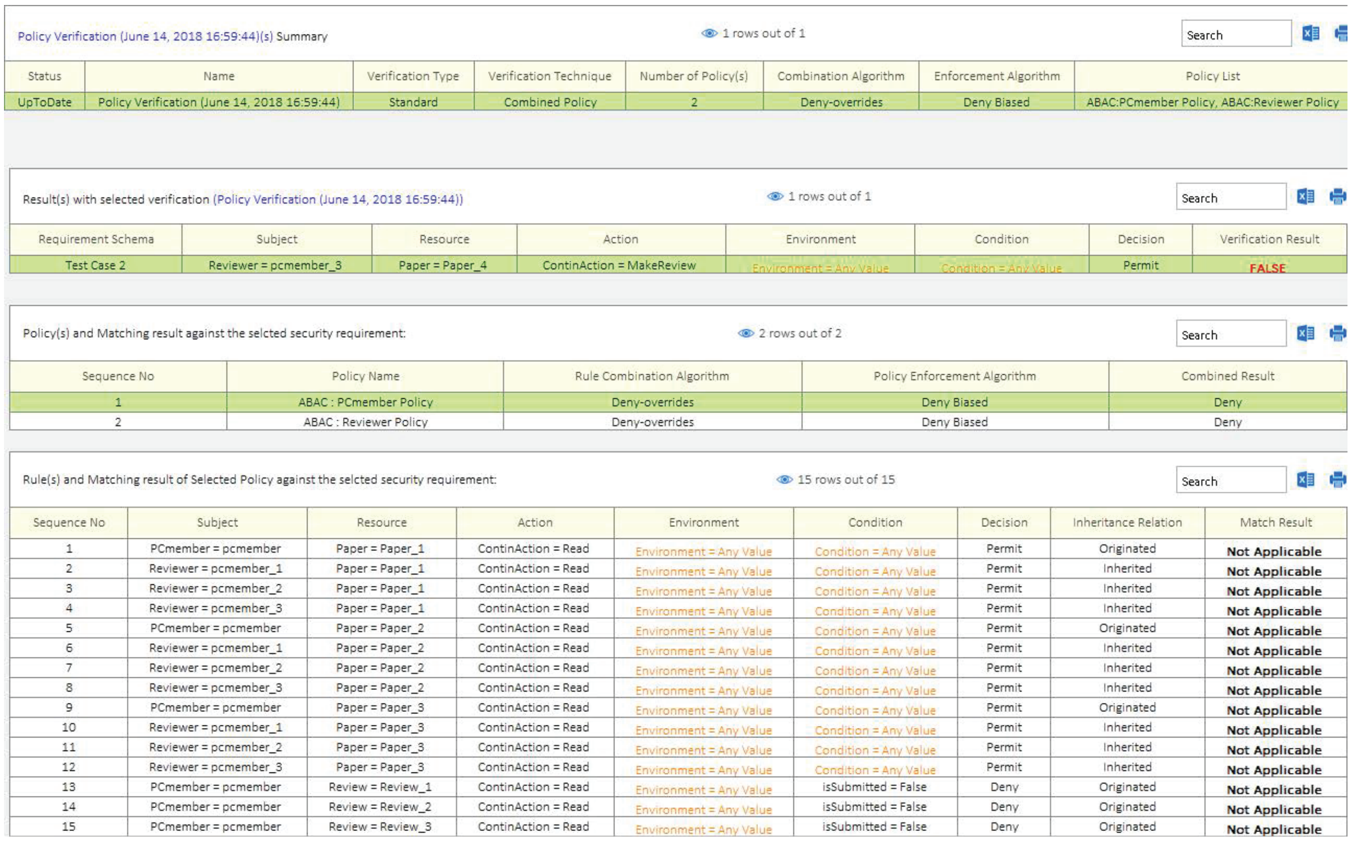
Status	Name	Verification Type	Verification Technique	Number of Policy(s)	Combination Algorithm	Enforcement Algorithm	Policy List
UpToDate	Policy Verification (June 14, 2018 16:59:44)	Standard	Combined Policy	2	Deny-overrides	Deny Biased	ABAC:PCmember Policy, ABAC:Reviewer Policy

Result(s) with selected verification (Policy Verification (June 14, 2018 16:59:44)) · 1 rows out of 1 · Search

Requirement Schema	Subject	Resource	Action	Environment	Condition	Decision	Verification Result
Test Case 2	Reviewer = pcmember_3	Paper = Paper_4	ContinAction = MakeReview	Environment = Any Value	Condition = Any Value	Permit	FALSE

Fig. 18. Combined Policy x Test Case 2

By clicking on the Verification Result, we can analyze deeper the reasoning for the "False" result we have received. Here is where we will notice we have not created any Rules that are attached to Resource = Paper_4. We see this by noticing that every "Match Result" is "Not Applicable" whereas if there were Rules protecting this resource we would have seen at least one Rule with a (Permit or Deny) Match Result.


Policy Verification (June 14, 2018 16:59:44)(s) Summary — 1 rows out of 1 — Search

Status	Name	Verification Type	Verification Technique	Number of Policy(s)	Combination Algorithm	Enforcement Algorithm	Policy List
UpToDate	Policy Verification (June 14, 2018 16:59:44)	Standard	Combined Policy	2	Deny-overrides	Deny Biased	ABAC:PCmember Policy, ABAC:Reviewer Policy

Result(s) with selected verification (Policy Verification (June 14, 2018 16:59:44)) — 1 rows out of 1 — Search

Requirement Schema	Subject	Resource	Action	Environment	Condition	Decision	Verification Result
Test Case 2	Reviewer = pcmember_3	Paper = Paper_4	ContinAction = MakeReview	Environment = Any Value	Condition = Any Value	Permit	FALSE

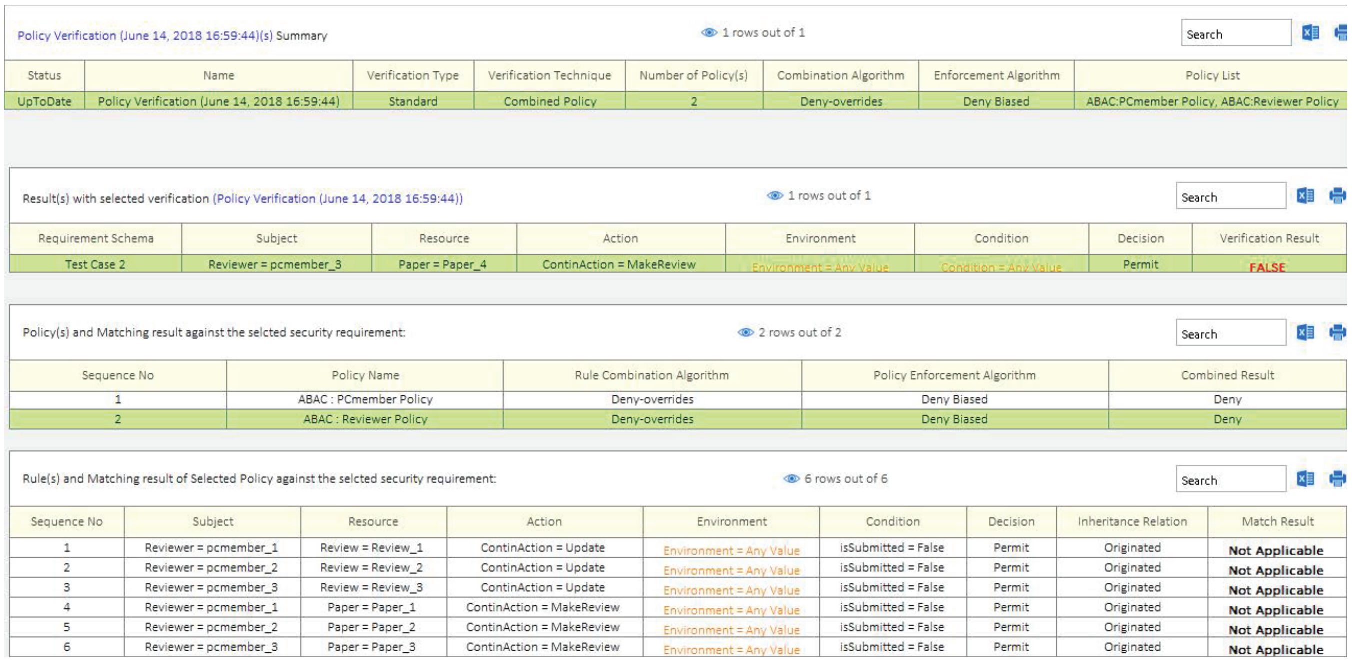
Policy(s) and Matching result against the selcted security requirement: — 2 rows out of 2 — Search

