Bank 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 Bank Policy test cases have been created by InfoBeyond Technology LLC to demonstrate commonly found errors in access control policy design such as Leak Privilege, Block Privilege, Inheritance Loop, Separation of Duty, and Inconsistent Assignment. These test cases consist of policies/rules 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 (LEAK PRIVILEGE)

This bank example contains three policies (BankTeller Policy & LoanOfficer Policy & FManager Policy). 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.

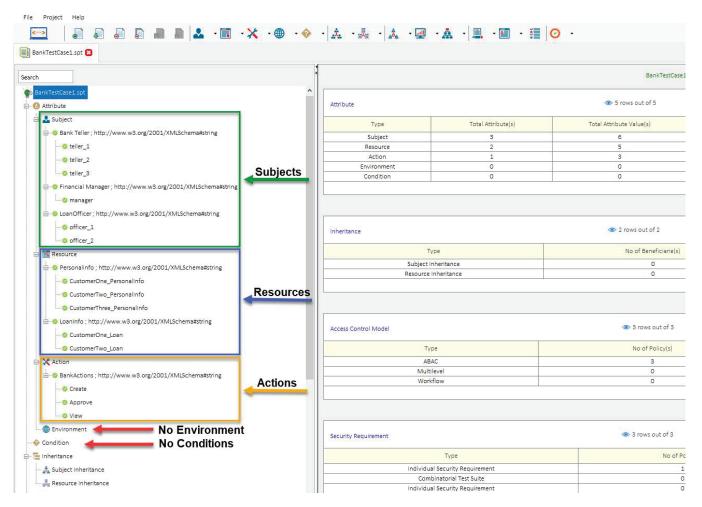


Fig. 1. Test Case 1

3 MODELING YOUR POLICY – TEST CASE 1 (LEAK PRIVILEGE)

Now that we have entered our attributes we can model our three policies (BankTeller Policy & LoanOfficer Policy & FManager 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:

BankTeller Policy:

Dunik rener roney.
(Bank Teller = teller_1, CustomerOne_PersonalInfo, Create) \rightarrow Permit
(Bank Teller = teller_1, CustomerOne_PersonalInfo, View) \rightarrow Permit
(Bank Teller = teller_1, CustomerTwo_PersonalInfo, Create) \rightarrow Permit
(Bank Teller = teller_1, CustomerTwo_PersonalInfo, View) →Permit
(Bank Teller = teller_1, CustomerThree_PersonalInfo, Create) \rightarrow Permit
(Bank Teller = teller_1, CustomerThree_PersonalInfo, View) \rightarrow Permit
(Bank Teller = teller_2, CustomerOne_PersonalInfo, Create) \rightarrow Permit
(Bank Teller = teller_2, CustomerOne_PersonalInfo, View) \rightarrow Permit
(Bank Teller = teller_2, CustomerTwo_PersonalInfo, Create) →Permit
(Bank Teller = teller_2, CustomerTwo_PersonalInfo, View) →Permit
(Bank Teller = teller_2, CustomerThree_PersonalInfo, Create) \rightarrow Permit
(Bank Teller = teller_2, CustomerThree_PersonalInfo, View) \rightarrow Permit
(Bank Teller = teller_3, CustomerOne_PersonalInfo, Create) \rightarrow Permit
(Bank Teller = teller_3, CustomerOne_PersonalInfo, View) \rightarrow Permit
(Bank Teller = teller_3, CustomerTwo_PersonalInfo, Create) →Permit
(Bank Teller = teller_3, CustomerTwo_PersonalInfo, View) →Permit
(Bank Teller = teller_3, CustomerThree_PersonalInfo, Create) \rightarrow Permit
(Bank Teller = teller_3, CustomerThree_PersonalInfo, View) \rightarrow Permit

(Bank Teller = teller_1, CustomerOne_Loan, View) \rightarrow Permit (Bank Teller = teller_1, CustomerTwo_Loan, View) \rightarrow Permit (Bank Teller = teller_2, CustomerOne_Loan, View) \rightarrow Permit (Bank Teller = teller_3, CustomerTwo_Loan, View) \rightarrow Permit (Bank Teller = teller_3, CustomerOne_Loan, View) \rightarrow Permit (Bank Teller = teller_1, CustomerTwo_Loan, View) \rightarrow Permit (Bank Teller = teller_1, CustomerOne_Loan, Approve) \rightarrow Deny (Bank Teller = teller_1, CustomerTwo_Loan, Approve) \rightarrow Deny (Bank Teller = teller_2, CustomerOne_Loan, Approve) \rightarrow Deny (Bank Teller = teller_2, CustomerOne_Loan, Approve) \rightarrow Deny (Bank Teller = teller_2, CustomerTwo_Loan, Approve) \rightarrow Deny (Bank Teller = teller_3, CustomerOne_Loan, Approve) \rightarrow Deny (Bank Teller = teller_3, CustomerOne_Loan, Approve) \rightarrow Deny

LoanOfficer Policy:

(Loan Officer = officer_1, CustomerOne_PersonalInfo, View) \rightarrow Permit
(Loan Officer = officer_1, CustomerOne_PersonalInfo, Create) \rightarrow Permit
(Loan Officer = officer_1, CustomerTwo_PersonalInfo, View) →Permit
(Loan Officer = officer_1, CustomerTwo_PersonalInfo, Create) \rightarrow Permit
(Loan Officer = officer_1, CustomerThree_PersonalInfo, View) →Permit
(Loan Officer = officer_1, CustomerThree_PersonalInfo, Create) \rightarrow Permit
(Loan Officer = officer_1, CustomerOne_Loan, View) →Permit
(Loan Officer = officer_1, CustomerOne_Loan, Create) \rightarrow Permit
(Loan Officer = officer_1, CustomerTwo_Loan, View) \rightarrow Permit
(Loan Officer = officer_1, CustomerTwo_Loan, Create) \rightarrow Permit
(Loan Officer = officer_2, CustomerOne_PersonalInfo, View) \rightarrow Permit
(Loan Officer = officer_2, CustomerOne_PersonalInfo, Create) \rightarrow Permit
(Loan Officer = officer_2, CustomerTwo_PersonalInfo, View) \rightarrow Permit
(Loan Officer = officer_2, CustomerTwo_PersonalInfo, Create) \rightarrow Permit
(Loan Officer = officer_2, CustomerThree_PersonalInfo, View) \rightarrow Permit
(Loan Officer = officer_2, CustomerThree_PersonalInfo, Create) \rightarrow Permit
(Loan Officer = officer_2, CustomerOne_Loan, View) \rightarrow Permit
(Loan Officer = officer_2, CustomerOne_Loan, Create) \rightarrow Permit
(Loan Officer = officer_2, CustomerTwo_Loan, View) \rightarrow Permit
(Loan Officer = officer_2, CustomerTwo_Loan, Create) \rightarrow Permit
(Loan Officer = officer_1, CustomerOne_Loan, Approve) \rightarrow Deny
(Loan Officer = officer_1, CustomerTwo_Loan, Approve) \rightarrow Deny
(Loan Officer = officer_2, CustomerOne_Loan, Approve) \rightarrow Deny
(Loan Officer = officer_2, CustomerTwo_Loan, Approve) \rightarrow Deny

