

Healthcare 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 attached 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 Healthcare 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. 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 healthcare example contains two policies (ManagerPolicy & DoctorPolicy). The Attribute /Attribute Values include 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.

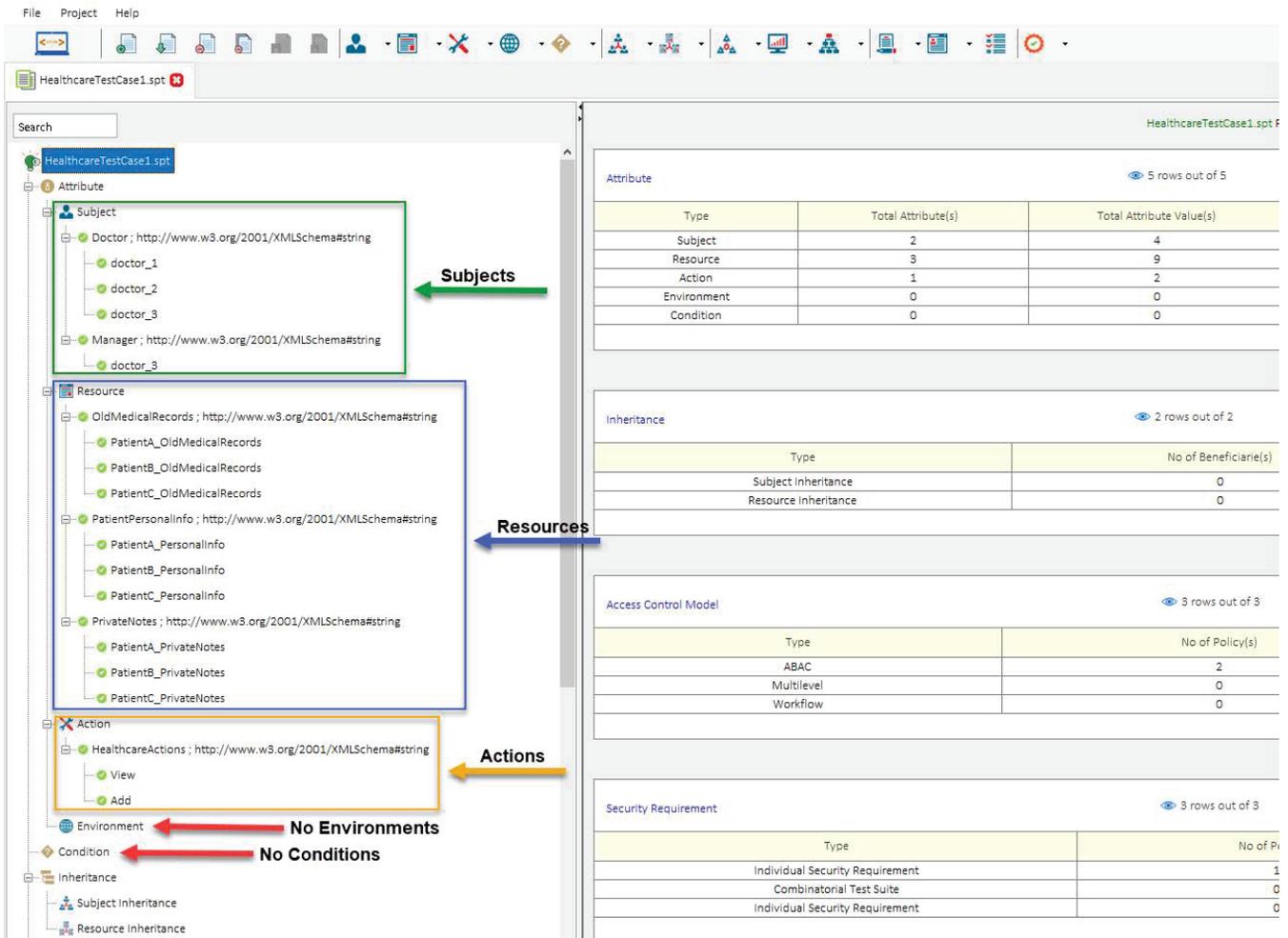


Fig. 1. Test Case 1

3 MODELING YOUR POLICY – TEST CASE 1 (RULE CONFLICT)

Now that we have entered our attributes we can model our two policies (ManagerPolicy & DoctorPolicy). 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:

ManagerPolicy:

- (Subject = Any Value & Manager = doctor_3, View, PatientA_OldMedicalRecords) → Permit
- (Subject = Any Value & Manager = doctor_3, View, PatientB_OldMedicalRecords) →Permit
- (Subject = Any Value & Manager = doctor_3, View, PatientC_OldMedicalRecords) →Permit
- (Subject = Any Value & Manager = doctor_3, View, PatientA_PersonalInfo) →Permit
- (Subject = Any Value & Manager = doctor_3, View, PatientB_PersonalInfo) →Permit
- (Subject = Any Value & Manager = doctor_3, View, PatientC_PersonalInfo) →Permit
- (Subject = Any Value & Manager = doctor_3, View, PatientA_PrivateNotes) →Deny
- (Subject = Any Value & Manager = doctor_3, View, PatientB_PrivateNotes) →Deny
- (Subject = Any Value & Manager = doctor_3, View, PatientC_PrivateNotes) →Deny

DoctorPolicy:

- (Subject = Any Value & Doctor = doctor_1, View, PatientA_OldMedicalRecords) →Permit
- (Subject = Any Value & Doctor = doctor_1, View, PatientB_OldMedicalRecords) →Deny
- (Subject = Any Value & Doctor = doctor_1, View, PatientC_OldMedicalRecords) →Deny
- (Subject = Any Value & Doctor = doctor_1, View, PatientA_PersonalInfo) →Deny
- (Subject = Any Value & Doctor = doctor_1, View, PatientB_PersonalInfo) →Deny
- (Subject = Any Value & Doctor = doctor_1, View, PatientC_PersonalInfo) →Deny
- (Subject = Any Value & Doctor = doctor_1, View, PatientA_PrivateNotes) →Permit
- (Subject = Any Value & Doctor = doctor_1, Add, PatientA_PrivateNotes) →Permit

(Subject = Any Value & Doctor = doctor_1, Action: Any, PatientB_PrivateNotes) →Deny
 (Subject = Any Value & Doctor = doctor_1, Action: Any, PatientC_PrivateNotes) →Deny
 (Subject = Any Value & Doctor = doctor_2, View, PatientA_OldMedicalRecords) →Deny
 (Subject = Any Value & Doctor = doctor_2, View, PatientB_OldMedicalRecords) →Permit
 (Subject = Any Value & Doctor = doctor_2, View, PatientC_OldMedicalRecords) →Deny
 (Subject = Any Value & Doctor = doctor_2, View, PatientA_PersonalInfo) →Deny
 (Subject = Any Value & Doctor = doctor_2, View, PatientB_PersonalInfo) →Deny
 (Subject = Any Value & Doctor = doctor_2, View, PatientC_PersonalInfo) →Deny
 (Subject = Any Value & Doctor = doctor_2, Action: Any, PatientA_PrivateNotes) →Deny
 (Subject = Any Value & Doctor = doctor_2, View, PatientB_PrivateNotes) →Permit
 (Subject = Any Value & Doctor = doctor_2, Add, PatientB_PrivateNotes) →Permit
 (Subject = Any Value & Doctor = doctor_2, Action: Any, PatientC_PrivateNotes) →Deny
 (Subject = Any Value & Doctor = doctor_3, View, PatientA_OldMedicalRecords) →Deny
 (Subject = Any Value & Doctor = doctor_3, View, PatientB_OldMedicalRecords) →Deny
 (Subject = Any Value & Doctor = doctor_3, View, PatientC_OldMedicalRecords) →Permit
 (Subject = Any Value & Doctor = doctor_3, View, PatientA_PersonalInfo) →Deny
 (Subject = Any Value & Doctor = doctor_3, View, PatientB_PersonalInfo) →Deny
 (Subject = Any Value & Doctor = doctor_3, View, PatientC_PersonalInfo) →Deny
 (Subject = Any Value & Doctor = doctor_3, Action: Any, PatientA_PrivateNotes) →Deny
 (Subject = Any Value & Doctor = doctor_3, Action: Any, PatientB_PrivateNotes) →Deny
 (Subject = Any Value & Doctor = doctor_3, View, PatientC_PrivateNotes) →Permit
 (Subject = Any Value & Doctor = doctor_3, Add, PatientC_PrivateNotes) →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: HealthcareTestCase1 and these policies will have been already created for you.

Model	Policy Name	Rule Combination Algorithm	Policy Enforcement Algorithm	No. of Rule(s)	Time Created	Last Modified
ABAC	ManagerPolicy	Deny-overrides	Deny Biased	9	June 13, 2018 12:29:42	June 13, 2018 12:29:42

Sequence No	Subject	Resource	Action	Environment	Condition	Decision	Inheritance Relation
1	Subject = Any Value & Manager = doctor_3	OldMedicalRecords = PatientA_OldMedicalRecords	HealthcareActions = View	Environment = Any Value	Condition = Any Value	Permit	Originated
2	Subject = Any Value & Manager = doctor_3	OldMedicalRecords = PatientB_OldMedicalRecords	HealthcareActions = View	Environment = Any Value	Condition = Any Value	Permit	Originated
3	Subject = Any Value & Manager = doctor_3	OldMedicalRecords = PatientC_OldMedicalRecords	HealthcareActions = View	Environment = Any Value	Condition = Any Value	Permit	Originated
4	Subject = Any Value & Manager = doctor_3	PatientPersonalInfo = PatientA_PersonalInfo	HealthcareActions = View	Environment = Any Value	Condition = Any Value	Permit	Originated
5	Subject = Any Value & Manager = doctor_3	PatientPersonalInfo = PatientB_PersonalInfo	HealthcareActions = View	Environment = Any Value	Condition = Any Value	Permit	Originated
6	Subject = Any Value & Manager = doctor_3	PatientPersonalInfo = PatientC_PersonalInfo	HealthcareActions = View	Environment = Any Value	Condition = Any Value	Permit	Originated
7	Subject = Any Value & Manager = doctor_3	PrivateNotes = PatientA_PrivateNotes	HealthcareActions = View	Environment = Any Value	Condition = Any Value	Deny	Originated
8	Subject = Any Value & Manager = doctor_3	PrivateNotes = PatientB_PrivateNotes	HealthcareActions = View	Environment = Any Value	Condition = Any Value	Deny	Originated
9	Subject = Any Value & Manager = doctor_3	PrivateNotes = PatientC_PrivateNotes	HealthcareActions = View	Environment = Any Value	Condition = Any Value	Deny	Originated

Fig. 2. ManagerPolicy

DoctorPolicy 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	DoctorPolicy	Deny-overrides	Deny Biased	30	June 13, 2018 12:33:14	June 13, 2018 12:33:14				