Sequence No	Policy Name	Rule Combination Algorithm	Policy Enforcement Algorithm	Combined Result
1	ABAC : PCmember Policy	Deny-overrides	Deny Biased	Deny
2	ABAC : Reviewer Policy	Deny-overrides	Deny Biased	Deny

Rule(s) and Matching result of Selected Policy against the selcted security requirement: — 15 rows out of 15 — Search

Sequence No	Subject	Resource	Action	Environment	Condition	Decision	Inheritance Relation	Match Result
1	PCmember = pcmember	Paper = Paper_1	ContinAction = Read	Environment = Any Value	Condition = Any Value	Permit	Originated	Not Applicable
2	Reviewer = pcmember_1	Paper = Paper_1	ContinAction = Read	Environment = Any Value	Condition = Any Value	Permit	Inherited	Not Applicable
3	Reviewer = pcmember_2	Paper = Paper_1	ContinAction = Read	Environment = Any Value	Condition = Any Value	Permit	Inherited	Not Applicable
4	Reviewer = pcmember_3	Paper = Paper_1	ContinAction = Read	Environment = Any Value	Condition = Any Value	Permit	Inherited	Not Applicable
5	PCmember = pcmember	Paper = Paper_2	ContinAction = Read	Environment = Any Value	Condition = Any Value	Permit	Originated	Not Applicable
6	Reviewer = pcmember_1	Paper = Paper_2	ContinAction = Read	Environment = Any Value	Condition = Any Value	Permit	Inherited	Not Applicable
7	Reviewer = pcmember_2	Paper = Paper_2	ContinAction = Read	Environment = Any Value	Condition = Any Value	Permit	Inherited	Not Applicable
8	Reviewer = pcmember_3	Paper = Paper_2	ContinAction = Read	Environment = Any Value	Condition = Any Value	Permit	Inherited	Not Applicable
9	PCmember = pcmember	Paper = Paper_3	ContinAction = Read	Environment = Any Value	Condition = Any Value	Permit	Originated	Not Applicable
10	Reviewer = pcmember_1	Paper = Paper_3	ContinAction = Read	Environment = Any Value	Condition = Any Value	Permit	Inherited	Not Applicable
11	Reviewer = pcmember_2	Paper = Paper_3	ContinAction = Read	Environment = Any Value	Condition = Any Value	Permit	Inherited	Not Applicable
12	Reviewer = pcmember_3	Paper = Paper_3	ContinAction = Read	Environment = Any Value	Condition = Any Value	Permit	Inherited	Not Applicable
13	PCmember = pcmember	Review = Review_1	ContinAction = Read	Environment = Any Value	isSubmitted = False	Deny	Originated	Not Applicable
14	PCmember = pcmember	Review = Review_2	ContinAction = Read	Environment = Any Value	isSubmitted = False	Deny	Originated	Not Applicable
15	PCmember = pcmember	Review = Review_3	ContinAction = Read	Environment = Any Value	isSubmitted = False	Deny	Originated	Not Applicable

Fig. 19. PCmember Policy: Match Results


Policy Verification (June 14, 2018 16:59:44)(s) Summary — 1 rows out of 1 — Search

Status	Name	Verification Type	Verification Technique	Number of Policy(s)	Combination Algorithm	Enforcement Algorithm	Policy List
UpToDate	Policy Verification (June 14, 2018 16:59:44)	Standard	Combined Policy	2	Deny-overrides	Deny Biased	ABAC:PCmember Policy, ABAC:Reviewer Policy

Result(s) with selected verification (Policy Verification (June 14, 2018 16:59:44)) — 1 rows out of 1 — Search

Requirement Schema	Subject	Resource	Action	Environment	Condition	Decision	Verification Result
Test Case 2	Reviewer = pcmember_3	Paper = Paper_4	ContinAction = MakeReview	Environment = Any Value	Condition = Any Value	Permit	FALSE

Policy(s) and Matching result against the selcted security requirement: — 2 rows out of 2 — Search

Sequence No	Policy Name	Rule Combination Algorithm	Policy Enforcement Algorithm	Combined Result
1	ABAC : PCmember Policy	Deny-overrides	Deny Biased	Deny
2	ABAC : Reviewer Policy	Deny-overrides	Deny Biased	Deny

Rule(s) and Matching result of Selected Policy against the selcted security requirement: — 6 rows out of 6 — Search

Sequence No	Subject	Resource	Action	Environment	Condition	Decision	Inheritance Relation	Match Result
1	Reviewer = pcmember_1	Review = Review_1	ContinAction = Update	Environment = Any Value	isSubmitted = False	Permit	Originated	Not Applicable
2	Reviewer = pcmember_2	Review = Review_2	ContinAction = Update	Environment = Any Value	isSubmitted = False	Permit	Originated	Not Applicable
3	Reviewer = pcmember_3	Review = Review_3	ContinAction = Update	Environment = Any Value	isSubmitted = False	Permit	Originated	Not Applicable
4	Reviewer = pcmember_1	Paper = Paper_1	ContinAction = MakeReview	Environment = Any Value	isSubmitted = False	Permit	Originated	Not Applicable
5	Reviewer = pcmember_2	Paper = Paper_2	ContinAction = MakeReview	Environment = Any Value	isSubmitted = False	Permit	Originated	Not Applicable
6	Reviewer = pcmember_3	Paper = Paper_3	ContinAction = MakeReview	Environment = Any Value	isSubmitted = False	Permit	Originated	Not Applicable

Fig. 20. Reviewer Policy: Match Results

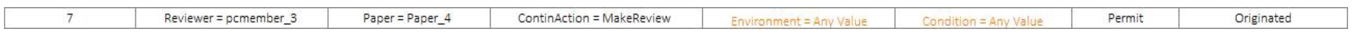
13 Resolving This Error - Test Case 2 (Not Protected Resource)

To eliminate a Not Protected Resource vulnerability the policy author would need to define a specific rule for the unprotected resource (Paper_4) and then test again to verify the intended access decision is being made based on the new rule's design.

For example, if we're to add this new Rule (7) below to the Reviewer Policy...

Reviewer Policy: Add (1) New Rule:

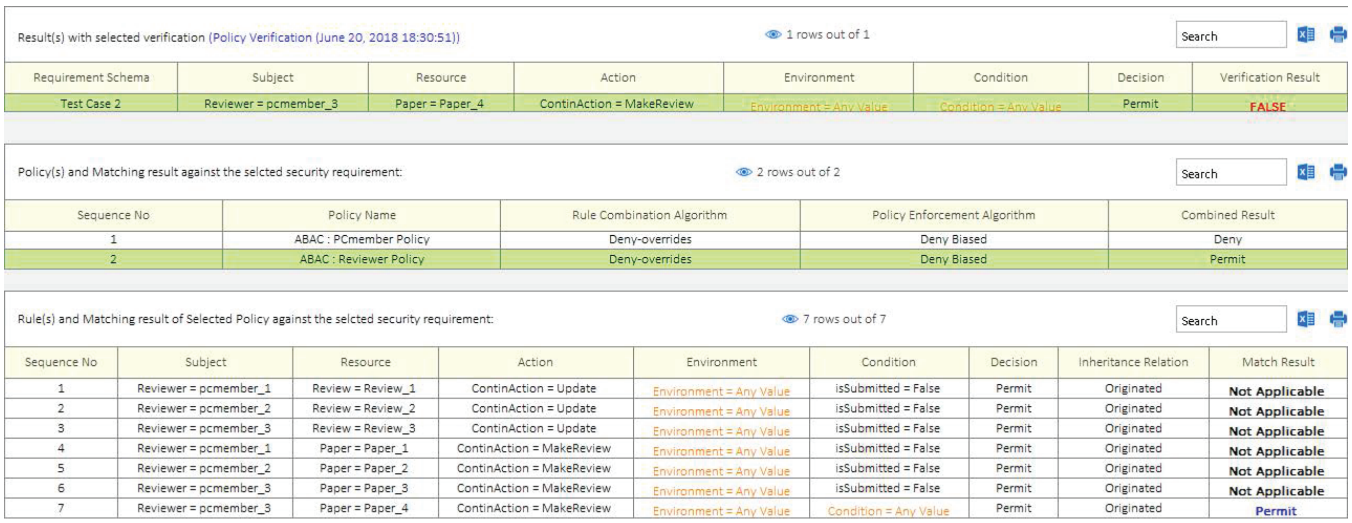
(Rule No. = 7) → (Reviewer = pcmember_3) → (Action = MakeReview) → (Resource = Paper_4) → decision = Permit



7	Reviewer = pcmember_3	Paper = Paper_4	ContinAction = MakeReview	Environment = Any Value	Condition = Any Value	Permit	Originated

Fig. 21. Reviewer Policy: New Rule (7)

Then retest using the same Policy Verification selections as last time we will get the same False Verification result due to how we defined our Combination Algorithms and Enforcement Algorithms. However, we can see in the Match Results that we have provided a rule for the system to evaluate for pcmember_3 accessing this resource.