FManager Policy:

(Manager = manager, CustomerOne_PersonalInfo, Create) →Permit (Manager = manager, CustomerOne_PersonalInfo, View) →Permit (Manager = manager, CustomerTwo_PersonalInfo, Create) →Permit (Manager = manager, CustomerTwo_PersonalInfo, View) →Permit (Manager = manager, CustomerThree_PersonalInfo, View) →Permit (Manager = manager, CustomerThree_PersonalInfo, View) →Permit (Manager = manager, CustomerOne_Loan, Create) →Permit (Manager = manager, CustomerOne_Loan, Create) →Permit (Manager = manager, CustomerTwo_Loan, Approve) →Permit (Manager = manager, CustomerTwo_Loan, Approve) →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: BankTestCase1 and these policies will have been already created for you.

WWW.SECURITYPOLICYTOOL.COM

nkTeller Policy P	olicy(s) Summary		I rows out of 1				Search
Model	Policy Name	Rule Combination Algorithm	Policy Enforcement Algorithm	No. of Rule(s)	Time Created		Last Modified
ABAC	BankTeller Policy	Deny-overrides	Deny Biased	30	July 3, 2018 14:28:3	32	July 3, 2018 14:28:32
le (s) defined wit	h selected policy (BankTeller I	Policy):	🐵 30 rov	vs out of 30			Search
Sequence No	Subject	Resource	Action	Environment	Condition	Decision	Inheritance Relat
1	Bank Teller = teller_1	Personalinfo = CustomerOne_Personalinfo	BankActions = Create	Environment = Any Value	Condition = Any Value	Permit	Originated
2	Bank Teller = teller_1	Personalinfo = CustomerOne_Personalinfo	BankActions = View	Environment = Any Value	Condition = Any Value	Permit	Originated
3	Bank Teller = teller_1	Personalinfo = CustomerTwo_Personalinfo	BankActions = Create	Environment = Any Value	Condition = Any Value	Permit	Originated
4	Bank Teller = teller_1	Personalinfo = CustomerTwo_Personalinfo	BankActions = View	Environment = Any Value	Condition = Any Value	Permit	Originated
5	Bank Teller = teller_1	Personalinfo = CustomerThree_Personalinfo	BankActions = Create	Environment = Any Value	Condition = Any Value	Permit	Originated
6	Bank Teller = teller_1	Personalinfo = CustomerThree_Personalinfo	BankActions = View	Environment = Any Value	Condition = Any Value	Permit	Originated
7	Bank Teller = teller_2	Personalinfo = CustomerOne_Personalinfo	BankActions = Create	Environment = Any Value	Condition = Any Value	Permit	Originated
8	Bank Teller = teller_2	Personalinfo = CustomerOne_Personalinfo	BankActions = View	Environment = Any Value	Condition = Any Value	Permit	Originated
9	Bank Teller = teller_2	Personalinfo = CustomerTwo_Personalinfo	BankActions = Create	Environment = Any Value	Condition = Any Value	Permit	Originated
10	Bank Teller = teller_2	Personalinfo = CustomerTwo_Personalinfo	BankActions = View	Environment = Any Value	Condition = Any Value	Permit	Originated
11	Bank Teller = teller_2	Personalinfo = CustomerThree_Personalinfo	BankActions = Create	Environment = Any Value	Condition = Any Value	Permit	Originated
12	Bank Teller = teller_2	Personalinfo = CustomerThree_Personalinfo	BankActions = View	Environment = Any Value	Condition = Any Value	Permit	Originated
13	Bank Teller = teller_3	Personalinfo = CustomerOne_Personalinfo	BankActions = Create	Environment = Any Value	Condition = Any Value	Permit	Originated
14	Bank Teller = teller_3	Personalinfo = CustomerOne_Personalinfo	BankActions = View	Environment = Any Value	Condition = Any Value	Permit	Originated
15	Bank Teller = teller_3	Personalinfo = CustomerTwo_Personalinfo	BankActions = Create	Environment = Any Value	Condition = Any Value	Permit	Originated
16	Bank Teller = teller_3	Personalinfo = CustomerTwo_Personalinfo	BankActions = View	Environment = Any Value	Condition = Any Value	Permit	Originated
17	Bank Teller = teller_3	Personalinfo = CustomerThree_Personalinfo	BankActions = Create	Environment = Any Value	Condition = Any Value	Permit	Originated
18	Bank Teller = teller_3	Personalinfo = CustomerThree_Personalinfo	BankActions = View	Environment = Any Value	Condition = Any Value	Permit	Originated
19	Bank Teller = teller_1	LoanInfo = CustomerOne_Loan	BankActions = View	Environment = Any Value	Condition = Any Value	Permit	Originated
20	Bank Teller = teller_1	LoanInfo = CustomerTwo_Loan	BankActions = View	Environment = Any Value	Condition = Any Value	Permit	Originated
21	Bank Teller = teller_2	Loaninfo = CustomerOne_Loan	BankActions = View	Environment = Any Value	Condition = Any Value	Permit	Originated
22	Bank Teller = teller_2	LoanInfo = CustomerTwo_Loan	BankActions = View	Environment = Any Value	Condition = Any Value	Permit	Originated
23	Bank Teller = teller_3	LoanInfo = CustomerOne_Loan	BankActions = View	Environment = Any Value	Condition = Any Value	Permit	Originated
24	Bank Teller = teller_3	Loaninfo = CustomerTwo_Loan	BankActions = View	Environment = Any Value	Condition = Any Value	Permit	Originated
25	Bank Teller = teller_1	Loaninfo = CustomerOne_Loan	BankActions = Approve	Environment = Any Value	Condition = Any Value	Deny	Originated
26	Bank Teller = teller_1	Loaninfo = CustomerTwo_Loan	BankActions = Approve	Environment = Any Value	Condition = Any Value	Deny	Originated
27	Bank Teller = teller_2	Loaninfo = CustomerOne_Loan	BankActions = Approve	Environment = Any Value	Condition = Any Value	Deny	Originated
28	Bank Teller = teller_2	Loaninfo = CustomerTwo_Loan	BankActions = Approve	Environment = Any Value	Condition = Any Value	Permit	Originated
29	Bank Teller = teller_3	Loaninfo = CustomerOne_Loan	BankActions = Approve	Environment = Any Value	Condition = Any Value	Deny	Originated
30	Bank Teller = teller 3	Loaninfo = CustomerTwo Loan	BankActions = Approve	Environment = Any Value	Condition = Any Value	Denv	Originated