Rule (s) defined with selected policy (DoctorPolicy):								30 rows out of 30	Search	🔍	📄
Sequence No	Subject	Resource	Action	Environment	Condition	Decision	Inheritance Relation				
1	Subject = Any Value & Doctor = doctor_1	OldMedicalRecords = PatientA_OldMedicalRecords	HealthcareActions = View	Environment = Any Value	Condition = Any Value	Permit	Originated				
2	Subject = Any Value & Doctor = doctor_1	OldMedicalRecords = PatientB_OldMedicalRecords	HealthcareActions = View	Environment = Any Value	Condition = Any Value	Deny	Originated				
3	Subject = Any Value & Doctor = doctor_1	OldMedicalRecords = PatientC_OldMedicalRecords	HealthcareActions = View	Environment = Any Value	Condition = Any Value	Deny	Originated				
4	Subject = Any Value & Doctor = doctor_1	PatientPersonallInfo = PatientA_PersonalInfo	HealthcareActions = View	Environment = Any Value	Condition = Any Value	Deny	Originated				
5	Subject = Any Value & Doctor = doctor_1	PatientPersonallInfo = PatientB_PersonalInfo	HealthcareActions = View	Environment = Any Value	Condition = Any Value	Deny	Originated				
6	Subject = Any Value & Doctor = doctor_1	PatientPersonallInfo = PatientC_PersonalInfo	HealthcareActions = View	Environment = Any Value	Condition = Any Value	Deny	Originated				
7	Subject = Any Value & Doctor = doctor_1	PrivateNotes = PatientA_PrivateNotes	HealthcareActions = View	Environment = Any Value	Condition = Any Value	Permit	Originated				
8	Subject = Any Value & Doctor = doctor_1	PrivateNotes = PatientA_PrivateNotes	HealthcareActions = Add	Environment = Any Value	Condition = Any Value	Permit	Originated				
9	Subject = Any Value & Doctor = doctor_1	PrivateNotes = PatientB_PrivateNotes	Action = Any Value	Environment = Any Value	Condition = Any Value	Deny	Originated				
10	Subject = Any Value & Doctor = doctor_1	PrivateNotes = PatientC_PrivateNotes	Action = Any Value	Environment = Any Value	Condition = Any Value	Deny	Originated				
11	Subject = Any Value & Doctor = doctor_2	OldMedicalRecords = PatientA_OldMedicalRecords	HealthcareActions = View	Environment = Any Value	Condition = Any Value	Deny	Originated				
12	Subject = Any Value & Doctor = doctor_2	OldMedicalRecords = PatientB_OldMedicalRecords	HealthcareActions = View	Environment = Any Value	Condition = Any Value	Permit	Originated				
13	Subject = Any Value & Doctor = doctor_2	OldMedicalRecords = PatientC_OldMedicalRecords	HealthcareActions = View	Environment = Any Value	Condition = Any Value	Deny	Originated				
14	Subject = Any Value & Doctor = doctor_2	PatientPersonallInfo = PatientA_PersonalInfo	HealthcareActions = View	Environment = Any Value	Condition = Any Value	Deny	Originated				
15	Subject = Any Value & Doctor = doctor_2	PatientPersonallInfo = PatientB_PersonalInfo	HealthcareActions = View	Environment = Any Value	Condition = Any Value	Deny	Originated				
16	Subject = Any Value & Doctor = doctor_2	PatientPersonallInfo = PatientC_PersonalInfo	HealthcareActions = View	Environment = Any Value	Condition = Any Value	Deny	Originated				
17	Subject = Any Value & Doctor = doctor_2	PrivateNotes = PatientA_PrivateNotes	Action = Any Value	Environment = Any Value	Condition = Any Value	Deny	Originated				
18	Subject = Any Value & Doctor = doctor_2	PrivateNotes = PatientB_PrivateNotes	HealthcareActions = View	Environment = Any Value	Condition = Any Value	Permit	Originated				
19	Subject = Any Value & Doctor = doctor_2	PrivateNotes = PatientB_PrivateNotes	HealthcareActions = Add	Environment = Any Value	Condition = Any Value	Permit	Originated				
20	Subject = Any Value & Doctor = doctor_2	PrivateNotes = PatientC_PrivateNotes	Action = Any Value	Environment = Any Value	Condition = Any Value	Deny	Originated				
21	Subject = Any Value & Doctor = doctor_3	OldMedicalRecords = PatientA_OldMedicalRecords	HealthcareActions = View	Environment = Any Value	Condition = Any Value	Deny	Originated				
22	Subject = Any Value & Doctor = doctor_3	OldMedicalRecords = PatientB_OldMedicalRecords	HealthcareActions = View	Environment = Any Value	Condition = Any Value	Deny	Originated				
23	Subject = Any Value & Doctor = doctor_3	OldMedicalRecords = PatientC_OldMedicalRecords	HealthcareActions = View	Environment = Any Value	Condition = Any Value	Permit	Originated				
24	Subject = Any Value & Doctor = doctor_3	PatientPersonallInfo = PatientA_PersonalInfo	HealthcareActions = View	Environment = Any Value	Condition = Any Value	Deny	Originated				
25	Subject = Any Value & Doctor = doctor_3	PatientPersonallInfo = PatientB_PersonalInfo	HealthcareActions = View	Environment = Any Value	Condition = Any Value	Deny	Originated				
26	Subject = Any Value & Doctor = doctor_3	PatientPersonallInfo = PatientC_PersonalInfo	HealthcareActions = View	Environment = Any Value	Condition = Any Value	Deny	Originated				
27	Subject = Any Value & Doctor = doctor_3	PrivateNotes = PatientA_PrivateNotes	Action = Any Value	Environment = Any Value	Condition = Any Value	Deny	Originated				
28	Subject = Any Value & Doctor = doctor_3	PrivateNotes = PatientB_PrivateNotes	Action = Any Value	Environment = Any Value	Condition = Any Value	Deny	Originated				
29	Subject = Any Value & Doctor = doctor_3	PrivateNotes = PatientC_PrivateNotes	HealthcareActions = View	Environment = Any Value	Condition = Any Value	Permit	Originated				
30	Subject = Any Value & Doctor = doctor_3	PrivateNotes = PatientC_PrivateNotes	HealthcareActions = Add	Environment = Any Value	Condition = Any Value	Permit	Originated				

Fig. 3. DoctorPolicy

4 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 security requirements as follows.

Individual Security Requirements:

(Manager = doctor_3 & Doctor = doctor_3) & (Action = View) & (PrivateNotes = PatientC_PrivateNotes) → decision = Permit
 (Manager = doctor_3 & Doctor = doctor_3) & (Action = View) & (PatientPersonallInfo = PatientC_PersonalInfo)
 → decision = Permit

After entering the rules above your individual security requirements should look like the screenshot below. If you did not create your own Project File you can simply open Security Policy Tool – Project File: HealthcareTestCase1 and these requirements 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	2				

Security Requirement (s) defined under selected Requirement Schema (Test Case 1):							2 rows out of 2	Search	🔍	📄
Sequence No	Subject	Resource	Action	Environment	Condition	Decision				
1	Doctor = doctor_3 & Manager = doctor_3	PrivateNotes = PatientC_PrivateNotes	HealthcareActions = View	Environment = Any Value	Condition = Any Value	Permit				
2	Doctor = doctor_3 & Manager = doctor_3	PatientPersonallInfo = PatientC_PersonalInfo	HealthcareActions = View	Environment = Any Value	Condition = Any Value	Permit				

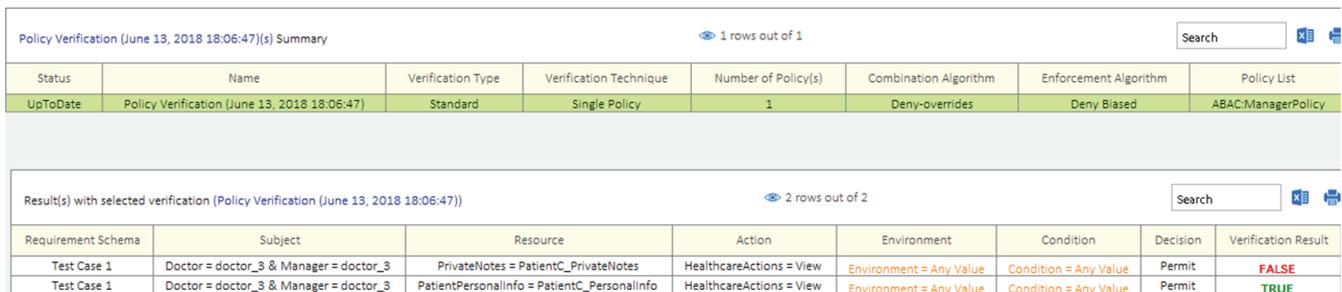
Fig. 4. Individual Security Requirements

5 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 at this hospital has a role of both doctor and manager at the facility. Due to this, the individual is assigned both (Doctor: doctor_3 and Manager: doctor_3) attribute values by the system during access evaluation. In the ManagerPolicy it defines that managers can view PatientPersonalInfo but cannot view PrivateNotes. However, in the DoctorPolicy the opposite has been defined (e.g., can view PrivateNotes; cannot view PatientPersonalInfo).

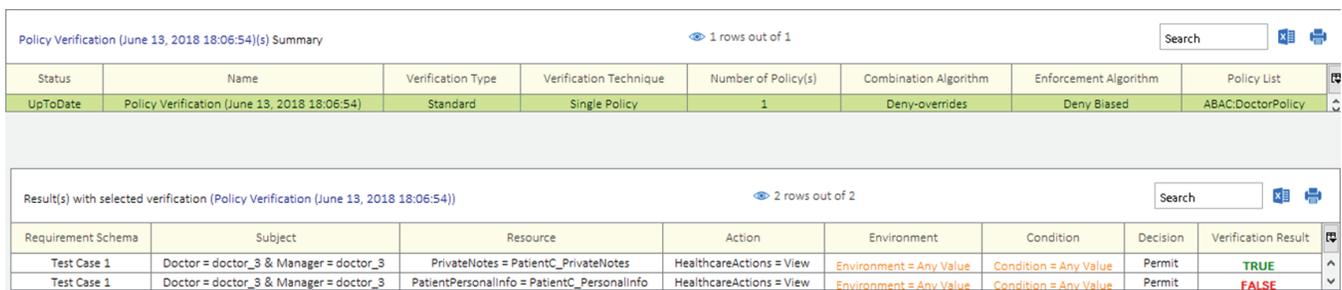
Next, we will run two “Single Policy” Verifications to reveal the Rule Conflict that is present in our policies. To do this, we will select ManagerPolicy and Test Case 1 (security requirement) as a Single Policy Verification and also choose DoctorPolicy and Test Case 1 (security requirement) as a Single Policy Verification and analyze our two verification results. Again, this will have already been done for you if you open Project File: HealthcareTestCase1.



Status	Name	Verification Type	Verification Technique	Number of Policy(s)	Combination Algorithm	Enforcement Algorithm	Policy List
UpToDate	Policy Verification (June 13, 2018 18:06:47)	Standard	Single Policy	1	Deny-overrides	Deny Biased	ABAC:ManagerPolicy

Requirement Schema	Subject	Resource	Action	Environment	Condition	Decision	Verification Result
Test Case 1	Doctor = doctor_3 & Manager = doctor_3	PrivateNotes = PatientC_PrivateNotes	HealthcareActions = View	Environment = Any Value	Condition = Any Value	Permit	FALSE
Test Case 1	Doctor = doctor_3 & Manager = doctor_3	PatientPersonalInfo = PatientC_PersonalInfo	HealthcareActions = View	Environment = Any Value	Condition = Any Value	Permit	TRUE

Fig. 5. ManagerPolicy x Test Case 1



Status	Name	Verification Type	Verification Technique	Number of Policy(s)	Combination Algorithm	Enforcement Algorithm	Policy List
UpToDate	Policy Verification (June 13, 2018 18:06:54)	Standard	Single Policy	1	Deny-overrides	Deny Biased	ABAC:DoctorPolicy

Requirement Schema	Subject	Resource	Action	Environment	Condition	Decision	Verification Result
Test Case 1	Doctor = doctor_3 & Manager = doctor_3	PrivateNotes = PatientC_PrivateNotes	HealthcareActions = View	Environment = Any Value	Condition = Any Value	Permit	TRUE
Test Case 1	Doctor = doctor_3 & Manager = doctor_3	PatientPersonalInfo = PatientC_PersonalInfo	HealthcareActions = View	Environment = Any Value	Condition = Any Value	Permit	FALSE

Fig. 6. DoctorPolicy x Test Case 1

As you can see from our verification results our policies are both Permitting and Denying the individual (Doctor = doctor_3/Manager = doctor_3) from viewing PatientC_PersonalInfo and PatientC_Private Notes which is known as a Rule Conflict error.

6 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 all (4) of our specific Results (DoctorPolicyxTestCase1: False;True & ManagerPolicyxTestCase1: True;False) and see which Rules have “Match Results”.