Result(s) with selected verification (Policy Verification (June 20, 2018 18:30:51))

1 rows out of 1

Search

Requirement Schema	Subject	Resource	Action	Environment	Condition	Decision	Verification Result
Test Case 2	Reviewer = pcmember_3	Paper = Paper_4	ContinAction = MakeReview	Environment = Any Value	Condition = Any Value	Permit	FALSE

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

2 rows out of 2

Search

Sequence No	Policy Name	Rule Combination Algorithm	Policy Enforcement Algorithm	Combined Result
1	ABAC : PCmember Policy	Deny-overrides	Deny Biased	Deny
2	ABAC : Reviewer Policy	Deny-overrides	Deny Biased	Permit

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

7 rows out of 7

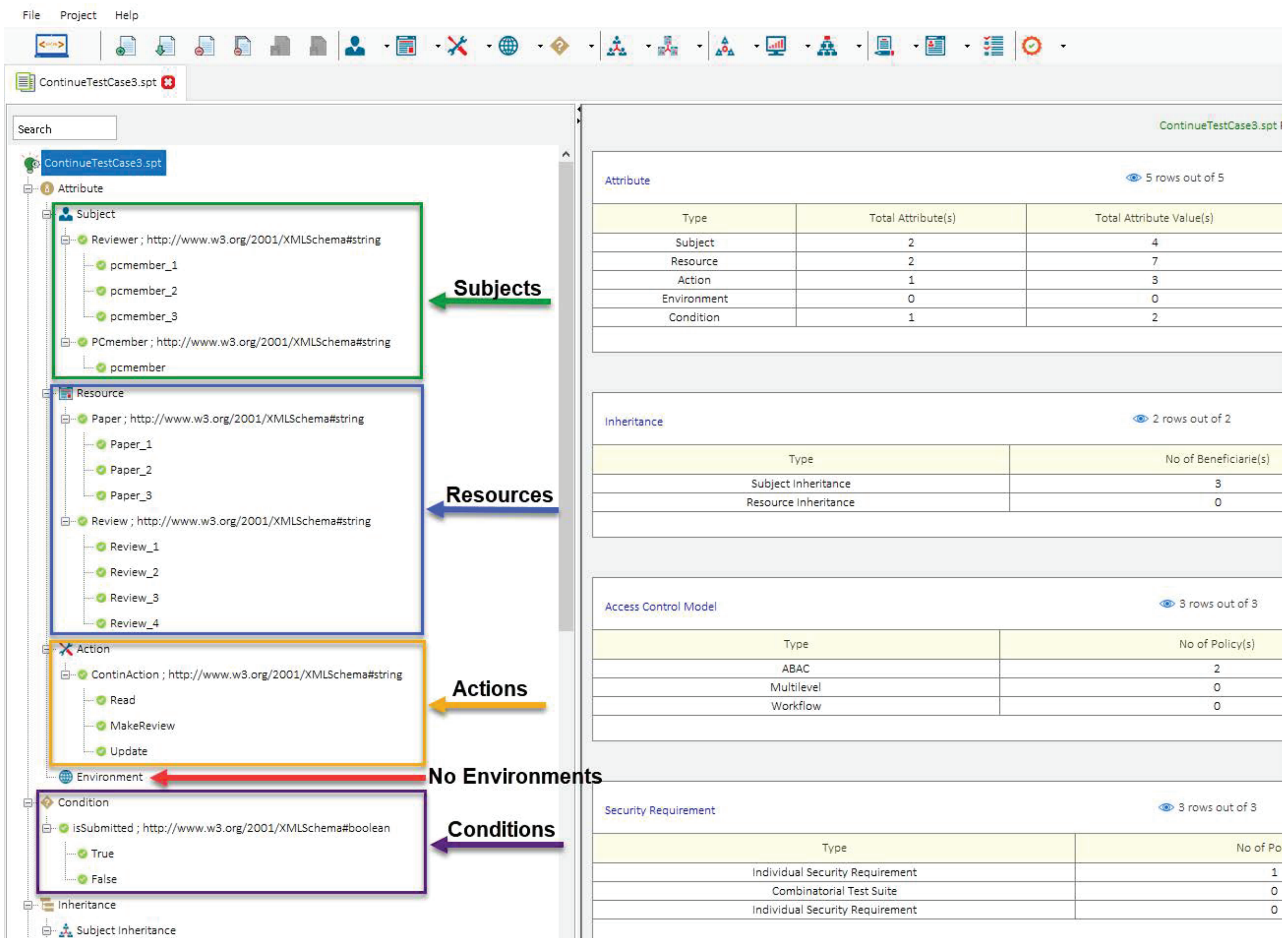
Search

Sequence No	Subject	Resource	Action	Environment	Condition	Decision	Inheritance Relation	Match Result
1	Reviewer = pcmember_1	Review = Review_1	ContinAction = Update	Environment = Any Value	isSubmitted = False	Permit	Originated	**Not Applicable**
2	Reviewer = pcmember_2	Review = Review_2	ContinAction = Update	Environment = Any Value	isSubmitted = False	Permit	Originated	**Not Applicable**
3	Reviewer = pcmember_3	Review = Review_3	ContinAction = Update	Environment = Any Value	isSubmitted = False	Permit	Originated	**Not Applicable**
4	Reviewer = pcmember_1	Paper = Paper_1	ContinAction = MakeReview	Environment = Any Value	isSubmitted = False	Permit	Originated	**Not Applicable**
5	Reviewer = pcmember_2	Paper = Paper_2	ContinAction = MakeReview	Environment = Any Value	isSubmitted = False	Permit	Originated	**Not Applicable**
6	Reviewer = pcmember_3	Paper = Paper_3	ContinAction = MakeReview	Environment = Any Value	isSubmitted = False	Permit	Originated	**Not Applicable**
7	Reviewer = pcmember_3	Paper = Paper_4	ContinAction = MakeReview	Environment = Any Value	Condition = Any Value	Permit	Originated	**Permit**

Fig. 22. Updated Policy: Resource Now Protected

14 Setting Up The Policies – Test Case 3 (Undecided Rule)

This continue example contains two policies (PCmember Policy & Reviewer Policy). The attributes in this example have been changed slightly from previous Test Case 2. Paper no longer has attribute value "Paper_4" and now Review has gained attribute value "Review_4". The Attribute/Attribute Values included in these policies are as shown in Figure 23.



ContinueTestCase3.spt
- Attribute
  - Subject
    - Reviewer ; http://www.w3.org/2001/XMLSchema#string
      - pcmember_1
      - pcmember_2
      - pcmember_3
    - PCmember ; http://www.w3.org/2001/XMLSchema#string
      - pcmember
  - Resource
    - Paper ; http://www.w3.org/2001/XMLSchema#string
      - Paper_1
      - Paper_2
      - Paper_3
    - Review ; http://www.w3.org/2001/XMLSchema#string
      - Review_1
      - Review_2
      - Review_3
      - Review_4
  - Action
    - ContinAction ; http://www.w3.org/2001/XMLSchema#string
      - Read
      - MakeReview
      - Update
  - Environment
- Condition
  - isSubmitted ; http://www.w3.org/2001/XMLSchema#boolean
    - True
    - False
- Inheritance
  - Subject Inheritance

ContinueTestCase3.spt

Attribute — 5 rows out of 5

Type	Total Attribute(s)	Total Attribute Value(s)
Subject	2	4
Resource	2	7
Action	1	3
Environment	0	0
Condition	1	2

Inheritance — 2 rows out of 2

Type	No of Beneficiarie(s)
Subject Inheritance	3
Resource Inheritance	0

Access Control Model — 3 rows out of 3

Type	No of Policy(s)
ABAC	2
Multilevel	0
Workflow	0

Security Requirement — 3 rows out of 3

Type	No of Po
Individual Security Requirement	1
Combinatorial Test Suite	0
Individual Security Requirement	0