Fig. 2. BankTeller Policy

oanOfficer 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	LoanOfficer Policy	Deny-overrides	Deny Biased	24	July 3, 2018 14:40:4	16	July 3, 2018 14:40):46
Rule (s) defined wit	th selected policy (LoanOfficer F	Policy):	@ 24 row	rs out of 24			Search	×
Sequence No	Subject	Resource	Action	Environment	Condition	Decision	Inheritance R	elatior
1	LoanOfficer = officer_1	Personalinfo = CustomerOne_Personalinfo	BankActions = View	Environment = Anv Value	Condition = Any Value	Permit	Originate	ed
2	LoanOfficer = officer_1	Personalinfo = CustomerOne_Personalinfo	BankActions = Create	Environment = Any Value	Condition = Any Value	Permit	Originate	ed
3	LoanOfficer = officer_1	Personalinfo = CustomerTwo_Personalinfo	BankActions = View	Environment = Any Value	Condition = Any Value	Permit	Originate	ed
4	LoanOfficer = officer_1	Personalinfo = CustomerTwo_Personalinfo	BankActions = Create	Environment = Any Value	Condition = Any Value	Permit	Originate	ed
5	LoanOfficer = officer_1	Personalinfo = CustomerThree_Personalinfo	BankActions = View	Environment = Any Value	Condition = Any Value	Permit	Originate	ed
6	LoanOfficer = officer_1	Personalinfo = CustomerThree_Personalinfo	BankActions = Create	Environment = Any Value	Condition = Any Value	Permit	Originate	ed
7	LoanOfficer = officer_1	Loaninfo = CustomerOne_Loan	BankActions = View	Environment = Any Value	Condition = Any Value	Permit	Originate	ed
8	LoanOfficer = officer_1	Loaninfo = CustomerOne_Loan	BankActions = Create	Environment = Any Value	Condition = Any Value	Permit	Originate	ed
9	LoanOfficer = officer_1	LoanInfo = CustomerTwo_Loan	BankActions = View	Environment = Any Value	Condition = Any Value	Permit	Originate	ed
10	LoanOfficer = officer_1	Loaninfo = CustomerTwo_Loan	BankActions = Create	Environment = Any Value	Condition = Any Value	Permit	Originate	ed
11	LoanOfficer = officer_2	Personalinfo = CustomerOne_Personalinfo	BankActions = View	Environment = Any Value	Condition = Any Value	Permit	Originate	ed
12	LoanOfficer = officer_2	Personalinfo = CustomerOne_Personalinfo	BankActions = Create	Environment = Any Value	Condition = Any Value	Permit	Originate	ed
13	LoanOfficer = officer_2	Personalinfo = CustomerTwo_Personalinfo	BankActions = View	Environment = Any Value	Condition = Any Value	Permit	Originate	ed
14	LoanOfficer = officer_2	Personalinfo = CustomerTwo_Personalinfo	BankActions = Create	Environment = Any Value	Condition = Any Value	Permit	Originate	ed
15	LoanOfficer = officer_2	Personalinfo = CustomerThree_Personalinfo	BankActions = View	Environment = Any Value	Condition = Any Value	Permit	Originate	ed
16	LoanOfficer = officer_2	Personalinfo = CustomerThree_Personalinfo	BankActions = Create	Environment = Any Value	Condition = Any Value	Permit	Originate	ed
17	LoanOfficer = officer_2	Loaninfo = CustomerOne_Loan	BankActions = View	Environment = Any Value	Condition = Any Value	Permit	Originate	ed
18	LoanOfficer = officer_2	LoanInfo = CustomerOne_Loan	BankActions = Create	Environment = Any Value	Condition = Any Value	Permit	Originate	ed
19	LoanOfficer = officer_2	LoanInfo = CustomerTwo_Loan	BankActions = View	Environment = Any Value	Condition = Any Value	Permit	Originate	ad
20	LoanOfficer = officer_2	Loaninfo = CustomerTwo_Loan	BankActions = Create	Environment = Any Value	Condition = Any Value	Permit	Originate	ad
21	LoanOfficer = officer_1	Loaninfo = CustomerOne_Loan	BankActions = Approve	Environment = Any Value	Condition = Any Value	Deny	Originate	ed
22	LoanOfficer = officer_1	Loaninfo = CustomerTwo_Loan	BankActions = Approve	Environment = Any Value	Condition = Any Value	Deny	Originate	ed
23	LoanOfficer = officer_2	Loaninfo = CustomerOne_Loan	BankActions = Approve	Environment = Any Value	Condition = Any Value	Deny	Originate	ad
24	LoanOfficer = officer_2	Loaninfo = CustomerTwo_Loan	BankActions = Approve	Environment = Any Value	Condition = Any Value	Deny	Originate	ed