See the screenshots below of our two Policies Match Results to discover which specific rules are related to our Verification Results (e.g., False, True).

Policy Verification (June 13, 2018 18:06:47)(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 13, 2018 18:06:47)	Standard	Single Policy	1	Deny-overrides	Deny Biased	ABAC:ManagerPolicy

Result(s) with selected verification (Policy Verification (June 13, 2018 18:06:47)) 2 rows out of 2

Requirement Schema	Subject	Resource	Action	Environment	Condition	Decision	Verification Result
Test Case 1	Doctor = doctor_3 & Manager = doctor_3	PrivateNotes = PatientC_PrivateNotes	HealthcareActions = View	Environment = Any Value	Condition = Any Value	Permit	FALSE
Test Case 1	Doctor = doctor_3 & Manager = doctor_3	PatientPersonalInfo = PatientC_PersonalInfo	HealthcareActions = View	Environment = Any Value	Condition = Any Value	Permit	TRUE

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

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

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

Sequence No	Subject	Resource	Action	Environment	Condition	Decision	Inheritance Relation	Match Result
1	Subject = Any Value & Manager = doctor_3	OldMedicalRecords = PatientA_OldMedicalRecords	HealthcareActions = View	Environment = Any Value	Condition = Any Value	Permit	Originated	Not Applicable
2	Subject = Any Value & Manager = doctor_3	OldMedicalRecords = PatientB_OldMedicalRecords	HealthcareActions = View	Environment = Any Value	Condition = Any Value	Permit	Originated	Not Applicable
3	Subject = Any Value & Manager = doctor_3	OldMedicalRecords = PatientC_OldMedicalRecords	HealthcareActions = View	Environment = Any Value	Condition = Any Value	Permit	Originated	Not Applicable
4	Subject = Any Value & Manager = doctor_3	PatientPersonalInfo = PatientA_PersonalInfo	HealthcareActions = View	Environment = Any Value	Condition = Any Value	Permit	Originated	Not Applicable
5	Subject = Any Value & Manager = doctor_3	PatientPersonalInfo = PatientB_PersonalInfo	HealthcareActions = View	Environment = Any Value	Condition = Any Value	Permit	Originated	Not Applicable
6	Subject = Any Value & Manager = doctor_3	PatientPersonalInfo = PatientC_PersonalInfo	HealthcareActions = View	Environment = Any Value	Condition = Any Value	Permit	Originated	Not Applicable
7	Subject = Any Value & Manager = doctor_3	PrivateNotes = PatientA_PrivateNotes	HealthcareActions = View	Environment = Any Value	Condition = Any Value	Deny	Originated	Not Applicable
8	Subject = Any Value & Manager = doctor_3	PrivateNotes = PatientB_PrivateNotes	HealthcareActions = View	Environment = Any Value	Condition = Any Value	Deny	Originated	Not Applicable
9	Subject = Any Value & Manager = doctor_3	PrivateNotes = PatientC_PrivateNotes	HealthcareActions = View	Environment = Any Value	Condition = Any Value	Deny	Originated	Deny

Fig. 7. ManagerPolicy: Match Results (PatientC_PrivateNotes)

Policy Verification (June 13, 2018 18:06:47)(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 13, 2018 18:06:47)	Standard	Single Policy	1	Deny-overrides	Deny Biased	ABAC:ManagerPolicy

Result(s) with selected verification (Policy Verification (June 13, 2018 18:06:47)) 2 rows out of 2

Requirement Schema	Subject	Resource	Action	Environment	Condition	Decision	Verification Result
Test Case 1	Doctor = doctor_3 & Manager = doctor_3	PrivateNotes = PatientC_PrivateNotes	HealthcareActions = View	Environment = Any Value	Condition = Any Value	Permit	FALSE
Test Case 1	Doctor = doctor_3 & Manager = doctor_3	PatientPersonalInfo = PatientC_PersonalInfo	HealthcareActions = View	Environment = Any Value	Condition = Any Value	Permit	TRUE

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

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

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

Sequence No	Subject	Resource	Action	Environment	Condition	Decision	Inheritance Relation	Match Result
1	Subject = Any Value & Manager = doctor_3	OldMedicalRecords = PatientA_OldMedicalRecords	HealthcareActions = View	Environment = Any Value	Condition = Any Value	Permit	Originated	Not Applicable
2	Subject = Any Value & Manager = doctor_3	OldMedicalRecords = PatientB_OldMedicalRecords	HealthcareActions = View	Environment = Any Value	Condition = Any Value	Permit	Originated	Not Applicable
3	Subject = Any Value & Manager = doctor_3	OldMedicalRecords = PatientC_OldMedicalRecords	HealthcareActions = View	Environment = Any Value	Condition = Any Value	Permit	Originated	Not Applicable
4	Subject = Any Value & Manager = doctor_3	PatientPersonalInfo = PatientA_PersonalInfo	HealthcareActions = View	Environment = Any Value	Condition = Any Value	Permit	Originated	Not Applicable
5	Subject = Any Value & Manager = doctor_3	PatientPersonalInfo = PatientB_PersonalInfo	HealthcareActions = View	Environment = Any Value	Condition = Any Value	Permit	Originated	Not Applicable
6	Subject = Any Value & Manager = doctor_3	PatientPersonalInfo = PatientC_PersonalInfo	HealthcareActions = View	Environment = Any Value	Condition = Any Value	Permit	Originated	Permit
7	Subject = Any Value & Manager = doctor_3	PrivateNotes = PatientA_PrivateNotes	HealthcareActions = View	Environment = Any Value	Condition = Any Value	Deny	Originated	Not Applicable
8	Subject = Any Value & Manager = doctor_3	PrivateNotes = PatientB_PrivateNotes	HealthcareActions = View	Environment = Any Value	Condition = Any Value	Deny	Originated	Not Applicable
9	Subject = Any Value & Manager = doctor_3	PrivateNotes = PatientC_PrivateNotes	HealthcareActions = View	Environment = Any Value	Condition = Any Value	Deny	Originated	Not Applicable

Fig. 8. ManagerPolicy: Match Results (PatientC_PersonalInfo)

Status	Name	Verification Type	Verification Technique	Number of Policy(s)	Combination Algorithm	Enforcement Algorithm	Policy List		
UpToDate	Policy Verification (June 13, 2018 18:06:54)	Standard	Single Policy	1	Deny-overrides	Deny Biased	ABAC:DoctorPolicy		

Requirement Schema	Subject	Resource	Action	Environment	Condition	Decision	Verification Result
Test Case 1	Doctor = doctor_3 & Manager = doctor_3	PrivateNotes = PatientC_PrivateNotes	HealthcareActions = View	Environment = Any Value	Condition = Any Value	Permit	TRUE
Test Case 1	Doctor = doctor_3 & Manager = doctor_3	PatientPersonalInfo = PatientC_PersonalInfo	HealthcareActions = View	Environment = Any Value	Condition = Any Value	Permit	FALSE

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

Sequence No	Subject	Resource	Action	Environment	Condition	Decision	Inheritance Relation	Match Result
14	Subject = Any Value & Doctor = doctor_2	PatientPersonalInfo = PatientA_PersonalInfo	HealthcareActions = View	Environment = Any Value	Condition = Any Value	Deny	Originated	Not Applicable
15	Subject = Any Value & Doctor = doctor_2	PatientPersonalInfo = PatientB_PersonalInfo	HealthcareActions = View	Environment = Any Value	Condition = Any Value	Deny	Originated	Not Applicable
16	Subject = Any Value & Doctor = doctor_2	PatientPersonalInfo = PatientC_PersonalInfo	HealthcareActions = View	Environment = Any Value	Condition = Any Value	Deny	Originated	Not Applicable
17	Subject = Any Value & Doctor = doctor_2	PrivateNotes = PatientA_PrivateNotes	Action = Any Value	Environment = Any Value	Condition = Any Value	Deny	Originated	Not Applicable
18	Subject = Any Value & Doctor = doctor_2	PrivateNotes = PatientB_PrivateNotes	HealthcareActions = View	Environment = Any Value	Condition = Any Value	Permit	Originated	Not Applicable
19	Subject = Any Value & Doctor = doctor_2	PrivateNotes = PatientB_PrivateNotes	HealthcareActions = Add	Environment = Any Value	Condition = Any Value	Permit	Originated	Not Applicable
20	Subject = Any Value & Doctor = doctor_2	PrivateNotes = PatientC_PrivateNotes	Action = Any Value	Environment = Any Value	Condition = Any Value	Deny	Originated	Not Applicable
21	Subject = Any Value & Doctor = doctor_3	OldMedicalRecords = PatientA_OldMedicalRecords	HealthcareActions = View	Environment = Any Value	Condition = Any Value	Deny	Originated	Not Applicable
22	Subject = Any Value & Doctor = doctor_3	OldMedicalRecords = PatientB_OldMedicalRecords	HealthcareActions = View	Environment = Any Value	Condition = Any Value	Deny	Originated	Not Applicable
23	Subject = Any Value & Doctor = doctor_3	OldMedicalRecords = PatientC_OldMedicalRecords	HealthcareActions = View	Environment = Any Value	Condition = Any Value	Permit	Originated	Not Applicable
24	Subject = Any Value & Doctor = doctor_3	PatientPersonalInfo = PatientA_PersonalInfo	HealthcareActions = View	Environment = Any Value	Condition = Any Value	Deny	Originated	Not Applicable
25	Subject = Any Value & Doctor = doctor_3	PatientPersonalInfo = PatientB_PersonalInfo	HealthcareActions = View	Environment = Any Value	Condition = Any Value	Deny	Originated	Not Applicable
26	Subject = Any Value & Doctor = doctor_3	PatientPersonalInfo = PatientC_PersonalInfo	HealthcareActions = View	Environment = Any Value	Condition = Any Value	Deny	Originated	Not Applicable
27	Subject = Any Value & Doctor = doctor_3	PrivateNotes = PatientA_PrivateNotes	Action = Any Value	Environment = Any Value	Condition = Any Value	Deny	Originated	Not Applicable
28	Subject = Any Value & Doctor = doctor_3	PrivateNotes = PatientB_PrivateNotes	Action = Any Value	Environment = Any Value	Condition = Any Value	Deny	Originated	Not Applicable
29	Subject = Any Value & Doctor = doctor_3	PrivateNotes = PatientC_PrivateNotes	HealthcareActions = View	Environment = Any Value	Condition = Any Value	Permit	Originated	Permit
30	Subject = Any Value & Doctor = doctor_3	PrivateNotes = PatientC_PrivateNotes	HealthcareActions = Add	Environment = Any Value	Condition = Any Value	Permit	Originated	Not Applicable

Fig. 9. DoctorPolicy: Match Results (PatientC_PrivateNotes)

Status	Name	Verification Type	Verification Technique	Number of Policy(s)	Combination Algorithm	Enforcement Algorithm	Policy List
UpToDate	Policy Verification (June 13, 2018 18:06:54)	Standard	Single Policy	1	Deny-overrides	Deny Biased	ABAC:DoctorPolicy

Requirement Schema	Subject	Resource	Action	Environment	Condition	Decision	Verification Result
Test Case 1	Doctor = doctor_3 & Manager = doctor_3	PrivateNotes = PatientC_PrivateNotes	HealthcareActions = View	Environment = Any Value	Condition = Any Value	Permit	TRUE
Test Case 1	Doctor = doctor_3 & Manager = doctor_3	PatientPersonalInfo = PatientC_PersonalInfo	HealthcareActions = View	Environment = Any Value	Condition = Any Value	Permit	FALSE