Fig. 23. Test Case 3

15 Subject Inheritance – Test Case 3 (Undecided Rule)

To help us generate policy Rules more quickly we will define some Subject Inheritance relationships. Please note these are the same definitions from Test Case 1. For this continue policy example we will define (3) Subject Inheritance Relationships as follows:

Subject Inheritance:

Beneficiary Values → Reviewer (pcmember_1, pcmember_2, pcmember_3)

Inherited Values → PCmember

If we have created these Relationships correctly based on the above direction it will look like this in Security Policy Tool:

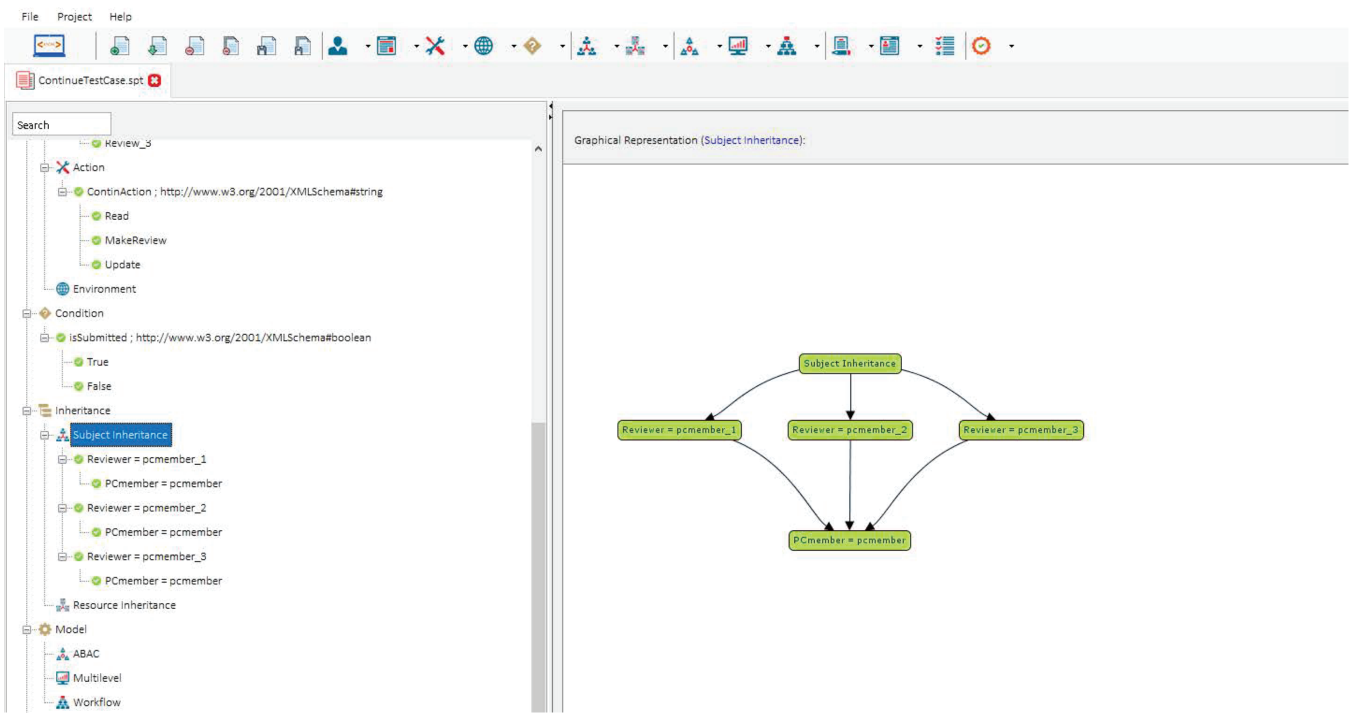


Fig. 24. Subject Inheritance Relationships

By defining these relationships, any (Originated) Rules with Decision = Permit given to PCmember = pcmember will now also be given to pcmember_1, pcmember_2, and pcmember_3 as (Inherited) Rules in our policies. Originated Rules with Decision = Deny are never Inherited. This is because typically Beneficiaries in these relationships are higher ranking/senior roles that by nature will have less restrictions (e.g., denying access) than roles that are providing the Inheritance Value (e.g., pcmember in our example).

Hence, it will authorize the Beneficiaries to obtain all privileges of Inherited Values (e.g., generally lower-level roles) while not obtaining their typically tighter restrictions. If you would like Beneficiaries to be Denied access similar to their Inherited Value you can still do so by manually creating individual rules when you begin modeling.

16 MODELING YOUR POLICY – TEST CASE 3 (UNDECIDED RULE)

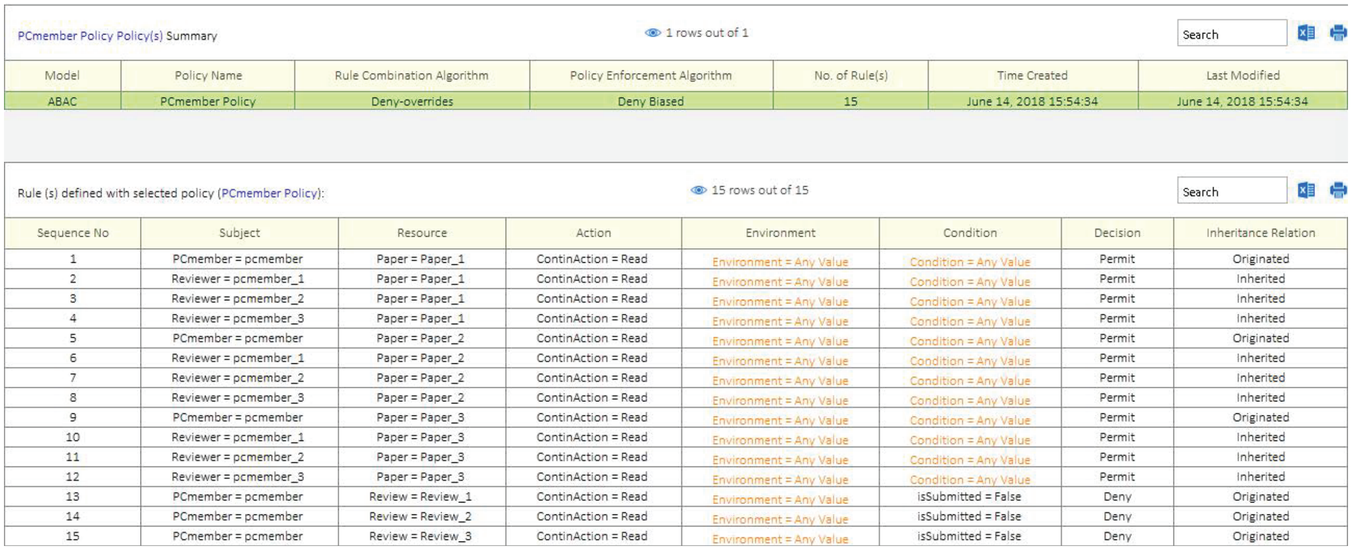
Now that we have entered our attributes we can model our two policies (PCmember Policy & Reviewer Policy). See the list below of the rules contained in each of these policies. You can open a "New (blank) Project" and build these policies by entering the following rules below:

PCmember Policy:
(PCmember = pcmember, Read, Paper_1) → Permit
(PCmember = pcmember, Read, Paper_2) →Permit
(PCmember = pcmember, Read, Paper_3) →Permit
(PCmember = pcmember, isSubmitted: False, Read, Review_1) →Deny
(PCmember = pcmember, isSubmitted: False, Read, Review_2) →Deny
(PCmember = pcmember, isSubmitted: False, Read, Review_3) →Deny
NOTE: You will also generate 9 additional "'Inherited Rules" due to the first 3 Permit Rules

Reviewer Policy:
(Reviewer = pcmember_1, isSubmitted: False, Update, Review_1) →Permit
(Reviewer = pcmember_2, isSubmitted: False, Update, Review_2) →Permit
(Reviewer = pcmember_3, isSubmitted: False, Update, Review_3) →Permit
(Reviewer = pcmember_1, isSubmitted: False, MakeReview, Paper_1) →Permit
(Reviewer = pcmember_2, isSubmitted: False, MakeReview, Paper_2) →Permit

(Reviewer = pcmember_3, isSubmitted: False, MakeReview, Paper_3) →Permit
(Reviewer = pcmember_3, isSubmitted: False, Update, Review_4) →Permit

After entering the rules above your modeled policies should look like the screenshots below. If you did not create your own Project File, you can simply open Security Policy Tool – Project File: ContinueTestCase3 and these policies will have been already created for you.