Fig. 3. LoanOfficer Policy

FManager Policy P	olicy(s) Summary		1 rows out of 1			Se	arch 🚺
Model	Policy Name	Rule Combination Algorithm	Policy Enforcement Algorithm	No. of Rule(s)	Time Created		Last Modified
ABAC	FManager Policy	Permit-overrides	Permit Biased	12	July 3, 2018 15:03:03		uly 3, 2018 15:03:03
Rule (s) defined wi	th selected policy (FManager Policy):		@ 12 rows ou	it of 12		Se	arch
Sequence No	Subject	Resource	Action	Environment	Condition	Decision	Inheritance Relation
1	Financial Manager = manager	Personalinfo = CustomerOne_Personalinfo	BankActions = Create	Environment = Anv Value	Condition = Any Value	Permit	Originated
2	Financial Manager = manager	Personalinfo = CustomerOne_Personalinfo	BankActions = View	Environment = Any Value	Condition = Any Value	Permit	Originated
3	Financial Manager = manager	Personalinfo = CustomerTwo_Personalinfo	BankActions = Create	Environment = Any Value	Condition = Any Value	Permit	Originated
4	Financial Manager = manager	Personalinfo = CustomerTwo_Personalinfo	BankActions = View	Environment = Any Value	Condition = Any Value	Permit	Originated
5	Financial Manager = manager	Personalinfo = CustomerThree_Personalinfo	BankActions = Create	Environment = Any Value	Condition = Any Value	Permit	Originated
6	Financial Manager = manager	Personalinfo = CustomerThree_Personalinfo	BankActions = View	Environment = Any Value	Condition = Any Value	Permit	Originated
7	Financial Manager = manager	Loaninfo = CustomerOne_Loan	BankActions = Create	Environment = Any Value	Condition = Any Value	Permit	Originated
8	Financial Manager = manager	LoanInfo = CustomerOne_Loan	BankActions = View	Environment = Any Value	Condition = Any Value	Permit	Originated
9	Financial Manager = manager	Loaninfo = CustomerTwo_Loan	BankActions = Create	Environment = Any Value	Condition = Any Value	Permit	Originated
10	Financial Manager = manager	LoanInfo = CustomerTwo_Loan	BankActions = View	Environment = Any Value	Condition = Any Value	Permit	Originated
11	Financial Manager = manager	LoanInfo = CustomerOne_Loan	BankActions = Approve	Environment = Any Value	Condition = Any Value	Permit	Originated
		Loaninfo = CustomerTwo Loan	BankActions = Approve			Permit	Originated

Fig. 4. FManager Policy

4 INDIVIDUAL SECURITY REQUIREMENTS - TEST CASE 1 (LEAK PRIVILEGE)

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:

(BankTeller = teller_1) & (Action = Approve) & (LoanInfo = CustomerOne_Loan) \rightarrow decision = Permit
(BankTeller = teller_1) & (Action = Approve) & (LoanInfo = CustomerTwo_Loan) \rightarrow decision = Permit
(BankTeller = teller_2) & (Action = Approve) & (LoanInfo = CustomerOne_Loan) \rightarrow decision = Permit
(BankTeller = teller_2) & (Action = Approve) & (LoanInfo = CustomerTwo_Loan) \rightarrow decision = Permit
(BankTeller = teller_3) & (Action = Approve) & (LoanInfo = CustomerOne_Loan) \rightarrow decision = Permit
(BankTeller = teller_3) & (Action = Approve) & (LoanInfo = CustomerTwo_Loan) \rightarrow 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: BankTestCase1 and these requirements will have been already created for you.

Case 1(s) Summary			I rows out of 1		Search	×
A	ccess Control Security Requirement	:	Requirement Schema		No. of Security Requirement(s)	
	Individual		Test Case 1		6	
urity Requirement (s)	defined under selected Requiremen	t Schema (Test Case 1):	@ 6	rows out of 6	Search	×
urity Requirement (s) Sequence No	defined under selected Requiremen	t Schema (Test Case 1): Resource	Action	rows out of 6 Environment	Search Condition	Decision
-						
	Subject	Resource	Action	Environment	Condition	Decision
	Subject Bank Teller = teller_1	Resource Loaninfo = CustomerOne_Loan	Action BankActions = Approve	Environment Environment = Any Value	Condition Condition = Any Value	Decision Permit
Sequence No 1 2	Subject Bank Teller = teller_1 Bank Teller = teller_1	Resource Loaninfo = CustomerOne_Loan Loaninfo = CustomerTwo_Loan	Action BankActions = Approve BankActions = Approve	Environment Environment = Any Value Environment = Any Value	Condition Condition = Any Value Condition = Any Value	Decision Permit Permit
Sequence No 1 2 3	Subject Bank Teller = teller_1 Bank Teller = teller_1 Bank Teller = teller_2	Resource Loaninfo = CustomerOne_Loan Loaninfo = CustomerTwo_Loan Loaninfo = CustomerOne_Loan	Action BankActions = Approve BankActions = Approve BankActions = Approve	Environment Environment = Any Value Environment = Any Value Environment = Any Value	Condition Condition = Any Value Condition = Any Value Condition = Any Value	Decision Permit Permit Permit

Fig. 5. Individual Security Requirements

5 POLICY VERIFICATION/ANALYZING RESULTS - TEST CASE 1 (LEAK PRIVILEGE)

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 "Leak Privilege" being created. A Leak Privilege occurs when a flaw in your policy logic is authorizing a subject to take an action

you did not intend for them to take. This error can occur due to a mistaken privilege assignment directly or careless privilege inheritance indirectly as well.

In our example, an individual at this bank has been assigned a role of Bank Teller = teller_2 at the facility. We have designed several rules in the BankTeller Policy, a few of which specifying Bank Teller subjects are unable to Approve any LoanInfo resources. In this test case, we are going to verify that this is true.