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

Sequence No	Subject	Resource	Action	Environment	Condition	Decision	Inheritance Relation	Match Result
14	Subject = Any Value & Doctor = doctor_2	PatientPersonalInfo = PatientA_PersonalInfo	HealthcareActions = View	Environment = Any Value	Condition = Any Value	Deny	Originated	Not Applicable
15	Subject = Any Value & Doctor = doctor_2	PatientPersonalInfo = PatientB_PersonalInfo	HealthcareActions = View	Environment = Any Value	Condition = Any Value	Deny	Originated	Not Applicable
16	Subject = Any Value & Doctor = doctor_2	PatientPersonalInfo = PatientC_PersonalInfo	HealthcareActions = View	Environment = Any Value	Condition = Any Value	Deny	Originated	Not Applicable
17	Subject = Any Value & Doctor = doctor_2	PrivateNotes = PatientA_PrivateNotes	Action = Any Value	Environment = Any Value	Condition = Any Value	Deny	Originated	Not Applicable
18	Subject = Any Value & Doctor = doctor_2	PrivateNotes = PatientB_PrivateNotes	HealthcareActions = View	Environment = Any Value	Condition = Any Value	Permit	Originated	Not Applicable
19	Subject = Any Value & Doctor = doctor_2	PrivateNotes = PatientB_PrivateNotes	HealthcareActions = Add	Environment = Any Value	Condition = Any Value	Permit	Originated	Not Applicable
20	Subject = Any Value & Doctor = doctor_2	PrivateNotes = PatientC_PrivateNotes	Action = Any Value	Environment = Any Value	Condition = Any Value	Deny	Originated	Not Applicable
21	Subject = Any Value & Doctor = doctor_3	OldMedicalRecords = PatientA_OldMedicalRecords	HealthcareActions = View	Environment = Any Value	Condition = Any Value	Deny	Originated	Not Applicable
22	Subject = Any Value & Doctor = doctor_3	OldMedicalRecords = PatientB_OldMedicalRecords	HealthcareActions = View	Environment = Any Value	Condition = Any Value	Deny	Originated	Not Applicable
23	Subject = Any Value & Doctor = doctor_3	OldMedicalRecords = PatientC_OldMedicalRecords	HealthcareActions = View	Environment = Any Value	Condition = Any Value	Permit	Originated	Not Applicable
24	Subject = Any Value & Doctor = doctor_3	PatientPersonalInfo = PatientA_PersonalInfo	HealthcareActions = View	Environment = Any Value	Condition = Any Value	Deny	Originated	Not Applicable
25	Subject = Any Value & Doctor = doctor_3	PatientPersonalInfo = PatientB_PersonalInfo	HealthcareActions = View	Environment = Any Value	Condition = Any Value	Deny	Originated	Not Applicable
26	Subject = Any Value & Doctor = doctor_3	PatientPersonalInfo = PatientC_PersonalInfo	HealthcareActions = View	Environment = Any Value	Condition = Any Value	Deny	Originated	Deny
27	Subject = Any Value & Doctor = doctor_3	PrivateNotes = PatientA_PrivateNotes	Action = Any Value	Environment = Any Value	Condition = Any Value	Deny	Originated	Not Applicable
28	Subject = Any Value & Doctor = doctor_3	PrivateNotes = PatientB_PrivateNotes	Action = Any Value	Environment = Any Value	Condition = Any Value	Deny	Originated	Not Applicable
29	Subject = Any Value & Doctor = doctor_3	PrivateNotes = PatientC_PrivateNotes	HealthcareActions = View	Environment = Any Value	Condition = Any Value	Permit	Originated	Not Applicable
30	Subject = Any Value & Doctor = doctor_3	PrivateNotes = PatientC_PrivateNotes	HealthcareActions = Add	Environment = Any Value	Condition = Any Value	Permit	Originated	Not Applicable

Fig. 10. DoctorPolicy: Match Results (PatientC_PersonalInfo)

Now that we have pinpointed our (4) 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, we are going to modify Rule 9 in the ManagerPolicy to Permit “Manager = doctor_3” to View Patient C_PrivateNotes and modify Rule 26 in the DoctorPolicy to Permit “Doctor = doctor_3” to View Patient C_PersonalInfo which will in turn resolve the Rule Conflict. For this example, we are making an exception for this individual (doctor_3) because they are required to be able to view PatientPersonalInfo and PrivateNotes to perform their job as a Manager and a Doctor.

ManagerPolicy: Modify (1) Rule:

(Rule No. = 9) → (Subject = Any Value & Manager = doctor_3) → (Action = View) → (Resource = PatientC_PrivateNotes) → decision = Permit

DoctorPolicy: Modify (1) Rule:

(Rule No. = 26) → (Subject = Any Value & Doctor = doctor_3) → (Action = View) → (Resource = PatientC_PersonalInfo) → decision = Permit

9	Subject = Any Value & Manager = doctor_3	PrivateNotes = PatientC_PrivateNotes	HealthcareActions = View	Environment = Any Value	Condition = Any Value	Permit	Originated
---	--	--------------------------------------	--------------------------	-------------------------	-----------------------	--------	------------

Fig. 11. ManagerPolicy: Modified Rule (9)

26	Subject=Any Value & Doctor=doctor_3	PatientPersonalInfo = PatientC_PersonalInfo	HealthcareActions = View	Environment=Any Value	Condition=Any Value	Permit	Originated
----	-------------------------------------	---	--------------------------	-----------------------	---------------------	--------	------------

Fig. 12. DoctorPolicy: Modified Rule (26)

Which then when we “Refresh” our previous Verification Results we no longer have a Rule Conflict occurring:

Status	Name	Verification Type	Verification Technique	Number of Policy(s)	Combination Algorithm	Enforcement Algorithm	Policy List
UpToDate	Policy Verification (June 20, 2018 12:27:37)	Standard	Single Policy	1	Deny-overrides	Deny Biased	ABAC:ManagerPolicy

Requirement Schema	Subject	Resource	Action	Environment	Condition	Decision	Verification Result
Test Case 1	Doctor = doctor_3 & Manager = doctor_3	PrivateNotes = PatientC_PrivateNotes	HealthcareActions = View	Environment = Any Value	Condition = Any Value	Permit	TRUE
Test Case 1	Doctor = doctor_3 & Manager = doctor_3	PatientPersonalInfo = PatientC_PersonalInfo	HealthcareActions = View	Environment = Any Value	Condition = Any Value	Permit	TRUE

Fig. 13. Updated Results: Manager Policy (No Rule Conflict)

Status	Name	Verification Type	Verification Technique	Number of Policy(s)	Combination Algorithm	Enforcement Algorithm	Policy List
UpToDate	Policy Verification (June 20, 2018 12:28:07)	Standard	Single Policy	1	Deny-overrides	Deny Biased	ABAC:DoctorPolicy

Requirement Schema	Subject	Resource	Action	Environment	Condition	Decision	Verification Result
Test Case 1	Doctor = doctor_3 & Manager = doctor_3	PrivateNotes = PatientC_PrivateNotes	HealthcareActions = View	Environment = Any Value	Condition = Any Value	Permit	TRUE
Test Case 1	Doctor = doctor_3 & Manager = doctor_3	PatientPersonalInfo = PatientC_PersonalInfo	HealthcareActions = View	Environment = Any Value	Condition = Any Value	Permit	TRUE

Fig. 14. Updated Results: Doctor Policy (No Rule Conflict)

7 SETTING UP THE POLICIES – TEST CASE 2 (NOT PROTECTED RESOURCE)

This healthcare example contains two policies (ManagerPolicy & DoctorPolicy). The attributes in this example have been changed slightly from previous Test Case 1. Manager’s attribute value has been changed from “doctor_3” to “manager” and also OldMedicalRecords has gained a new attribute value called “PatientD_OldMedicalRecords.” The Attribute/Attribute Values include in these policies are as shown in Figure 15.

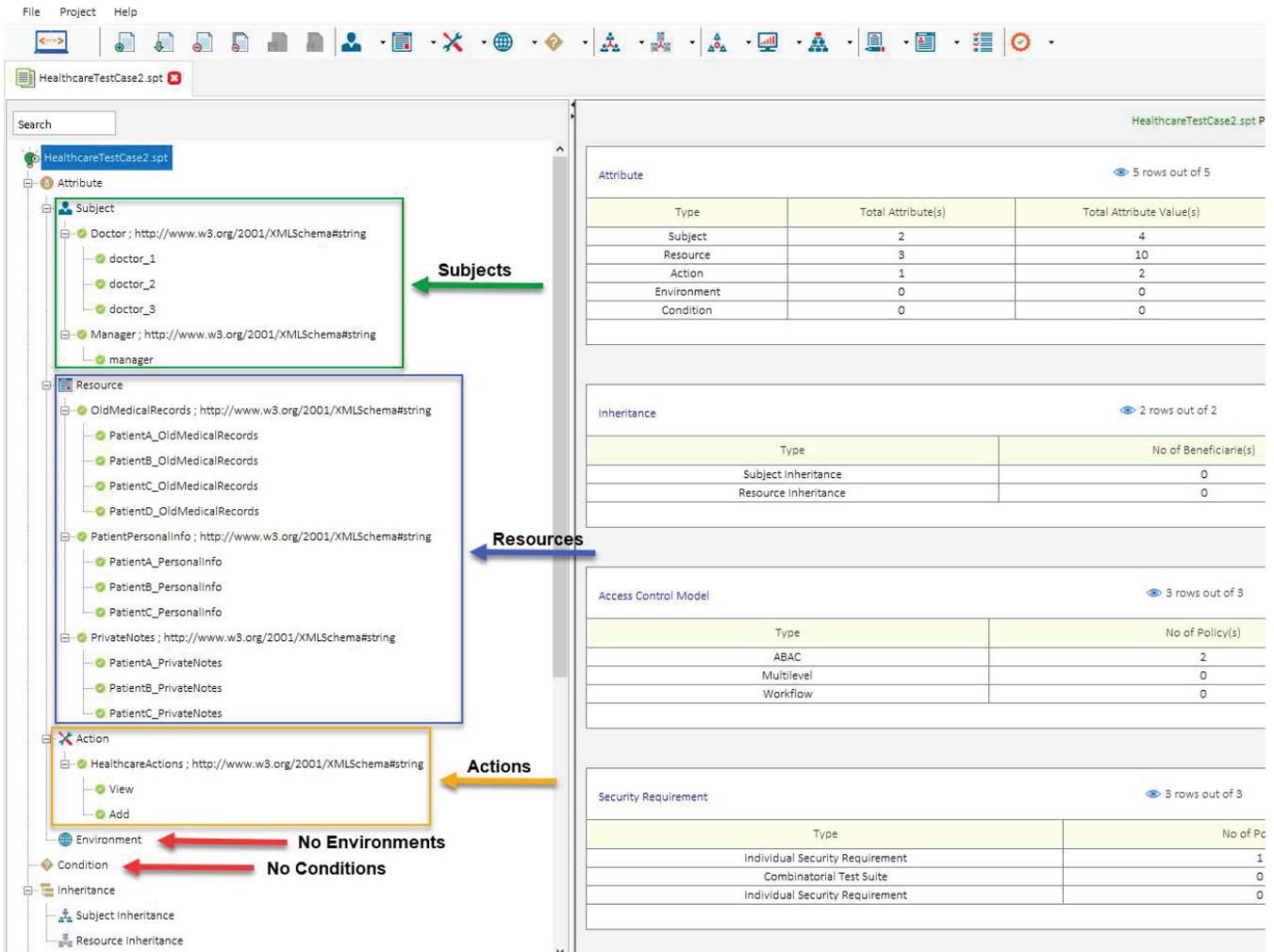


Fig. 15. Test Case 2

8 MODELING YOUR POLICY – TEST CASE 2 (NOT PROTECTED RESOURCE)

Now that we have entered our attributes we can model our two policies (ManagerPolicy & DoctorPolicy). 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:

ManagerPolicy:

- (Manager = manager, View, PatientA_OldMedicalRecords) → Permit
- (Manager = manager, View, PatientB_OldMedicalRecords) → Permit
- (Manager = manager, View, PatientC_OldMedicalRecords) → Permit
- (Manager = manager, View, PatientA_PersonalInfo) → Permit
- (Manager = manager, View, PatientB_PersonalInfo) → Permit
- (Manager = manager, View, PatientC_PersonalInfo) → Permit
- (Manager = manager, View, PatientA_PrivateNotes) → Deny
- (Manager = manager, View, PatientB_PrivateNotes) → Deny
- (Manager = manager, View, PatientC_PrivateNotes) → Deny

DoctorPolicy:

- (Doctor = doctor_1, View, PatientA_OldMedicalRecords) → Permit
- (Doctor = doctor_1, View, PatientB_OldMedicalRecords) → Deny
- (Doctor = doctor_1, View, PatientC_OldMedicalRecords) → Deny
- (Doctor = doctor_1, View, PatientA_PersonalInfo) → Deny
- (Doctor = doctor_1, View, PatientB_PersonalInfo) → Deny
- (Doctor = doctor_1, View, PatientC_PersonalInfo) → Deny
- (Doctor = doctor_1, View, PatientA_PrivateNotes) → Permit

(Doctor = doctor_1, Add, PatientA_PrivateNotes) →Permit
 (Doctor = doctor_1, Action: Any, PatientB_PrivateNotes) →Deny
 (Doctor = doctor_1, Action: Any, PatientC_PrivateNotes) →Deny
 (Doctor = doctor_2, View, PatientA_OldMedicalRecords) →Deny
 (Doctor = doctor_2, View, PatientB_OldMedicalRecords) →Permit
 (Doctor = doctor_2, View, PatientC_OldMedicalRecords) →Deny
 (Doctor = doctor_2, View, PatientA_PersonalInfo) →Deny
 (Doctor = doctor_2, View, PatientB_PersonalInfo) →Deny
 (Doctor = doctor_2, View, PatientC_PersonalInfo) →Deny
 (Doctor = doctor_2, Action: Any, PatientA_PrivateNotes) →Deny
 (Doctor = doctor_2, View, PatientB_PrivateNotes) →Permit
 (Doctor = doctor_2, Add, PatientB_PrivateNotes) →Permit
 (Doctor = doctor_2, Action: Any, PatientC_PrivateNotes) →Deny
 (Doctor = doctor_3, View, PatientA_OldMedicalRecords) →Deny
 (Doctor = doctor_3, View, PatientB_OldMedicalRecords) →Deny
 (Doctor = doctor_3, View, PatientC_OldMedicalRecords) →Permit
 (Doctor = doctor_3, View, PatientA_PersonalInfo) →Deny
 (Doctor = doctor_3, View, PatientB_PersonalInfo) →Deny
 (Doctor = doctor_3, View, PatientC_PersonalInfo) →Deny
 (Doctor = doctor_3, Action: Any, PatientA_PrivateNotes) →Deny
 (Doctor = doctor_3, Action: Any, PatientB_PrivateNotes) →Deny
 (Doctor = doctor_3, View, PatientC_PrivateNotes) →Permit
 (Doctor = doctor_3, Add, PatientC_PrivateNotes) →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: HealthcareTestCase2 and these policies will have been already created for you.

Model	Policy Name	Rule Combination Algorithm	Policy Enforcement Algorithm	No. of Rule(s)	Time Created	Last Modified
ABAC	ManagerPolicy	Deny-overrides	Deny Biased	9	June 13, 2018 12:29:42	June 13, 2018 12:29:42

Sequence No	Subject	Resource	Action	Environment	Condition	Decision	Inheritance Relation
1	Manager = manager	OldMedicalRecords = PatientA_OldMedicalRecords	HealthcareActions = View	Environment = Any Value	Condition = Any Value	Permit	Originated
2	Manager = manager	OldMedicalRecords = PatientB_OldMedicalRecords	HealthcareActions = View	Environment = Any Value	Condition = Any Value	Permit	Originated
3	Manager = manager	OldMedicalRecords = PatientC_OldMedicalRecords	HealthcareActions = View	Environment = Any Value	Condition = Any Value	Permit	Originated
4	Manager = manager	PatientPersonalInfo = PatientA_PersonalInfo	HealthcareActions = View	Environment = Any Value	Condition = Any Value	Permit	Originated
5	Manager = manager	PatientPersonalInfo = PatientB_PersonalInfo	HealthcareActions = View	Environment = Any Value	Condition = Any Value	Permit	Originated
6	Manager = manager	PatientPersonalInfo = PatientC_PersonalInfo	HealthcareActions = View	Environment = Any Value	Condition = Any Value	Permit	Originated
7	Manager = manager	PrivateNotes = PatientA_PrivateNotes	HealthcareActions = View	Environment = Any Value	Condition = Any Value	Deny	Originated
8	Manager = manager	PrivateNotes = PatientB_PrivateNotes	HealthcareActions = View	Environment = Any Value	Condition = Any Value	Deny	Originated
9	Manager = manager	PrivateNotes = PatientC_PrivateNotes	HealthcareActions = View	Environment = Any Value	Condition = Any Value	Deny	Originated

Fig. 16. ManagerPolicy

DoctorPolicy 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	DoctorPolicy	Deny-overrides	Deny Biased	30	June 13, 2018 12:33:14	June 13, 2018 12:33:14				

Rule (s) defined with selected policy (DoctorPolicy):								30 rows out of 30	Search		
Sequence No	Subject	Resource	Action	Environment	Condition	Decision	Inheritance Relation				
1	Doctor = doctor_1	OldMedicalRecords = PatientA_OldMedicalRecords	HealthcareActions = View	Environment = Any Value	Condition = Any Value	Permit	Originated				
2	Doctor = doctor_1	OldMedicalRecords = PatientB_OldMedicalRecords	HealthcareActions = View	Environment = Any Value	Condition = Any Value	Deny	Originated				
3	Doctor = doctor_1	OldMedicalRecords = PatientC_OldMedicalRecords	HealthcareActions = View	Environment = Any Value	Condition = Any Value	Deny	Originated				
4	Doctor = doctor_1	PatientPersonalInfo = PatientA_PersonalInfo	HealthcareActions = View	Environment = Any Value	Condition = Any Value	Deny	Originated				
5	Doctor = doctor_1	PatientPersonalInfo = PatientB_PersonalInfo	HealthcareActions = View	Environment = Any Value	Condition = Any Value	Deny	Originated				
6	Doctor = doctor_1	PatientPersonalInfo = PatientC_PersonalInfo	HealthcareActions = View	Environment = Any Value	Condition = Any Value	Deny	Originated				
7	Doctor = doctor_1	PrivateNotes = PatientA_PrivateNotes	HealthcareActions = View	Environment = Any Value	Condition = Any Value	Permit	Originated				
8	Doctor = doctor_1	PrivateNotes = PatientA_PrivateNotes	HealthcareActions = Add	Environment = Any Value	Condition = Any Value	Permit	Originated				
9	Doctor = doctor_1	PrivateNotes = PatientB_PrivateNotes	Action = Any Value	Environment = Any Value	Condition = Any Value	Deny	Originated				
10	Doctor = doctor_1	PrivateNotes = PatientC_PrivateNotes	Action = Any Value	Environment = Any Value	Condition = Any Value	Deny	Originated				
11	Doctor = doctor_2	OldMedicalRecords = PatientA_OldMedicalRecords	HealthcareActions = View	Environment = Any Value	Condition = Any Value	Deny	Originated				
12	Doctor = doctor_2	OldMedicalRecords = PatientB_OldMedicalRecords	HealthcareActions = View	Environment = Any Value	Condition = Any Value	Permit	Originated				
13	Doctor = doctor_2	OldMedicalRecords = PatientC_OldMedicalRecords	HealthcareActions = View	Environment = Any Value	Condition = Any Value	Deny	Originated				
14	Doctor = doctor_2	PatientPersonalInfo = PatientA_PersonalInfo	HealthcareActions = View	Environment = Any Value	Condition = Any Value	Deny	Originated				
15	Doctor = doctor_2	PatientPersonalInfo = PatientB_PersonalInfo	HealthcareActions = View	Environment = Any Value	Condition = Any Value	Deny	Originated				
16	Doctor = doctor_2	PatientPersonalInfo = PatientC_PersonalInfo	HealthcareActions = View	Environment = Any Value	Condition = Any Value	Deny	Originated				
17	Doctor = doctor_2	PrivateNotes = PatientA_PrivateNotes	Action = Any Value	Environment = Any Value	Condition = Any Value	Deny	Originated				
18	Doctor = doctor_2	PrivateNotes = PatientB_PrivateNotes	HealthcareActions = View	Environment = Any Value	Condition = Any Value	Permit	Originated				
19	Doctor = doctor_2	PrivateNotes = PatientB_PrivateNotes	HealthcareActions = Add	Environment = Any Value	Condition = Any Value	Permit	Originated				
20	Doctor = doctor_2	PrivateNotes = PatientC_PrivateNotes	Action = Any Value	Environment = Any Value	Condition = Any Value	Deny	Originated				
21	Doctor = doctor_3	OldMedicalRecords = PatientA_OldMedicalRecords	HealthcareActions = View	Environment = Any Value	Condition = Any Value	Deny	Originated				
22	Doctor = doctor_3	OldMedicalRecords = PatientB_OldMedicalRecords	HealthcareActions = View	Environment = Any Value	Condition = Any Value	Deny	Originated				
23	Doctor = doctor_3	OldMedicalRecords = PatientC_OldMedicalRecords	HealthcareActions = View	Environment = Any Value	Condition = Any Value	Permit	Originated				
24	Doctor = doctor_3	PatientPersonalInfo = PatientA_PersonalInfo	HealthcareActions = View	Environment = Any Value	Condition = Any Value	Deny	Originated				
25	Doctor = doctor_3	PatientPersonalInfo = PatientB_PersonalInfo	HealthcareActions = View	Environment = Any Value	Condition = Any Value	Deny	Originated				
26	Doctor = doctor_3	PatientPersonalInfo = PatientC_PersonalInfo	HealthcareActions = View	Environment = Any Value	Condition = Any Value	Deny	Originated				
27	Doctor = doctor_3	PrivateNotes = PatientA_PrivateNotes	Action = Any Value	Environment = Any Value	Condition = Any Value	Deny	Originated				
28	Doctor = doctor_3	PrivateNotes = PatientB_PrivateNotes	Action = Any Value	Environment = Any Value	Condition = Any Value	Deny	Originated				
29	Doctor = doctor_3	PrivateNotes = PatientC_PrivateNotes	HealthcareActions = View	Environment = Any Value	Condition = Any Value	Permit	Originated				
30	Doctor = doctor_3	PrivateNotes = PatientC_PrivateNotes	HealthcareActions = Add	Environment = Any Value	Condition = Any Value	Permit	Originated				

Fig. 17. DoctorPolicy

9 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:

(Manager = manager) & (Action = View) & (OldMedicalRecords = PatientD_OldMedicalRecords) → 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: HealthcareTestCase2 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	Manager = manager	OldMedicalRecords = PatientD_OldMedicalRecords	HealthcareActions = View	Environment = Any Value	Condition = Any Value	Permit				

Fig. 18. Individual Security Requirement

10 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. 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 healthcare policies; the author created a resource "PatientD_OldMedicalRecords" 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 ManagerPolicy & DoctorPolicy as a Combined Policy Verification and analyze our verification result. Again, this will have already been done for you if you open Project File: HealthcareTest-Case2.

Status	Name	Verification Type	Verification Technique	Number of Policy(s)	Combination Algorithm	Enforcement Algorithm	Policy List
UpToDate	Policy Verification (June 18, 2018 15:04:10)	Standard	Combined Policy	2	Deny-overrides	Deny Biased	ABAC:ManagerPolicy, ABAC:DoctorPolicy

Requirement Schema	Subject	Resource	Action	Environment	Condition	Decision	Verification Result
Test Case 2	Manager = manager	OldMedicalRecords = PatientD_OldMedicalRecords	HealthcareActions = View	Environment = Any Value	Condition = Any Value	Permit	FALSE

Fig. 19. 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 Rules attached to Resource = PatientD_OldMedicalRecords. 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.