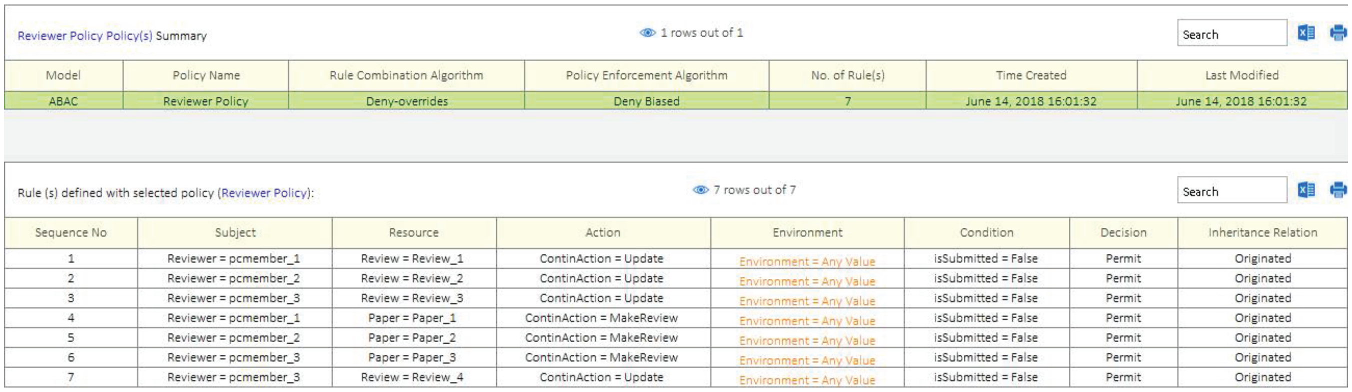
PCmember Policy Policy(s) Summary — 1 rows out of 1

Model	Policy Name	Rule Combination Algorithm	Policy Enforcement Algorithm	No. of Rule(s)	Time Created	Last Modified
ABAC	PCmember Policy	Deny-overrides	Deny Biased	15	June 14, 2018 15:54:34	June 14, 2018 15:54:34

Rule (s) defined with selected policy (PCmember Policy): — 15 rows out of 15

Sequence No	Subject	Resource	Action	Environment	Condition	Decision	Inheritance Relation
1	PCmember = pcmember	Paper = Paper_1	ContinAction = Read	Environment = Any Value	Condition = Any Value	Permit	Originated
2	Reviewer = pcmember_1	Paper = Paper_1	ContinAction = Read	Environment = Any Value	Condition = Any Value	Permit	Inherited
3	Reviewer = pcmember_2	Paper = Paper_1	ContinAction = Read	Environment = Any Value	Condition = Any Value	Permit	Inherited
4	Reviewer = pcmember_3	Paper = Paper_1	ContinAction = Read	Environment = Any Value	Condition = Any Value	Permit	Inherited
5	PCmember = pcmember	Paper = Paper_2	ContinAction = Read	Environment = Any Value	Condition = Any Value	Permit	Originated
6	Reviewer = pcmember_1	Paper = Paper_2	ContinAction = Read	Environment = Any Value	Condition = Any Value	Permit	Inherited
7	Reviewer = pcmember_2	Paper = Paper_2	ContinAction = Read	Environment = Any Value	Condition = Any Value	Permit	Inherited
8	Reviewer = pcmember_3	Paper = Paper_2	ContinAction = Read	Environment = Any Value	Condition = Any Value	Permit	Inherited
9	PCmember = pcmember	Paper = Paper_3	ContinAction = Read	Environment = Any Value	Condition = Any Value	Permit	Originated
10	Reviewer = pcmember_1	Paper = Paper_3	ContinAction = Read	Environment = Any Value	Condition = Any Value	Permit	Inherited
11	Reviewer = pcmember_2	Paper = Paper_3	ContinAction = Read	Environment = Any Value	Condition = Any Value	Permit	Inherited
12	Reviewer = pcmember_3	Paper = Paper_3	ContinAction = Read	Environment = Any Value	Condition = Any Value	Permit	Inherited
13	PCmember = pcmember	Review = Review_1	ContinAction = Read	Environment = Any Value	isSubmitted = False	Deny	Originated
14	PCmember = pcmember	Review = Review_2	ContinAction = Read	Environment = Any Value	isSubmitted = False	Deny	Originated
15	PCmember = pcmember	Review = Review_3	ContinAction = Read	Environment = Any Value	isSubmitted = False	Deny	Originated

Fig. 25. PCmember Policy



Reviewer Policy Policy(s) Summary — 1 rows out of 1

Model	Policy Name	Rule Combination Algorithm	Policy Enforcement Algorithm	No. of Rule(s)	Time Created	Last Modified
ABAC	Reviewer Policy	Deny-overrides	Deny Biased	7	June 14, 2018 16:01:32	June 14, 2018 16:01:32

Rule (s) defined with selected policy (Reviewer Policy): — 7 rows out of 7

Sequence No	Subject	Resource	Action	Environment	Condition	Decision	Inheritance Relation
1	Reviewer = pcmember_1	Review = Review_1	ContinAction = Update	Environment = Any Value	isSubmitted = False	Permit	Originated
2	Reviewer = pcmember_2	Review = Review_2	ContinAction = Update	Environment = Any Value	isSubmitted = False	Permit	Originated
3	Reviewer = pcmember_3	Review = Review_3	ContinAction = Update	Environment = Any Value	isSubmitted = False	Permit	Originated
4	Reviewer = pcmember_1	Paper = Paper_1	ContinAction = MakeReview	Environment = Any Value	isSubmitted = False	Permit	Originated
5	Reviewer = pcmember_2	Paper = Paper_2	ContinAction = MakeReview	Environment = Any Value	isSubmitted = False	Permit	Originated
6	Reviewer = pcmember_3	Paper = Paper_3	ContinAction = MakeReview	Environment = Any Value	isSubmitted = False	Permit	Originated
7	Reviewer = pcmember_3	Review = Review_4	ContinAction = Update	Environment = Any Value	isSubmitted = False	Permit	Originated

Fig. 26. Reviewer Policy

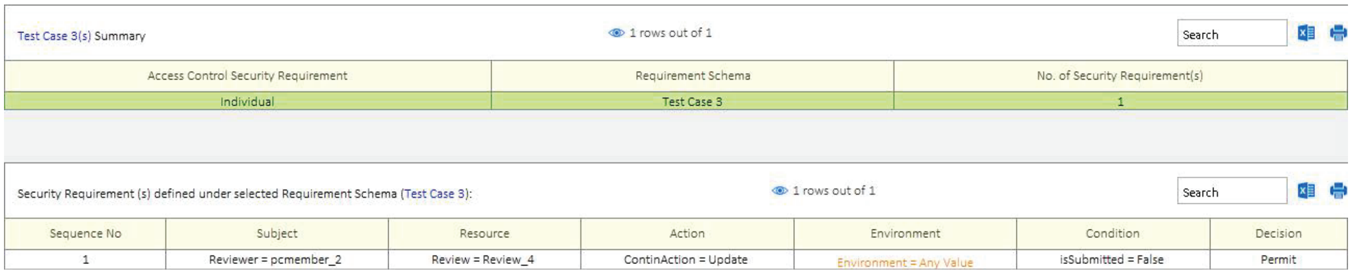
17 Individual Security Requirements - Test Case 3 (Undecided Rule)

The final step before analyzing these policies for errors is to create individual security requirements to use for testing. If you are building a "New (blank) Project" on your own you will enter the following security requirement below:

Individual Security Requirement:

(Reviewer = pcmember_2) & (isSubmitted = False) & (Action = Update) & (Review = Review_4) →decision = Permit

After entering the rule above your individual security requirement should look like the screenshot below. If you did not create your own Project File you can simply open Security Policy Tool – Project File: ContinueTestCase3 and this requirement will have been already created for you.