We will run one "Single Policy" Verification to reveal if their is a Leak Privilege present in our policies. To do this, we will right-click Model Verification and select New Policy Verification. Then we will choose Test Case 1 (security requirement), BankTeller Policy, Single Verification and select run. Again, this will have already been done for you if you open Project File: BankTestCase1.

licy Verification (Jul	/ 3, 2018 15:29:39)(s) Summary			1 rows out of 1					Search	×
Status	Name	Verification Type	Verification Technique	Number of Policy(s)	Combination A	Algorithm	Enforcement	Algorithm	Policy	/ List
UpToDate Pol	icy Verification (July 3, 2018 15:29:39) Standard	Single Policy	1	Deny-over	rrides	Deny Bi	ased	ABAC:BankT	eller Polic
esult(s) with selected	verification (Policy Verification (July	3, 2018 15:29:39))		🐵 6 rows ou	ut of 6				Search	×
esult(s) with selected		3, 2018 15:29:39)) Resource	Action		nment	Conditio	on	Decision		on Result
				Enviro		Condition = A		Ļ	Verificati	
Requirement Schen	na Subject	Resource	n BankActions = Appr	rove Environment	nment		ny Value	Decision	Verificati	on Result
Requirement Schen Test Case 1	Bank Teller = teller_1	Resource Loaninfo = CustomerOne_Loa	n BankActions = Appr n BankActions = Appr	rove Environment	nment t = Any Value	Condition = A	ny Value ny Value	Decision Permit	Verificati FA	on Result
Requirement Schen Test Case 1 Test Case 1	na Subject Bank Teller = teller_1 Bank Teller = teller_1	Resource Loaninfo = CustomerOne_Loa Loaninfo = CustomerTwo_Loa	n BankActions = Appr n BankActions = Appr n BankActions = Appr	rove Environment rove Environment rove Environment	nment t = Any Value t = Any Value	Condition = A Condition = A	ny Value ny Value ny Value	Decision Permit Permit	Verificati FAI FAI	on Result L <mark>SE</mark> L <mark>SE</mark>
Requirement Schen Test Case 1 Test Case 1 Test Case 1 Test Case 1	Subject Bank Teller = teller_1 Bank Teller = teller_1 Bank Teller = teller_2	Resource Loaninfo = CustomerOne_Loa Loaninfo = CustomerTwo_Loa Loaninfo = CustomerOne_Loa	n BankActions = Appr n BankActions = Appr n BankActions = Appr n BankActions = Appr	Environ rove Environment rove Environment rove Environment rove Environment	nment t = Any Value t = Any Value t = Any Value	Condition = A Condition = A Condition = A	ny Value ny Value ny Value ny Value	Decision Permit Permit Permit	Verificati FAI FAI TR	on Result LSE LSE

Fig. 6. BankTeller Policy x Test Case 1

As you can see from our verification results our BankTeller Policy is generating a True result for Security Requirement (4) thus Permitting (Bank Teller = teller_2) to approve CustomerTwo_LoanInfo which is known as a Leak Privilege error.

6 **RESOLVING THIS ERROR - TEST CASE 1 (LEAK PRIVILEGE)**

To solve a Leak Privilege the policy author would need to go back and either update or delete the related rule that is creating this error. To view which specific Rules are resulting in these Verification Results we can click on all (6) of our specific Results (False; False; False; True; False; False) and see which Rules have "Match Results". Since we are already aware of which Result is incorrect let's take a look at it.

See the screenshots below of our Policy Match Results to discover which specific rule is related to our unintended Verification Result (e.g., Permit = True).