Result(s) with selected verification (Policy Verification (June 18, 2018 15:04:10)) 1 rows out of 1  

Requirement Schema	Subject	Resource	Action	Environment	Condition	Decision	Verification Result
Test Case 2	Manager = manager	OldMedicalRecords = PatientD_OldMedicalRecords	HealthcareActions = View	Environment = Any Value	Condition = Any Value	Permit	FALSE

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

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

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

Sequence No	Subject	Resource	Action	Environment	Condition	Decision	Inheritance Relation	Match Result
1	Manager = manager	OldMedicalRecords = PatientA_OldMedicalRecords	HealthcareActions = View	Environment = Any Value	Condition = Any Value	Permit	Originated	Not Applicable
2	Manager = manager	OldMedicalRecords = PatientB_OldMedicalRecords	HealthcareActions = View	Environment = Any Value	Condition = Any Value	Permit	Originated	Not Applicable
3	Manager = manager	OldMedicalRecords = PatientC_OldMedicalRecords	HealthcareActions = View	Environment = Any Value	Condition = Any Value	Permit	Originated	Not Applicable
4	Manager = manager	PatientPersonalInfo = PatientA_PersonalInfo	HealthcareActions = View	Environment = Any Value	Condition = Any Value	Permit	Originated	Not Applicable
5	Manager = manager	PatientPersonalInfo = PatientB_PersonalInfo	HealthcareActions = View	Environment = Any Value	Condition = Any Value	Permit	Originated	Not Applicable
6	Manager = manager	PatientPersonalInfo = PatientC_PersonalInfo	HealthcareActions = View	Environment = Any Value	Condition = Any Value	Permit	Originated	Not Applicable
7	Manager = manager	PrivateNotes = PatientA_PrivateNotes	HealthcareActions = View	Environment = Any Value	Condition = Any Value	Deny	Originated	Not Applicable
8	Manager = manager	PrivateNotes = PatientB_PrivateNotes	HealthcareActions = View	Environment = Any Value	Condition = Any Value	Deny	Originated	Not Applicable
9	Manager = manager	PrivateNotes = PatientC_PrivateNotes	HealthcareActions = View	Environment = Any Value	Condition = Any Value	Deny	Originated	Not Applicable

Fig. 20. Manager Policy: Match Results

Result(s) with selected verification (Policy Verification (June 18, 2018 15:04:10)) 1 rows out of 1  

Requirement Schema	Subject	Resource	Action	Environment	Condition	Decision	Verification Result
Test Case 2	Manager = manager	OldMedicalRecords = PatientD_OldMedicalRecords	HealthcareActions = View	Environment = Any Value	Condition = Any Value	Permit	FALSE

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

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

Rule(s) and Matching result of Selected Policy against the selected security requirement: 30 rows out of 30  

Sequence No	Subject	Resource	Action	Environment	Condition	Decision	Inheritance Relation	Match Result
1	Doctor = doctor_1	OldMedicalRecords = PatientA_OldMedicalRecords	HealthcareActions = View	Environment = Any Value	Condition = Any Value	Permit	Originated	Not Applicable
2	Doctor = doctor_1	OldMedicalRecords = PatientB_OldMedicalRecords	HealthcareActions = View	Environment = Any Value	Condition = Any Value	Deny	Originated	Not Applicable
3	Doctor = doctor_1	OldMedicalRecords = PatientC_OldMedicalRecords	HealthcareActions = View	Environment = Any Value	Condition = Any Value	Deny	Originated	Not Applicable
4	Doctor = doctor_1	PatientPersonalInfo = PatientA_PersonalInfo	HealthcareActions = View	Environment = Any Value	Condition = Any Value	Deny	Originated	Not Applicable
5	Doctor = doctor_1	PatientPersonalInfo = PatientB_PersonalInfo	HealthcareActions = View	Environment = Any Value	Condition = Any Value	Deny	Originated	Not Applicable
6	Doctor = doctor_1	PatientPersonalInfo = PatientC_PersonalInfo	HealthcareActions = View	Environment = Any Value	Condition = Any Value	Deny	Originated	Not Applicable
7	Doctor = doctor_1	PrivateNotes = PatientA_PrivateNotes	HealthcareActions = View	Environment = Any Value	Condition = Any Value	Permit	Originated	Not Applicable
8	Doctor = doctor_1	PrivateNotes = PatientA_PrivateNotes	HealthcareActions = Add	Environment = Any Value	Condition = Any Value	Permit	Originated	Not Applicable
9	Doctor = doctor_1	PrivateNotes = PatientB_PrivateNotes	Action = Any Value	Environment = Any Value	Condition = Any Value	Deny	Originated	Not Applicable
10	Doctor = doctor_1	PrivateNotes = PatientC_PrivateNotes	Action = Any Value	Environment = Any Value	Condition = Any Value	Deny	Originated	Not Applicable
11	Doctor = doctor_2	OldMedicalRecords = PatientA_OldMedicalRecords	HealthcareActions = View	Environment = Any Value	Condition = Any Value	Deny	Originated	Not Applicable
12	Doctor = doctor_2	OldMedicalRecords = PatientB_OldMedicalRecords	HealthcareActions = View	Environment = Any Value	Condition = Any Value	Permit	Originated	Not Applicable
13	Doctor = doctor_2	OldMedicalRecords = PatientC_OldMedicalRecords	HealthcareActions = View	Environment = Any Value	Condition = Any Value	Deny	Originated	Not Applicable
14	Doctor = doctor_2	PatientPersonalInfo = PatientA_PersonalInfo	HealthcareActions = View	Environment = Any Value	Condition = Any Value	Deny	Originated	Not Applicable
15	Doctor = doctor_2	PatientPersonalInfo = PatientB_PersonalInfo	HealthcareActions = View	Environment = Any Value	Condition = Any Value	Deny	Originated	Not Applicable
16	Doctor = doctor_2	PatientPersonalInfo = PatientC_PersonalInfo	HealthcareActions = View	Environment = Any Value	Condition = Any Value	Deny	Originated	Not Applicable
17	Doctor = doctor_2	PrivateNotes = PatientA_PrivateNotes	Action = Any Value	Environment = Any Value	Condition = Any Value	Deny	Originated	Not Applicable
18	Doctor = doctor_2	PrivateNotes = PatientB_PrivateNotes	HealthcareActions = View	Environment = Any Value	Condition = Any Value	Permit	Originated	Not Applicable
19	Doctor = doctor_2	PrivateNotes = PatientB_PrivateNotes	HealthcareActions = Add	Environment = Any Value	Condition = Any Value	Permit	Originated	Not Applicable
20	Doctor = doctor_2	PrivateNotes = PatientC_PrivateNotes	Action = Any Value	Environment = Any Value	Condition = Any Value	Deny	Originated	Not Applicable
21	Doctor = doctor_3	OldMedicalRecords = PatientA_OldMedicalRecords	HealthcareActions = View	Environment = Any Value	Condition = Any Value	Deny	Originated	Not Applicable
22	Doctor = doctor_3	OldMedicalRecords = PatientB_OldMedicalRecords	HealthcareActions = View	Environment = Any Value	Condition = Any Value	Deny	Originated	Not Applicable
23	Doctor = doctor_3	OldMedicalRecords = PatientC_OldMedicalRecords	HealthcareActions = View	Environment = Any Value	Condition = Any Value	Permit	Originated	Not Applicable
24	Doctor = doctor_3	PatientPersonalInfo = PatientA_PersonalInfo	HealthcareActions = View	Environment = Any Value	Condition = Any Value	Deny	Originated	Not Applicable
25	Doctor = doctor_3	PatientPersonalInfo = PatientB_PersonalInfo	HealthcareActions = View	Environment = Any Value	Condition = Any Value	Deny	Originated	Not Applicable
26	Doctor = doctor_3	PatientPersonalInfo = PatientC_PersonalInfo	HealthcareActions = View	Environment = Any Value	Condition = Any Value	Deny	Originated	Not Applicable
27	Doctor = doctor_3	PrivateNotes = PatientA_PrivateNotes	Action = Any Value	Environment = Any Value	Condition = Any Value	Deny	Originated	Not Applicable
28	Doctor = doctor_3	PrivateNotes = PatientB_PrivateNotes	Action = Any Value	Environment = Any Value	Condition = Any Value	Deny	Originated	Not Applicable
29	Doctor = doctor_3	PrivateNotes = PatientC_PrivateNotes	HealthcareActions = View	Environment = Any Value	Condition = Any Value	Permit	Originated	Not Applicable
30	Doctor = doctor_3	PrivateNotes = PatientC_PrivateNotes	HealthcareActions = Add	Environment = Any Value	Condition = Any Value	Permit	Originated	Not Applicable

Fig. 21. Doctor Policy: Match Results

11 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 (PatientD_OldMedicalRecords) 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 rule below to the ManagerPolicy...

ManagerPolicy: Add (1) New Rule:

(Rule No. = 10) → (Manager = manager) → (Action = View) → (Resource = PatientD_OldMedicalRecords) → decision = Permit

10	Manager = manager	OldMedicalRecords = PatientD_OldMedicalRecords	HealthcareActions = View	Environment = Any Value	Condition = Any Value	Permit	Originated
----	-------------------	--	--------------------------	-------------------------	-----------------------	--------	------------

Fig. 22. Manager Policy: New Rule (10)

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

Sequence No	Subject	Resource	Action	Environment	Condition	Decision	Inheritance Relation	Match Result
1	Manager = manager	OldMedicalRecords = PatientA_OldMedicalRecords	HealthcareActions = View	Environment = Any Value	Condition = Any Value	Permit	Originated	Not Applicable
2	Manager = manager	OldMedicalRecords = PatientB_OldMedicalRecords	HealthcareActions = View	Environment = Any Value	Condition = Any Value	Permit	Originated	Not Applicable
3	Manager = manager	OldMedicalRecords = PatientC_OldMedicalRecords	HealthcareActions = View	Environment = Any Value	Condition = Any Value	Permit	Originated	Not Applicable
4	Manager = manager	PatientPersonallInfo = PatientA_PersonallInfo	HealthcareActions = View	Environment = Any Value	Condition = Any Value	Permit	Originated	Not Applicable
5	Manager = manager	PatientPersonallInfo = PatientB_PersonallInfo	HealthcareActions = View	Environment = Any Value	Condition = Any Value	Permit	Originated	Not Applicable
6	Manager = manager	PatientPersonallInfo = PatientC_PersonallInfo	HealthcareActions = View	Environment = Any Value	Condition = Any Value	Permit	Originated	Not Applicable
7	Manager = manager	PrivateNotes = PatientA_PrivateNotes	HealthcareActions = View	Environment = Any Value	Condition = Any Value	Deny	Originated	Not Applicable
8	Manager = manager	PrivateNotes = PatientB_PrivateNotes	HealthcareActions = View	Environment = Any Value	Condition = Any Value	Deny	Originated	Not Applicable
9	Manager = manager	PrivateNotes = PatientC_PrivateNotes	HealthcareActions = View	Environment = Any Value	Condition = Any Value	Deny	Originated	Not Applicable
10	Manager = manager	OldMedicalRecords = PatientD_OldMedicalRecords	HealthcareActions = View	Environment = Any Value	Condition = Any Value	Permit	Originated	Permit

Fig. 23. Updated Policy: Resource Now Protected

12 SETTING UP THE POLICIES – TEST CASE 3 (UNDECIDED RULE)

This healthcare example contains two policies (ManagerPolicy & DoctorPolicy). The attributes in this example have been changed slightly from previous Test Case 1 and Test Case 2. OldMedicalRecords no longer has attribute value “PatientD_OldMedicalRecords,” and Private Notes has gained a new attribute value “PatientD_PrivateNotes.” The Attribute/Attribute Values included in these policies are as shown in Figure 24.