Test Case 3(s) Summary — 1 rows out of 1 — Search

Access Control Security Requirement	Requirement Schema	No. of Security Requirement(s)
Individual	Test Case 3	1

Security Requirement (s) defined under selected Requirement Schema (Test Case 3): — 1 rows out of 1 — Search

Sequence No	Subject	Resource	Action	Environment	Condition	Decision
1	Reviewer = pcmember_2	Review = Review_4	ContinAction = Update	Environment = Any Value	isSubmitted = False	Permit

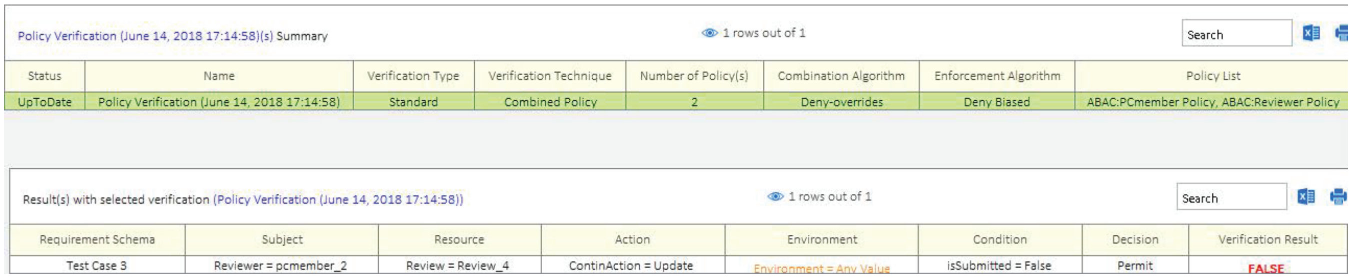
Fig. 27. Individual Security Requirement

18 Policy Verification/Analyzing Results - Test Case 3 (Undecided Rule)

Now that we are ready to test our policies let's discuss the error we will be looking at in this third example. When policies are designed there is potential for an "Undecided Rule" error being created. An Undecided Rule error occurs when your policy contains rules that are not entirely defined or are missing a step.

For example, when the policy author was designing the logic for these continue policies; the author created a rule for pcmember_3 to access "Review_4" but did not define access rules for pcmember_1 and pcmember_2 to access this resource. In this situation, if pcmember_1 or pcmember_2 were to attempt to take action on "Review_4," the system would be forced to make a default decision instead of a specified decision. This may create a security vulnerability due to your system's default evaluation decision being different than what you previously intended. Similar to the "Not Protected Resource" example previously, this error is caused due to the author missing rules. It is not caused due to flawed interpretation of existing rules contained in either of our policies as was the case in Test Case 1 (Rule Conflict).

Next, we will run one "Combined Policy" Verification to reveal the Undecided Rule error that is present in our policies. To do this, we will select Test Case 3 (security requirement) and PCmember Policy & Review Policy as a Combined Policy Verification and analyze our verification result. Again, this will have already been done for you if you open Project File: ContinueTestCase3.


Policy Verification (June 14, 2018 17:14:58)(s) Summary — 1 rows out of 1 — Search

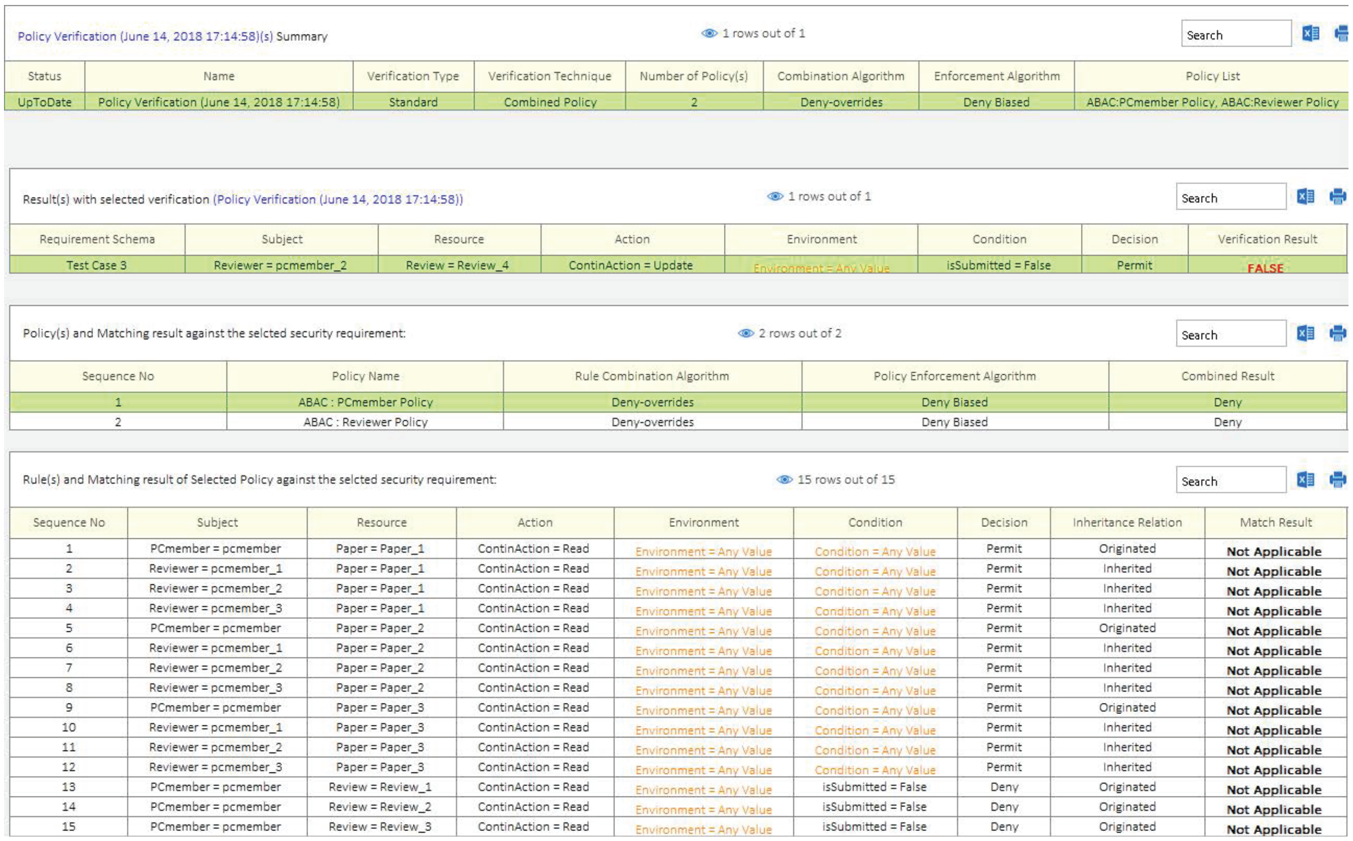
Status	Name	Verification Type	Verification Technique	Number of Policy(s)	Combination Algorithm	Enforcement Algorithm	Policy List
UpToDate	Policy Verification (June 14, 2018 17:14:58)	Standard	Combined Policy	2	Deny-overrides	Deny Biased	ABAC:PCmember Policy, ABAC:Reviewer Policy

Result(s) with selected verification (Policy Verification (June 14, 2018 17:14:58)) — 1 rows out of 1 — Search

Requirement Schema	Subject	Resource	Action	Environment	Condition	Decision	Verification Result
Test Case 3	Reviewer = pcmember_2	Review = Review_4	ContinAction = Update	Environment = Any Value	isSubmitted = False	Permit	FALSE

Fig. 28. Combined Policy x Test Case 3

Like we did in the "Not Protected Resource" example, by clicking on the Verification Result we can analyze deeper the reasoning for the "False" result we have received. Here is where we would notice we have not created Rules that are attached to Subject = pcmember_2 taking action on Resource = Review_4. We can see this by noticing that every "Match Result" is "Not Applicable" whereas if there were Rules existing for pcmember_2 and Resource = Review_4 we would have at least see one Rule with a (Permit or Deny) Match Result.