WWW.SECURITYPOLICYTOOL.COM

Result(s) with se	lected verification (Policy)	Verification (July	3, 2018 15:29:39))		6 rows out c	of 6		0	Search	×	
Requirement :	Schema Sul	bject	Resource	Action	Environn	nent	Condition	Decision	Verifica	ation Resu	ult
Test Case	a 1 Bank Telle	er = teller_1	Loaninfo = CustomerOne_Loa	n BankActions = Ap	prove Environment =	Any Value Condi	tion = Any Value	Permit	E	ALSE	
Test Case	e 1 Bank Telle	er = teller_1	Loaninfo = CustomerTwo_Loa	n BankActions = Ap			tion = Any Value	Permit		ALSE	
Test Case	a 1 Bank Telle	er = teller_2	LoanInfo = CustomerOne_Loa	n BankActions = Ap			tion = Any Value	Permit	F	ALSE	
Test Case	e 1 Bank Telle	er = teller_2	Loaninfo = CustomerTwo_Loa	n BankActions = Ap			ion = Any Value	Permit	Т	TRUE	
Test Case	e 1 Bank Telle	er = teller_3	LoanInfo = CustomerOne_Loa	n BankActions = Ap	prove Environment =		ion = Any Value	Permit	E	ALSE	
Test Case	e 1 Bank Telle	er = teller_3	Loaninfo = CustomerTwo_Loa	n BankActions = Ap			ion = Any Value	Permit		ALSE	
Policy(s) and Ma	atching result against the s	elcted security re	equirement:		1 rows out of 1				Search	×	
	Policy Name		Rule Combinatio	n Algorithm	Policy Enfo	prcement Algorithm		Com	bined Result		
	ABAC : BankTeller Policy	licy against the s	Deny-ove elcted security requirement:	rrides	© 30 rows o	eny Biased		1	Permit Search	×	1
Rule(s) and Mate		licy against the s		Action			Decision	Inheritance Rela	Search	atch Resu	
Rule(s) and Mate	ching result of Selected Po Subject		elcted security requirement:	Action	30 rows of Environment	out of 30 Condition	1000000	Inheritance Rela	Search ation M	latch Resu	ult
Rule(s) and Mate	ching result of Selected Po Subject Bank Teller = teller_3	Personalinfo	elcted security requirement: Resource = CustomerThree_Personalinfo		30 rows of Environment Environment = Any Value	Condition	Decision Permit	Inheritance Rela Originated	Search ation M. Not	atch Resu Applica	ult
Rule(s) and Mate Sequence No 18	ching result of Selected Po Subject Bank Teller = teller_3 Bank Teller = teller_1	Personalinfo	elcted security requirement: Resource = CustomerThree_PersonalInfo fo = CustomerOne_Loan	Action BankActions = View BankActions = View	30 rows of Environment Environment = Any Value Environment = Any Value	Condition Condition = Any Value Condition = Any Value	Permit Permit	Inheritance Rela Originated Originated	Search ation M Not Not	atch Resu Applica Applica	ult abl
Rule(s) and Mate Sequence No 18 19	ching result of Selected Po Subject Bank Teller = teller_3 Bank Teller = teller_1 Bank Teller = teller_1	Personalinfo Loanin Loanin	elcted security requirement: Resource = CustomerThree_Personalinfo fo = CustomerCne_Loan fo = CustomerTwo_Loan	Action BankActions = View	30 rows of Environment Environment = Any Value Environment = Any Value Environment = Any Value	Condition Condition = Any Value Condition = Any Value Condition = Any Value	Permit	Inheritance Rela Originated Originated Originated	Search ation M Not Not Not	atch Resu Applica Applica Applica	ult abl abl
Rule(s) and Mate Sequence No 18 19 20	ching result of Selected Po Subject Bank Teller = teller_3 Bank Teller = teller_1	Personalinfo Loanin Loanin Loanin	elcted security requirement: Resource = CustomerThree_PersonalInfo fo = CustomerOne_Loan	Action BankActions = View BankActions = View BankActions = View	S0 rows o Environment Environment = Any Value Environment = Any Value Environment = Any Value Environment = Any Value	Condition Condition = Any Value Condition = Any Value Condition = Any Value Condition = Any Value	Permit Permit Permit	Inheritance Rela Originated Originated	Search Search Not Not Not Not Not Not	Applica Applica Applica Applica Applica	ult abl abl
Rule(s) and Mate Sequence No 18 19 20 21	ching result of Selected Po Subject Bank Teller = teller_3 Bank Teller = teller_1 Bank Teller = teller_1 Bank Teller = teller_2	Personalinfo Loanin Loanin Loanin Loanin	elcted security requirement: Resource = CustomerThree_Personalinfo fo = CustomerOne_Loan fo = CustomerVmo_Loan fo = CustomerOne_Loan	Action BankActions = View BankActions = View BankActions = View BankActions = View	S0 rows of Environment = Any Value Environment = Any Value Environment = Any Value Environment = Any Value Environment = Any Value	Condition Condition = Any Value Condition = Any Value Condition = Any Value Condition = Any Value	Permit Permit Permit Permit	Inheritance Rela Originated Originated Originated Originated	Search ation M Not Not Not Not	Applica Applica Applica Applica Applica Applica	ult abl abl abl
Rule(s) and Mate Sequence No 18 19 20 21 21 22	ching result of Selected Po Subject Bank Teller = teller_3 Bank Teller = teller_1 Bank Teller = teller_1 Bank Teller = teller_2 Bank Teller = teller_2	Personalinfo Loanin Loanin Loanin Loanin Loanin	elcted security requirement: Resource = CustomerThree_Personalinfo fo = CustomerTwo_Loan fo = CustomerTwo_Loan fo = CustomerTwo_Loan	Action BankActions = View BankActions = View BankActions = View BankActions = View	S0 rows o Environment Environment = Any Value Environment = Any Value Environment = Any Value Environment = Any Value	Condition Condition = Any Value Condition = Any Value Condition = Any Value Condition = Any Value	Permit Permit Permit Permit Permit	Inheritance Rela Originated Originated Originated Originated Originated	Search stion M Not Not Not Not Not	Applica Applica Applica Applica Applica Applica Applica	ult abl abl abl abl
Rule(s) and Matr Sequence No 18 19 20 21 21 22 23	ching result of Selected Po Subject Bank Teller = teller_3 Bank Teller = teller_1 Bank Teller = teller_1 Bank Teller = teller_2 Bank Teller = teller_2 Bank Teller = teller_3	Personalinfo Loanin Loanin Loanin Loanin Loanin Loanin	elcted security requirement: Resource = CustomerThree_Personalinfo fo = CustomerOne_Loan fo = CustomerOne_Loan fo = CustomerOne_Loan fo = CustomerOne_Loan	Action BankActions = View BankActions = View BankActions = View BankActions = View BankActions = View	30 rows of Environment = Any Value Environment = Any Value	Condition Condition = Any Value Condition = Any Value Condition = Any Value Condition = Any Value Condition = Any Value	Permit Permit Permit Permit Permit Permit	Inheritance Rela Originated Originated Originated Originated Originated Originated	Search Search Not	atch Resu Applica Applica Applica Applica Applica Applica	ult abl abl abl abl abl
Rule(s) and Mate Sequence No 18 19 20 21 22 21 22 23 24	ching result of Selected Po Subject Bank Teller = teller_3 Bank Teller = teller_1 Bank Teller = teller_2 Bank Teller = teller_2 Bank Teller = teller_2 Bank Teller = teller_3	Personalinfo Loanin Loanin Loanin Loanin Loanin Loanin Loanin	eleted security requirement: Resource = CustomerThree_Personalinfo fo = CustomerTwo_Loan fo = CustomerTwo_Loan fo = CustomerTwo_Loan fo = CustomerTwo_Loan fo = CustomerTwo_Loan fo = CustomerTwo_Loan	Action BankActions = View BankActions = View BankActions = View BankActions = View BankActions = View BankActions = View	30 rows of Environment = Any Value Environment = Any Value	Condition Condition = Any Value Condition = Any Value	Permit Permit Permit Permit Permit Permit Permit	Inheritance Rela Originated Originated Originated Originated Originated Originated Originated	Search Stion M Not	Applica Applica Applica Applica Applica Applica Applica	ult abl abl abl abl abl
Sequence No 18 19 20 21 22 23 24 25	ching result of Selected Po Subject Bank Teller = teller_3 Bank Teller = teller_1 Bank Teller = teller_1 Bank Teller = teller_2 Bank Teller = teller_3 Bank Teller = teller_3 Bank Teller = teller_3 Bank Teller = teller_1	Personalinfo Loanin Loanin Loanin Loanin Loanin Loanin Loanin Loanin	elcted security requirement: Resource = CustomerThree_Personalinfo fo = CustomerThree_Loan fo = CustomerTwo_Loan fo = CustomerTwo_Loan fo = CustomerTwo_Loan fo = CustomerTwo_Loan fo = CustomerTwo_Loan fo = CustomerTwo_Loan	Action BankActions = View BankActions = View BankActions = View BankActions = View BankActions = View BankActions = View BankActions = View	S0 rows of Environment = Any Value	Condition Condition = Any Value Condition = Any Value	Permit Permit Permit Permit Permit Permit Deny	Inheritance Rela Originated Originated Originated Originated Originated Originated Originated	Search Search Not	Applica Applica Applica Applica Applica Applica Applica Applica	ult abl abl abl abl abl abl abl
Rule(s) and Mate Sequence No 18 19 20 21 22 23 24 25 26	ching result of Selected Po Subject Bank Teller = teller_3 Bank Teller = teller_1 Bank Teller = teller_1 Bank Teller = teller_2 Bank Teller = teller_3 Bank Teller = teller_3 Bank Teller = teller_3 Bank Teller = teller_1	Personalinfo Loanin Loanin Loanin Loanin Loanin Loanin Loanin Loanin	eletted security requirement: Resource = CustomerThree_Personalinfo fo = CustomerTwo_Loan fo = CustomerTwo_Loan	Action BankActions = View BankActions = View BankActions = View BankActions = View BankActions = View BankActions = Approve BankActions = Approve	30 rows of Environment = Any Value Environment = Any Value	Condition Condition = Any Value Condition = Any Value	Permit Permit Permit Permit Permit Permit Deny Deny	Inheritance Rela Originated Originated Originated Originated Originated Originated Originated Originated	Search Search Not	Applica Applica Applica Applica Applica Applica Applica Applica Applica	ult abl abl abl abl abl abl abl
Rule(s) and Mate Sequence No 18 19 20 21 22 23 24 25 26 27	ching result of Selected Po Subject Bank Teller = teller_3 Bank Teller = teller_1 Bank Teller = teller_1 Bank Teller = teller_2 Bank Teller = teller_3 Bank Teller = teller_3 Bank Teller = teller_1 Bank Teller = teller_1 Bank Teller = teller_1 Bank Teller = teller_2	Personalinfo Loanin Loanin Loanin Loanin Loanin Loanin Loanin Loanin Loanin	elcted security requirement: Resource = CustomerThree_Personalinfo fo = CustomerOne_Loan fo = CustomerOne_Loan fo = CustomerTwo_Loan fo = CustomerTwo_Loan fo = CustomerTwo_Loan fo = CustomerTwo_Loan fo = CustomerTwo_Loan fo = CustomerTwo_Loan fo = CustomerOne_Loan fo = CustomerOne_Loan	Action BankActions = View BankActions = View BankActions = View BankActions = View BankActions = View BankActions = Approve BankActions = Approve BankActions = Approve	30 rows of Environment = Any Value Environment = Any Value	Condition Condition = Any Value Condition = Any Value	Permit Permit Permit Permit Permit Permit Permit Deny Deny Deny	Inheritance Rela Originated Originated Originated Originated Originated Originated Originated Originated Originated	Search ttion Not	Applica Applica Applica Applica Applica Applica Applica Applica Applica Applica	ult abl abl abl abl abl abl abl