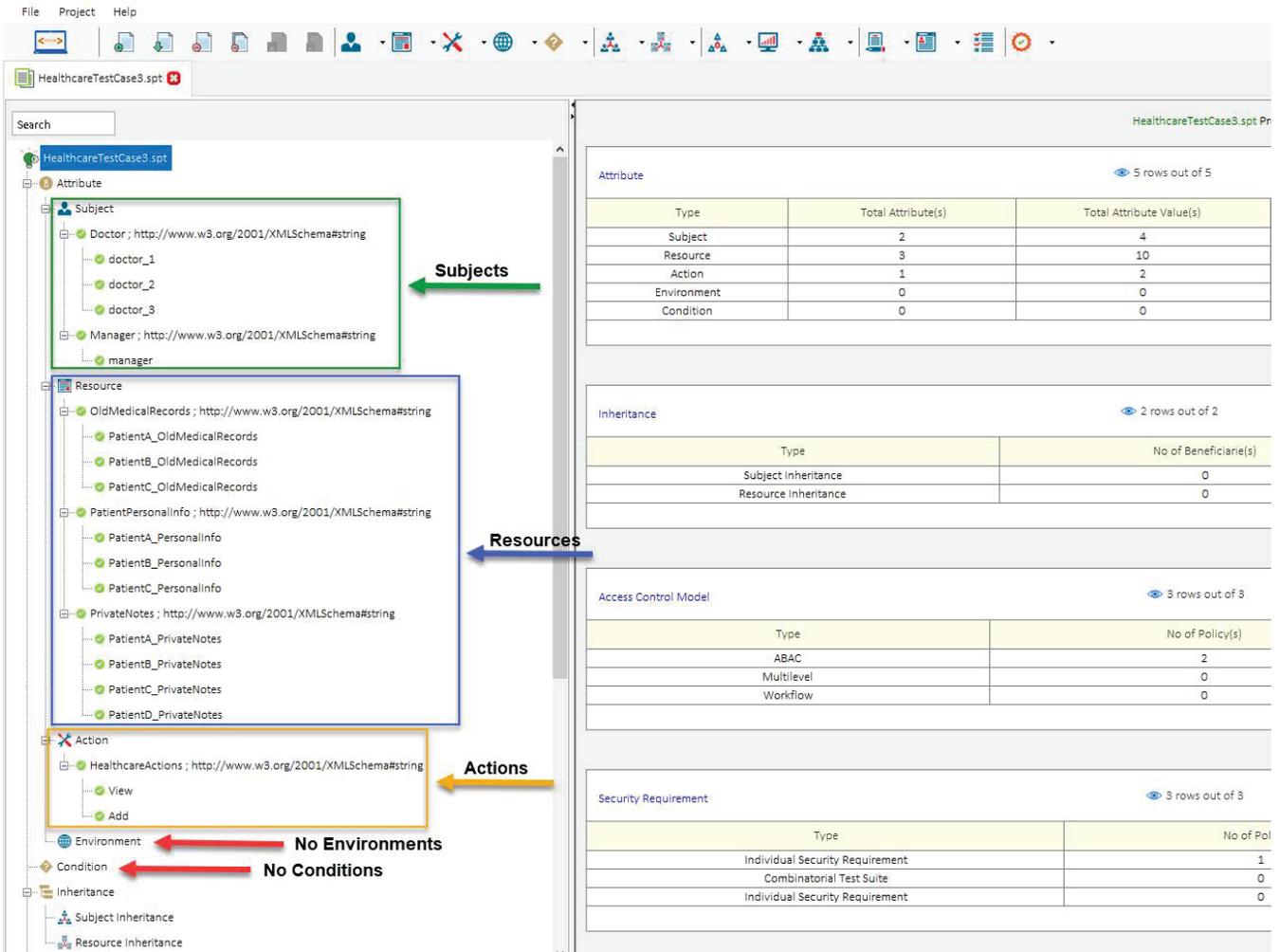


Fig. 24. Test Case 3

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

Now that we have entered our attributes we can model our two policies (ManagerPolicy & DoctorPolicy). 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:

ManagerPolicy:

- (Manager = manager, View, PatientA_OldMedicalRecords) → Permit
- (Manager = manager, View, PatientB_OldMedicalRecords) →Permit
- (Manager = manager, View, PatientC_OldMedicalRecords) →Permit
- (Manager = manager, View, PatientA_PersonalInfo) →Permit
- (Manager = manager, View, PatientB_PersonalInfo) →Permit
- (Manager = manager, View, PatientC_PersonalInfo) →Permit
- (Manager = manager, View, PatientA_PrivateNotes) →Deny
- (Manager = manager, View, PatientB_PrivateNotes) →Deny
- (Manager = manager, View, PatientC_PrivateNotes) →Deny
- (Manager = manager, View, PatientD_PrivateNotes) →Deny

DoctorPolicy:

- (Doctor = doctor_1, View, PatientA_OldMedicalRecords) →Permit
- (Doctor = doctor_1, View, PatientB_OldMedicalRecords) →Deny
- (Doctor = doctor_1, View, PatientC_OldMedicalRecords) →Deny
- (Doctor = doctor_1, View, PatientA_PersonalInfo) →Deny

(Doctor = doctor_1, View, PatientB_PersonalInfo) →Deny
 (Doctor = doctor_1, View, PatientC_PersonalInfo) →Deny
 (Doctor = doctor_1, View, PatientA_PrivateNotes) →Permit
 (Doctor = doctor_1, Add, PatientA_PrivateNotes) →Permit
 (Doctor = doctor_1, Action: Any, PatientB_PrivateNotes) →Deny
 (Doctor = doctor_1, Action: Any, PatientC_PrivateNotes) →Deny
 (Doctor = doctor_2, View, PatientA_OldMedicalRecords) →Deny
 (Doctor = doctor_2, View, PatientB_OldMedicalRecords) →Permit
 (Doctor = doctor_2, View, PatientC_OldMedicalRecords) →Deny
 (Doctor = doctor_2, View, PatientA_PersonalInfo) →Deny
 (Doctor = doctor_2, View, PatientB_PersonalInfo) →Deny
 (Doctor = doctor_2, View, PatientC_PersonalInfo) →Deny
 (Doctor = doctor_2, Action: Any, PatientA_PrivateNotes) →Deny
 (Doctor = doctor_2, View, PatientB_PrivateNotes) →Permit
 (Doctor = doctor_2, Add, PatientB_PrivateNotes) →Permit
 (Doctor = doctor_2, Action: Any, PatientC_PrivateNotes) →Deny
 (Doctor = doctor_3, View, PatientA_OldMedicalRecords) →Deny
 (Doctor = doctor_3, View, PatientB_OldMedicalRecords) →Deny
 (Doctor = doctor_3, View, PatientC_OldMedicalRecords) →Permit
 (Doctor = doctor_3, View, PatientA_PersonalInfo) →Deny
 (Doctor = doctor_3, View, PatientB_PersonalInfo) →Deny
 (Doctor = doctor_3, View, PatientC_PersonalInfo) →Deny
 (Doctor = doctor_3, Action: Any, PatientA_PrivateNotes) →Deny
 (Doctor = doctor_3, Action: Any, PatientB_PrivateNotes) →Deny
 (Doctor = doctor_3, View, PatientC_PrivateNotes) →Permit
 (Doctor = doctor_3, Add, PatientC_PrivateNotes) →Permit
 (Doctor = doctor_3, View, PatientD_PrivateNotes) →Permit
 (Doctor = doctor_3, Add, PatientD_PrivateNotes) →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: HealthcareTestCase3 and these policies will have been already created for you.

ManagerPolicy 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	ManagerPolicy	Deny-overrides	Deny Biased	10	June 13, 2018 15:47:46	June 13, 2018 15:47:46			

Rule (s) defined with selected policy (ManagerPolicy):								10 rows out of 10	Search	 
Sequence No	Subject	Resource	Action	Environment	Condition	Decision	Inheritance Relation			
1	Manager = manager	OldMedicalRecords = PatientA_OldMedicalRecords	HealthcareActions = View	Environment = Any Value	Condition = Any Value	Permit	Originated			
2	Manager = manager	OldMedicalRecords = PatientB_OldMedicalRecords	HealthcareActions = View	Environment = Any Value	Condition = Any Value	Permit	Originated			
3	Manager = manager	OldMedicalRecords = PatientC_OldMedicalRecords	HealthcareActions = View	Environment = Any Value	Condition = Any Value	Permit	Originated			
4	Manager = manager	PrivateNotes = PatientA_PrivateNotes	HealthcareActions = View	Environment = Any Value	Condition = Any Value	Deny	Originated			
5	Manager = manager	PrivateNotes = PatientB_PrivateNotes	HealthcareActions = View	Environment = Any Value	Condition = Any Value	Deny	Originated			
6	Manager = manager	PrivateNotes = PatientC_PrivateNotes	HealthcareActions = View	Environment = Any Value	Condition = Any Value	Deny	Originated			
7	Manager = manager	PatientPersonalInfo = PatientA_PersonalInfo	HealthcareActions = View	Environment = Any Value	Condition = Any Value	Permit	Originated			
8	Manager = manager	PatientPersonalInfo = PatientB_PersonalInfo	HealthcareActions = View	Environment = Any Value	Condition = Any Value	Permit	Originated			
9	Manager = manager	PatientPersonalInfo = PatientC_PersonalInfo	HealthcareActions = View	Environment = Any Value	Condition = Any Value	Permit	Originated			
10	Manager = manager	PrivateNotes = PatientD_PrivateNotes	HealthcareActions = View	Environment = Any Value	Condition = Any Value	Deny	Originated			

Fig. 25. ManagerPolicy

DoctorPolicy 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	DoctorPolicy	Deny-overrides	Deny Biased	32	June 13, 2018 16:00:58	June 13, 2018 16:00:58			