Policy Verification (June 14, 2018 17:14:58)(s) Summary — 1 rows out of 1

Status	Name	Verification Type	Verification Technique	Number of Policy(s)	Combination Algorithm	Enforcement Algorithm	Policy List
UpToDate	Policy Verification (June 14, 2018 17:14:58)	Standard	Combined Policy	2	Deny-overrides	Deny Biased	ABAC:PCmember Policy, ABAC:Reviewer Policy

Result(s) with selected verification (Policy Verification (June 14, 2018 17:14:58)) — 1 rows out of 1

Requirement Schema	Subject	Resource	Action	Environment	Condition	Decision	Verification Result
Test Case 3	Reviewer = pcmember_2	Review = Review_4	ContinAction = Update	Environment = Any Value	isSubmitted = False	Permit	FALSE

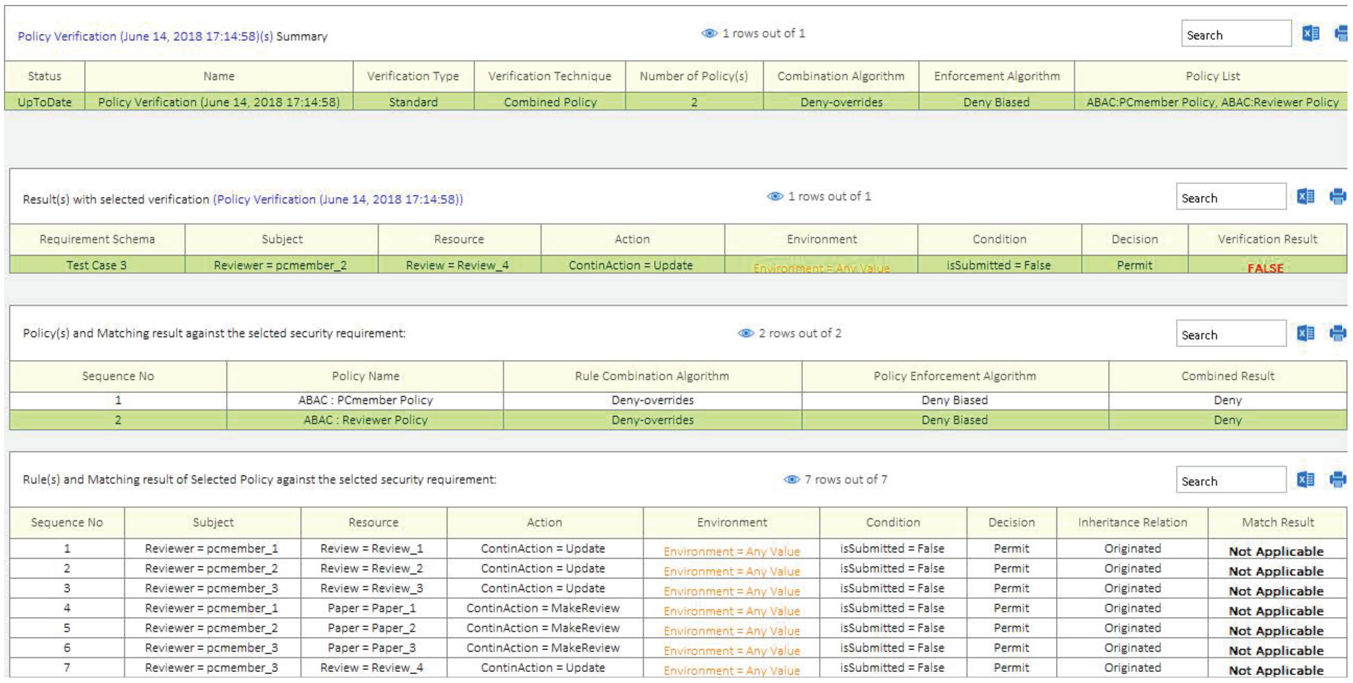
Policy(s) and Matching result against the selcted security requirement: — 2 rows out of 2

Sequence No	Policy Name	Rule Combination Algorithm	Policy Enforcement Algorithm	Combined Result
1	ABAC : PCmember Policy	Deny-overrides	Deny Biased	Deny
2	ABAC : Reviewer Policy	Deny-overrides	Deny Biased	Deny

Rule(s) and Matching result of Selected Policy against the selcted security requirement: — 15 rows out of 15

Sequence No	Subject	Resource	Action	Environment	Condition	Decision	Inheritance Relation	Match Result
1	PCmember = pcmember	Paper = Paper_1	ContinAction = Read	Environment = Any Value	Condition = Any Value	Permit	Originated	Not Applicable
2	Reviewer = pcmember_1	Paper = Paper_1	ContinAction = Read	Environment = Any Value	Condition = Any Value	Permit	Inherited	Not Applicable
3	Reviewer = pcmember_2	Paper = Paper_1	ContinAction = Read	Environment = Any Value	Condition = Any Value	Permit	Inherited	Not Applicable
4	Reviewer = pcmember_3	Paper = Paper_1	ContinAction = Read	Environment = Any Value	Condition = Any Value	Permit	Inherited	Not Applicable
5	PCmember = pcmember	Paper = Paper_2	ContinAction = Read	Environment = Any Value	Condition = Any Value	Permit	Originated	Not Applicable
6	Reviewer = pcmember_1	Paper = Paper_2	ContinAction = Read	Environment = Any Value	Condition = Any Value	Permit	Inherited	Not Applicable
7	Reviewer = pcmember_2	Paper = Paper_2	ContinAction = Read	Environment = Any Value	Condition = Any Value	Permit	Inherited	Not Applicable
8	Reviewer = pcmember_3	Paper = Paper_2	ContinAction = Read	Environment = Any Value	Condition = Any Value	Permit	Inherited	Not Applicable
9	PCmember = pcmember	Paper = Paper_3	ContinAction = Read	Environment = Any Value	Condition = Any Value	Permit	Originated	Not Applicable
10	Reviewer = pcmember_1	Paper = Paper_3	ContinAction = Read	Environment = Any Value	Condition = Any Value	Permit	Inherited	Not Applicable
11	Reviewer = pcmember_2	Paper = Paper_3	ContinAction = Read	Environment = Any Value	Condition = Any Value	Permit	Inherited	Not Applicable
12	Reviewer = pcmember_3	Paper = Paper_3	ContinAction = Read	Environment = Any Value	Condition = Any Value	Permit	Inherited	Not Applicable
13	PCmember = pcmember	Review = Review_1	ContinAction = Read	Environment = Any Value	isSubmitted = False	Deny	Originated	Not Applicable
14	PCmember = pcmember	Review = Review_2	ContinAction = Read	Environment = Any Value	isSubmitted = False	Deny	Originated	Not Applicable
15	PCmember = pcmember	Review = Review_3	ContinAction = Read	Environment = Any Value	isSubmitted = False	Deny	Originated	Not Applicable

Fig. 29. PCmember Policy: Match Results



Policy Verification (June 14, 2018 17:14:58)(s) Summary — 1 rows out of 1

Status	Name	Verification Type	Verification Technique	Number of Policy(s)	Combination Algorithm	Enforcement Algorithm	Policy List
UpToDate	Policy Verification (June 14, 2018 17:14:58)	Standard	Combined Policy	2	Deny-overrides	Deny Biased	ABAC:PCmember Policy, ABAC:Reviewer Policy

Result(s) with selected verification (Policy Verification (June 14, 2018 17:14:58)) — 1 rows out of 1

Requirement Schema	Subject	Resource	Action	Environment	Condition	Decision	Verification Result
Test Case 3	Reviewer = pcmember_2	Review = Review_4	ContinAction = Update	Environment = Any Value	isSubmitted = False	Permit	FALSE

Policy(s) and Matching result against the selcted security requirement: — 2 rows out of 2

Sequence No	Policy Name	Rule Combination Algorithm	Policy Enforcement Algorithm	Combined Result
1	ABAC : PCmember Policy	Deny-overrides	Deny Biased	Deny
2	ABAC : Reviewer Policy	Deny-overrides	Deny Biased	Deny

Rule(s) and Matching result of Selected Policy against the selcted security requirement: — 7 rows out of 7

Sequence No	Subject	Resource	Action	Environment	Condition	Decision	Inheritance Relation	Match Result
1	Reviewer = pcmember_1	Review = Review_1	ContinAction = Update	Environment = Any Value	isSubmitted = False	Permit	Originated	Not Applicable
2	Reviewer = pcmember_2	Review = Review_2	ContinAction = Update	Environment = Any Value	isSubmitted = False	Permit	Originated	Not Applicable
3	Reviewer = pcmember_3	Review = Review_3	ContinAction = Update	Environment = Any Value	isSubmitted = False	Permit	Originated	Not Applicable
4	Reviewer = pcmember_1	Paper = Paper_1	ContinAction = MakeReview	Environment = Any Value	isSubmitted = False	Permit	Originated	Not Applicable
5	Reviewer = pcmember_2	Paper = Paper_2	ContinAction = MakeReview	Environment = Any Value	isSubmitted = False	Permit	Originated	Not Applicable
6	Reviewer = pcmember_3	Paper = Paper_3	ContinAction = MakeReview	Environment = Any Value	isSubmitted = False	Permit	Originated	Not Applicable
7	Reviewer = pcmember_3	Review = Review_4	ContinAction = Update	Environment = Any Value	isSubmitted = False	Permit	Originated	Not Applicable

Fig. 30. Reviewer Policy: Match Results

As you can see there has not been a rule defined for pcmember_2 → Action → Review_4 which is known as an Undecided Rule error.

19 Resolving This Error - Test Case 3 (Undecided Rule)

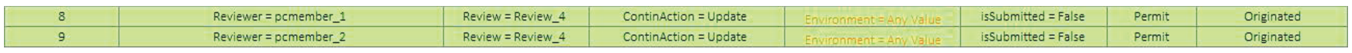
To solve this error, the policy author would need to define specific rules for all subject attributes in any policies (e.g., include pcmember_2) that determine pcmember access requets to Review_4.