Fig. 7. BankTeller Policy: Match Results

Now that we have pinpointed our rule related to our Leak Privilege Error we can go back and make changes or possibly remove this rule. 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 28 in the BankTeller Policy to Deny "BankTeller = teller_2" to Approve CustomerTwo_Loan which will in turn resolve the Leak Privilege. For this example, we are fixing our mistake in unintentionally selecting Permit for this individual (teller_2) to be able to Approve the resource.

BankTeller Policy: Modify (28) Rule:

```
(Rule No. = 28) \rightarrow (Bank Teller = teller_2) \rightarrow (Action = Approve) \rightarrow(Resource = CustomerTwo_Loan) \rightarrow decision = Deny
```

Fig. 8. BankTeller Policy: Modified Rule (28)

LoanInfo = CustomerTwo_Loan

Bank Teller = teller_2

Which then when we "Refresh" our previous Verification Results we no longer have a Leak Privilege occurring:

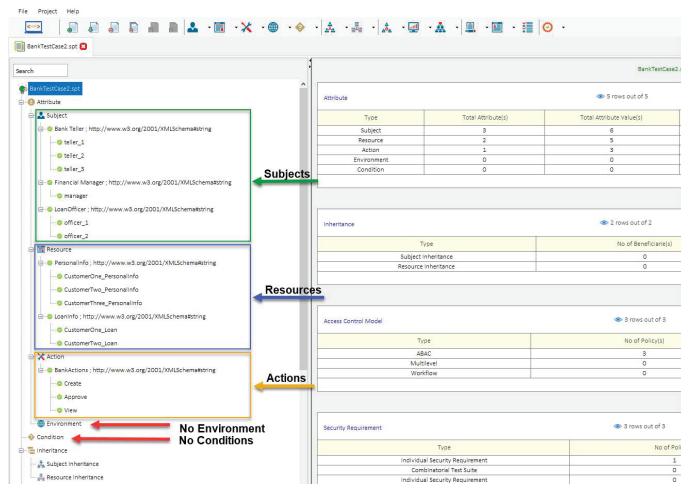
Originated

licy Verification (Ju	uly 4, 2018 18:51:30)(s) Summary			I rows out of 1				Search	×
Status	Name	Verification Type	Verification Technique	Number of Policy(s)	Combination Algorithm	n Enforcemer	nt Algorithm	Policy Li	st
IpToDate Po	Policy Verification (July 4, 2018 18:51:3	0) Standard	Single Policy	1	Deny-overrides	Deny	Biased	ABAC:BankTelle	r Polic
•sult(s) with selecte	ted verification (Policy Verification (July	(4, 2018 18:51:30))		👁 6 rows ou	ut of 6		s	Search	×
			Action			Condition			
	iema Subject	Resource	Action BankActions = App	Enviro	nment	Condition	Decision Permit	Verification	Result
Requirement Sche			n BankActions = App	rove Environment	nment t = Any Value Cond	tion = Any Value	Decision	Verification FALSE	Result
Requirement Sche Test Case 1	ema Subject Bank Teller = teller_1	Resource Loaninfo = CustomerOne_Loa	n BankActions = App n BankActions = App	rove Environment rove Environment	nment t = Any Value Cond t = Any Value Cond	anna a bhaire.	Decision Permit	Verification	Result
Requirement Sche Test Case 1 Test Case 1	Bank Teller = teller_1 Bank Teller = teller_1	Resource Loaninfo = CustomerOne_Loa Loaninfo = CustomerTwo_Loa	n BankActions = App an BankActions = App an BankActions = App	Environment rove Environment rove Environment rove Environment	nment = Any Value Cond = Any Value Cond = Any Value Cond	tion = Any Value tion = Any Value	Decision Permit Permit	Verification FALSE FALSE	Result
Requirement Sche Test Case 1 Test Case 1 Test Case 1	Bank Teller = teller_1 Bank Teller = teller_1 Bank Teller = teller_1 Bank Teller = teller_2	Resource Loaninfo = CustomerOne_Loa Loaninfo = CustomerTwo_Loa Loaninfo = CustomerOne_Loa	in BankActions = App in BankActions = App in BankActions = App in BankActions = App	Environment rove Environment rove Environment rove Environment rove Environment	nment Cond t = Any Value Cond	tion = Any Value tion = Any Value tion = Any Value	Decision Permit Permit Permit	Verification FALSE FALSE FALSE	Result

Fig. 9. Updated Results: BankTeller Policy (No Leak Privilege)

7 SETTING UP THE POLICIES – TEST CASE 2 (BLOCK PRIVILEGE)

This bank example contains three policies (BankTeller Policy & LoanOfficer Policy & FManager Policy). The attributes in this example have not been changed from previous Test Case 1. The Attribute/Attribute Values included in these policies are as shown in Figure 10.