Rule (s) defined with selected policy (DoctorPolicy):								32 rows out of 32	Search	
Sequence No	Subject	Resource	Action	Environment	Condition	Decision	Inheritance Relation			
1	Doctor = doctor_1	OldMedicalRecords = PatientA_OldMedicalRecords	HealthcareActions = View	Environment = Any Value	Condition = Any Value	Permit	Originated			
2	Doctor = doctor_1	OldMedicalRecords = PatientB_OldMedicalRecords	HealthcareActions = View	Environment = Any Value	Condition = Any Value	Deny	Originated			
3	Doctor = doctor_1	OldMedicalRecords = PatientC_OldMedicalRecords	HealthcareActions = View	Environment = Any Value	Condition = Any Value	Deny	Originated			
4	Doctor = doctor_1	PatientPersonalInfo = PatientA_PersonalInfo	HealthcareActions = View	Environment = Any Value	Condition = Any Value	Deny	Originated			
5	Doctor = doctor_1	PatientPersonalInfo = PatientB_PersonalInfo	HealthcareActions = View	Environment = Any Value	Condition = Any Value	Deny	Originated			
6	Doctor = doctor_1	PatientPersonalInfo = PatientC_PersonalInfo	HealthcareActions = View	Environment = Any Value	Condition = Any Value	Deny	Originated			
7	Doctor = doctor_1	PrivateNotes = PatientA_PrivateNotes	HealthcareActions = View	Environment = Any Value	Condition = Any Value	Permit	Originated			
8	Doctor = doctor_1	PrivateNotes = PatientA_PrivateNotes	HealthcareActions = Add	Environment = Any Value	Condition = Any Value	Permit	Originated			
9	Doctor = doctor_1	PrivateNotes = PatientB_PrivateNotes	Action = Any Value	Environment = Any Value	Condition = Any Value	Deny	Originated			
10	Doctor = doctor_1	PrivateNotes = PatientC_PrivateNotes	Action = Any Value	Environment = Any Value	Condition = Any Value	Deny	Originated			
11	Doctor = doctor_2	OldMedicalRecords = PatientA_OldMedicalRecords	HealthcareActions = View	Environment = Any Value	Condition = Any Value	Deny	Originated			
12	Doctor = doctor_2	OldMedicalRecords = PatientB_OldMedicalRecords	HealthcareActions = View	Environment = Any Value	Condition = Any Value	Permit	Originated			
13	Doctor = doctor_2	OldMedicalRecords = PatientC_OldMedicalRecords	HealthcareActions = View	Environment = Any Value	Condition = Any Value	Deny	Originated			
14	Doctor = doctor_2	PatientPersonalInfo = PatientA_PersonalInfo	HealthcareActions = View	Environment = Any Value	Condition = Any Value	Deny	Originated			
15	Doctor = doctor_2	PatientPersonalInfo = PatientB_PersonalInfo	HealthcareActions = View	Environment = Any Value	Condition = Any Value	Deny	Originated			
16	Doctor = doctor_2	PatientPersonalInfo = PatientC_PersonalInfo	HealthcareActions = View	Environment = Any Value	Condition = Any Value	Deny	Originated			
17	Doctor = doctor_2	PrivateNotes = PatientA_PrivateNotes	Action = Any Value	Environment = Any Value	Condition = Any Value	Deny	Originated			
18	Doctor = doctor_2	PrivateNotes = PatientB_PrivateNotes	HealthcareActions = View	Environment = Any Value	Condition = Any Value	Permit	Originated			
19	Doctor = doctor_2	PrivateNotes = PatientB_PrivateNotes	HealthcareActions = Add	Environment = Any Value	Condition = Any Value	Permit	Originated			
20	Doctor = doctor_2	PrivateNotes = PatientC_PrivateNotes	Action = Any Value	Environment = Any Value	Condition = Any Value	Deny	Originated			
21	Doctor = doctor_3	OldMedicalRecords = PatientA_OldMedicalRecords	HealthcareActions = View	Environment = Any Value	Condition = Any Value	Deny	Originated			
22	Doctor = doctor_3	OldMedicalRecords = PatientB_OldMedicalRecords	HealthcareActions = View	Environment = Any Value	Condition = Any Value	Deny	Originated			
23	Doctor = doctor_3	OldMedicalRecords = PatientC_OldMedicalRecords	HealthcareActions = View	Environment = Any Value	Condition = Any Value	Permit	Originated			
24	Doctor = doctor_3	PatientPersonalInfo = PatientA_PersonalInfo	HealthcareActions = View	Environment = Any Value	Condition = Any Value	Deny	Originated			
25	Doctor = doctor_3	PatientPersonalInfo = PatientB_PersonalInfo	HealthcareActions = View	Environment = Any Value	Condition = Any Value	Deny	Originated			
26	Doctor = doctor_3	PatientPersonalInfo = PatientC_PersonalInfo	HealthcareActions = View	Environment = Any Value	Condition = Any Value	Deny	Originated			
27	Doctor = doctor_3	PrivateNotes = PatientA_PrivateNotes	Action = Any Value	Environment = Any Value	Condition = Any Value	Deny	Originated			
28	Doctor = doctor_3	PrivateNotes = PatientB_PrivateNotes	Action = Any Value	Environment = Any Value	Condition = Any Value	Deny	Originated			
29	Doctor = doctor_3	PrivateNotes = PatientC_PrivateNotes	HealthcareActions = View	Environment = Any Value	Condition = Any Value	Permit	Originated			
30	Doctor = doctor_3	PrivateNotes = PatientC_PrivateNotes	HealthcareActions = Add	Environment = Any Value	Condition = Any Value	Permit	Originated			
31	Doctor = doctor_3	PrivateNotes = PatientD_PrivateNotes	HealthcareActions = View	Environment = Any Value	Condition = Any Value	Permit	Originated			
32	Doctor = doctor_3	PrivateNotes = PatientD_PrivateNotes	HealthcareActions = Add	Environment = Any Value	Condition = Any Value	Permit	Originated			

Fig. 26. DoctorPolicy

14 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:

(Doctor = doctor_1) & (Action = View) & (PrivateNotes = PatientD_PrivateNotes) →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: HealthcareTestCase3 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	Doctor = doctor_1	PrivateNotes = PatientD_PrivateNotes	HealthcareActions = View	Environment = Any Value	Condition = Any Value	Permit			

Fig. 27. Individual Security Requirement

15 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 missing a step.

For example, when the policy author was designing the logic for these healthcare policies; the author created rules for Doctor = doctor3 and Manager = manager to access “PatientD_PrivateNotes” but did not define access rules for Doctor = doctor1 and Doctor = doctor2. In this situation, if doctor1 or doctor2 were to attempt to take action on “PatientD_PrivateNotes,” the system would be forced to make a default decision instead of a defined 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 ManagerPolicy & DoctorPolicy as a Combined Policy Verification and analyze our verification result. Again, this will have already been done for you if you open Project File: HealthcareTestCase3.

Status	Name	Verification Type	Verification Technique	Number of Policy(s)	Combination Algorithm	Enforcement Algorithm	Policy List
UpToDate	Policy Verification (June 13, 2018 16:17:33)	Standard	Combined Policy	2	Deny-overrides	Deny Biased	ABAC:ManagerPolicy, ABAC:DoctorPolicy

Requirement Schema	Subject	Resource	Action	Environment	Condition	Decision	Verification Result
Test Case 3	Doctor = doctor_1	PrivateNotes = PatientD_PrivateNotes	HealthcareActions = View	Environment = Any Value	Condition = Any Value	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 = doctor_1 taking action on Resource = PatientD_PrivateNotes. We can see this by noticing that every “Match Result” is “Not Applicable” whereas if there were Rules existing for doctor_1 and Resource = PatientD_PrivateNotes we would have at least see one Rule with a (Permit or Deny) Match Result.

→ PatientD_PrivateNotes which is known as an Undecided Rule error.

16 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 (e.g., include doctor_1 and doctor_2) in any policies that handle access requests to PatientD_PrivateNotes. For example, adding the rules below to the DoctorPolicy for our specific example...

DoctorPolicy: Add (4) New Rules:

- (Rule No. = 33) → (Doctor = doctor_1) → (Action = View) → (Resource = PatientD_PrivateNotes) → decision = Permit
- (Rule No. = 34) → (Doctor = doctor_1) → (Action = Add) → (Resource = PatientD_PrivateNotes) → decision = Permit
- (Rule No. = 35) → (Doctor = doctor_2) → (Action = View) → (Resource = PatientD_PrivateNotes) → decision = Permit
- (Rule No. = 36) → (Doctor = doctor_2) → (Action = Add) → (Resource = PatientD_PrivateNotes) → decision = Permit

33	Doctor = doctor_1	PrivateNotes = PatientD_PrivateNotes	HealthcareActions = View	Environment = Any Value	Condition = Any Value	Permit	Originated
34	Doctor = doctor_1	PrivateNotes = PatientD_PrivateNotes	HealthcareActions = Add	Environment = Any Value	Condition = Any Value	Permit	Originated
35	Doctor = doctor_2	PrivateNotes = PatientD_PrivateNotes	HealthcareActions = View	Environment = Any Value	Condition = Any Value	Permit	Originated
36	Doctor = doctor_2	PrivateNotes = PatientD_PrivateNotes	HealthcareActions = Add	Environment = Any Value	Condition = Any Value	Permit	Originated

Fig. 31. Doctor Policy: New Rules (33,34,35,36)

Now, looking out 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, ManagerPolicy has no rules related to the security requirement (doctor_one → View → PatientD_PrivateNotes) 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 ManagerPolicy = Deny. However, in the case of the DoctorPolicy we have the Combined Result = Permit due to the new rules we added (e.g., see new Rule 33 below). Hence, we have opposing Combined Results (ManagerPolicy = Deny; DoctorPolicy = 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 DoctorPolicy in favor of the Deny result from the ManagerPolicy to make the final Verification Result = False.

Status	Name	Verification Type	Verification Technique	Number of Policy(s)	Combination Algorithm	Enforcement Algorithm	Policy List
UpToDate	Policy Verification (June 20, 2018 11:54:15)	Standard	Combined Policy	2	Deny-overrides	Deny Biased	ABAC:ManagerPolicy, ABAC:DoctorPolicy

Requirement Schema	Subject	Resource	Action	Environment	Condition	Decision	Verification Result
Test Case 3	Doctor = doctor_1	PrivateNotes = PatientD_PrivateNotes	HealthcareActions = View	Environment = Any Value	Condition = Any Value	Permit	FALSE

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

Sequence No	Subject	Resource	Action	Environment	Condition	Decision	Inheritance Relation	Match Result
20	Doctor = doctor_2	PrivateNotes = PatientC_PrivateNotes	Action = Any Value	Environment = Any Value	Condition = Any Value	Deny	Originated	Not Applicable
21	Doctor = doctor_3	OldMedicalRecords = PatientA_OldMedicalRecords	HealthcareActions = View	Environment = Any Value	Condition = Any Value	Deny	Originated	Not Applicable
22	Doctor = doctor_3	OldMedicalRecords = PatientB_OldMedicalRecords	HealthcareActions = View	Environment = Any Value	Condition = Any Value	Deny	Originated	Not Applicable
23	Doctor = doctor_3	OldMedicalRecords = PatientC_OldMedicalRecords	HealthcareActions = View	Environment = Any Value	Condition = Any Value	Permit	Originated	Not Applicable
24	Doctor = doctor_3	PatientPersonalInfo = PatientA_PersonalInfo	HealthcareActions = View	Environment = Any Value	Condition = Any Value	Deny	Originated	Not Applicable
25	Doctor = doctor_3	PatientPersonalInfo = PatientB_PersonalInfo	HealthcareActions = View	Environment = Any Value	Condition = Any Value	Deny	Originated	Not Applicable
26	Doctor = doctor_3	PatientPersonalInfo = PatientC_PersonalInfo	HealthcareActions = View	Environment = Any Value	Condition = Any Value	Deny	Originated	Not Applicable
27	Doctor = doctor_3	PrivateNotes = PatientA_PrivateNotes	Action = Any Value	Environment = Any Value	Condition = Any Value	Deny	Originated	Not Applicable
28	Doctor = doctor_3	PrivateNotes = PatientB_PrivateNotes	Action = Any Value	Environment = Any Value	Condition = Any Value	Deny	Originated	Not Applicable
29	Doctor = doctor_3	PrivateNotes = PatientC_PrivateNotes	HealthcareActions = View	Environment = Any Value	Condition = Any Value	Permit	Originated	Not Applicable
30	Doctor = doctor_3	PrivateNotes = PatientC_PrivateNotes	HealthcareActions = Add	Environment = Any Value	Condition = Any Value	Permit	Originated	Not Applicable
31	Doctor = doctor_3	PrivateNotes = PatientD_PrivateNotes	HealthcareActions = View	Environment = Any Value	Condition = Any Value	Permit	Originated	Not Applicable
32	Doctor = doctor_3	PrivateNotes = PatientD_PrivateNotes	HealthcareActions = Add	Environment = Any Value	Condition = Any Value	Permit	Originated	Not Applicable
33	Doctor = doctor_1	PrivateNotes = PatientD_PrivateNotes	HealthcareActions = View	Environment = Any Value	Condition = Any Value	Permit	Originated	Permit
34	Doctor = doctor_1	PrivateNotes = PatientD_PrivateNotes	HealthcareActions = Add	Environment = Any Value	Condition = Any Value	Permit	Originated	Not Applicable
35	Doctor = doctor_2	PrivateNotes = PatientD_PrivateNotes	HealthcareActions = View	Environment = Any Value	Condition = Any Value	Permit	Originated	Not Applicable
36	Doctor = doctor_2	PrivateNotes = PatientD_PrivateNotes	HealthcareActions = Add	Environment = Any Value	Condition = Any Value	Permit	Originated	Not Applicable

Fig. 32. Updated Results: No Undecided Rule

17 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 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.