For example, adding the rules below to the Reviewer Policy for our specific example…

Reviewer Policy: Add (2) New Rules:

(Rule No. = 8) → (Reviewer = pcmember_1) → (Action = Update) → (isSubmitted = False) →(Resource = Review_4) → decision = Permit

(Rule No. = 9) → (Reviewer = pcmember_2) → (Action = Update) → (isSubmitted = False) → (Resource = Review_4) → decision = Permit

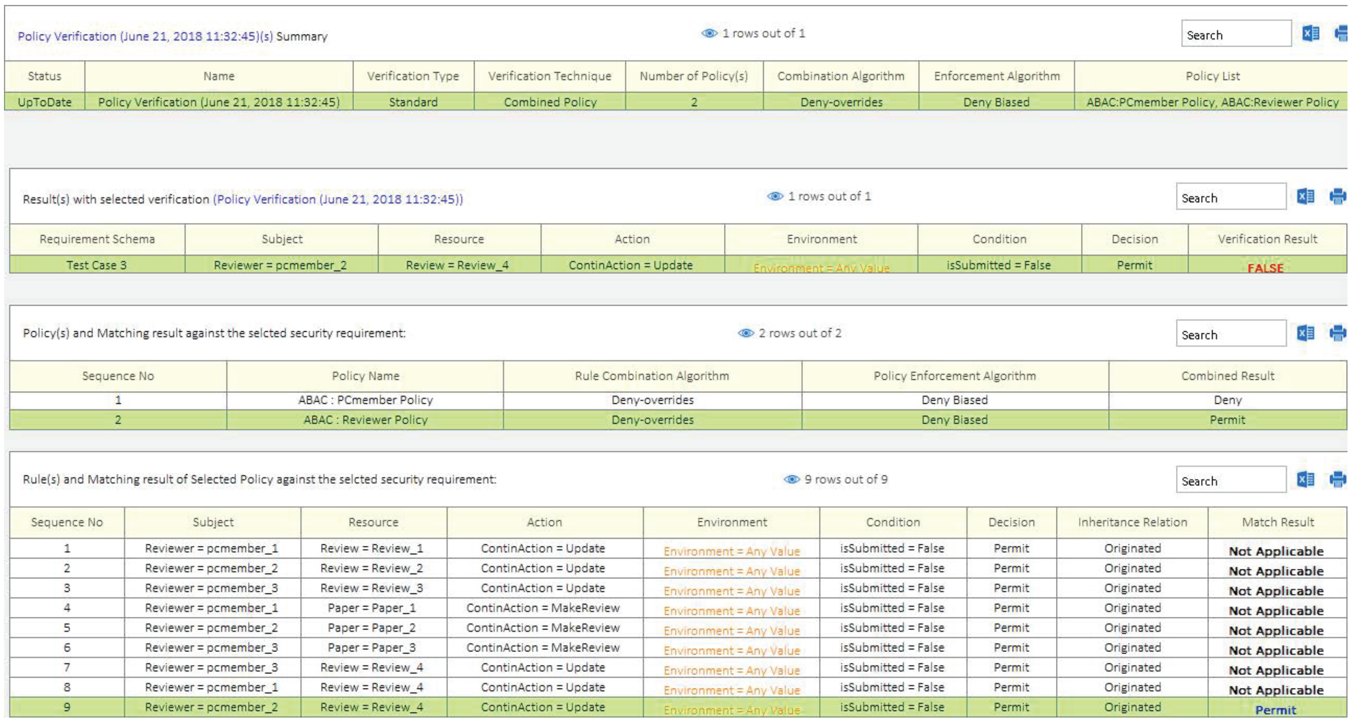


8	Reviewer = pcmember_1	Review = Review_4	ContinAction = Update	Environment = Any Value	isSubmitted = False	Permit	Originated
9	Reviewer = pcmember_2	Review = Review_4	ContinAction = Update	Environment = Any Value	isSubmitted = False	Permit	Originated

Fig. 31. Reviewer Policy: New Rules (8, 9)

Now, looking at our Verification results and Match Results we will see that we no longer have an "Undecided Rule" error occurring. The Verification Result is still "False" due to our choices in our Combination Algorithm = Deny-overrides and Enforcement Algorithm = Deny Biased.

For example, PCmember Policy has no rules related to the security requirement (pcmember_2 → Update → Review_4) we are using for testing which is why see all Match Rules = Not Applicable. Due to our selection to use Deny Biased for our Enforcement Algorithm the "Combined Result" for PCmember Policy = Deny. However, in the case of the Policy we have the Combined Result = Permit due to the new rules we added (e.g., see new Rule 9 below). Hence, we have opposing Combined Results (PCmember Policy = Deny; Reviewer Policy = Permit). Finally, the Combination Algorithm = Deny-overrides which makes a definitive answer for our Verification Results. The Deny-overrides selection overrules the Permit result from the PCmember Policy in favor of the Deny result from the Reviewer Policy to make the final Verification Result = False.



Policy Verification (June 21, 2018 11:32:45)(s) Summary — 1 rows out of 1 — Search

Status	Name	Verification Type	Verification Technique	Number of Policy(s)	Combination Algorithm	Enforcement Algorithm	Policy List
UpToDate	Policy Verification (June 21, 2018 11:32:45)	Standard	Combined Policy	2	Deny-overrides	Deny Biased	ABAC:PCmember Policy, ABAC:Reviewer Policy

Result(s) with selected verification (Policy Verification (June 21, 2018 11:32:45)) — 1 rows out of 1 — Search

Requirement Schema	Subject	Resource	Action	Environment	Condition	Decision	Verification Result
Test Case 3	Reviewer = pcmember_2	Review = Review_4	ContinAction = Update	Environment = Any Value	isSubmitted = False	Permit	FALSE

Policy(s) and Matching result against the selcted security requirement: — 2 rows out of 2 — Search

Sequence No	Policy Name	Rule Combination Algorithm	Policy Enforcement Algorithm	Combined Result
1	ABAC : PCmember Policy	Deny-overrides	Deny Biased	Deny
2	ABAC : Reviewer Policy	Deny-overrides	Deny Biased	Permit

Rule(s) and Matching result of Selected Policy against the selcted security requirement: — 9 rows out of 9 — Search

Sequence No	Subject	Resource	Action	Environment	Condition	Decision	Inheritance Relation	Match Result
1	Reviewer = pcmember_1	Review = Review_1	ContinAction = Update	Environment = Any Value	isSubmitted = False	Permit	Originated	Not Applicable
2	Reviewer = pcmember_2	Review = Review_2	ContinAction = Update	Environment = Any Value	isSubmitted = False	Permit	Originated	Not Applicable
3	Reviewer = pcmember_3	Review = Review_3	ContinAction = Update	Environment = Any Value	isSubmitted = False	Permit	Originated	Not Applicable
4	Reviewer = pcmember_1	Paper = Paper_1	ContinAction = MakeReview	Environment = Any Value	isSubmitted = False	Permit	Originated	Not Applicable
5	Reviewer = pcmember_2	Paper = Paper_2	ContinAction = MakeReview	Environment = Any Value	isSubmitted = False	Permit	Originated	Not Applicable
6	Reviewer = pcmember_3	Paper = Paper_3	ContinAction = MakeReview	Environment = Any Value	isSubmitted = False	Permit	Originated	Not Applicable
7	Reviewer = pcmember_3	Review = Review_4	ContinAction = Update	Environment = Any Value	isSubmitted = False	Permit	Originated	Not Applicable
8	Reviewer = pcmember_1	Review = Review_4	ContinAction = Update	Environment = Any Value	isSubmitted = False	Permit	Originated	Not Applicable
9	Reviewer = pcmember_2	Review = Review_4	ContinAction = Update	Environment = Any Value	isSubmitted = False	Permit	Originated	Permit

Fig. 32. Updated Results: No Undecided Rule

20 Conclusion

Now you should have a better understanding of what to look for as you go onto verify your access control policies with Security Policy Tool. In addition to this document there are other resources located in the Learning Center in your My account page that will help you start leveraging Security Policy Tool to prevent access control leaks, today!

If you have not yet, download Security Policy Tool – Lite Version for FREE now! Close the door the Access Control Leaks and save time and cost creating, modeling, testing, and verifying your access control policies, today.

Click here to begin securing your policies now → Lite Version.

INFOBEYOND

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.