8 MODELING YOUR POLICY - TEST CASE 2 (BLOCK PRIVILEGE)

Now that we have entered our attributes we can model our three policies (BankTeller Policy & LoanOfficer Policy & FManager 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:

BankTeller Policy:

(Bank Teller = teller_1, CustomerOne_PersonalInfo, Create) \rightarrow Permit (Bank Teller = teller_1, CustomerOne_PersonalInfo, View) →Permit (Bank Teller = teller_1, CustomerTwo_PersonalInfo, Create) →Permit (Bank Teller = teller_1, CustomerTwo_PersonalInfo, View) →Permit (Bank Teller = teller_1, CustomerThree_PersonalInfo, Create) \rightarrow Permit (Bank Teller = teller_1, CustomerThree_PersonalInfo, View) \rightarrow Permit (Bank Teller = teller_2, CustomerOne_PersonalInfo, Create) →Permit (Bank Teller = teller_2, CustomerOne_PersonalInfo, View) \rightarrow Permit (Bank Teller = teller_2, CustomerTwo_PersonalInfo, Create) →Permit (Bank Teller = teller_2, CustomerTwo_PersonalInfo, View) →Permit (Bank Teller = teller_2, CustomerThree_PersonalInfo, Create) \rightarrow Permit (Bank Teller = teller_2, CustomerThree_PersonalInfo, View) →Permit (Bank Teller = teller_3, CustomerOne_PersonalInfo, Create) →Permit (Bank Teller = teller_3, CustomerOne_PersonalInfo, View) →Permit (Bank Teller = teller_3, CustomerTwo_PersonalInfo, Create) →Permit (Bank Teller = teller_3, CustomerTwo_PersonalInfo, View) →Permit (Bank Teller = teller_3, CustomerThree_PersonalInfo, Create) \rightarrow Permit (Bank Teller = teller_3, CustomerThree_PersonalInfo, View) →Permit (Bank Teller = teller_1, CustomerOne_Loan, View) →Permit (Bank Teller = teller_1, CustomerTwo_Loan, View) →Permit (Bank Teller = teller_2, CustomerOne_Loan, View) →Permit (Bank Teller = teller_2, CustomerTwo_Loan, View) →Permit (Bank Teller = teller_3, CustomerOne_Loan, View) →Permit (Bank Teller = teller_3, CustomerTwo_Loan, View) →Permit (Bank Teller = teller_1, CustomerOne_Loan, Approve) →Deny (Bank Teller = teller_1, CustomerTwo_Loan, Approve) →Deny (Bank Teller = teller_2, CustomerOne_Loan, Approve) →Deny (Bank Teller = teller_2, CustomerTwo_Loan, Approve) →Deny (Bank Teller = teller_3, CustomerOne_Loan, Approve) \rightarrow Deny (Bank Teller = teller_3, CustomerTwo_Loan, Approve) \rightarrow Deny

LoanOfficer Policy:

FManager Policy:

(Manager = manager, CustomerThree_PersonalInfo, Create) →Permit (Manager = manager, CustomerThree_PersonalInfo, View) →Permit (Manager = manager, CustomerOne_Loan, Create) →Permit (Manager = manager, CustomerTwo_Loan, View) →Permit (Manager = manager, CustomerTwo_Loan, Create) →Permit (Manager = manager, CustomerTwo_Loan, View) →Permit (Manager = manager, CustomerTwo_Loan, Approve) →Permit (Manager = manager, CustomerTwo_Loan, Approve) →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: BankTestCase2 and these policies will have been already created for you.

ankTeller Policy Po	olic <mark>y(</mark> s) Summary		I rows out of 1				Search
Model	Policy Name	Rule Combination Algorithm	Policy Enforcement Algorithm	No. of Rule(s)	Time Created		Last Modified
ABAC	BankTeller Policy	Deny-overrides	Deny Biased	30	July 3, 2018 14:28:	32	July 3, 2018 14:28:32
ule (s) defined wit	h selected policy (BankTeller	Policy):	🐵 30 row	s out of 30			Search
Sequence No	Subject	Resource	Action	Environment	Condition	Decision	Inheritance Relat
1	Bank Teller = teller_1	Personalinfo = CustomerOne_Personalinfo	BankActions = Create	Environment = Any Value	Condition = Any Value	Permit	Originated
2	Bank Teller = teller_1	Personalinfo = CustomerOne_Personalinfo	BankActions = View	Environment = Any Value	Condition = Any Value	Permit	Originated
3	Bank Teller = teller_1	Personalinfo = CustomerTwo_Personalinfo	BankActions = Create	Environment = Any Value	Condition = Any Value	Permit	Originated
4	Bank Teller = teller_1	Personalinfo = CustomerTwo_Personalinfo	BankActions = View	Environment = Any Value	Condition = Any Value	Permit	Originated
5	Bank Teller = teller_1	Personalinfo = CustomerThree_Personalinfo	BankActions = Create	Environment = Any Value	Condition = Any Value	Permit	Originated
6	Bank Teller = teller_1	Personalinfo = CustomerThree_Personalinfo	BankActions = View	Environment = Any Value	Condition = Any Value	Permit	Originated
7	Bank Teller = teller_2	Personalinfo = CustomerOne_Personalinfo	BankActions = Create	Environment = Any Value	Condition = Any Value	Permit	Originated
8	Bank Teller = teller_2	Personalinfo = CustomerOne_Personalinfo	BankActions = View	Environment = Any Value	Condition = Any Value	Permit	Originated
9	Bank Teller = teller_2	Personalinfo = CustomerTwo_Personalinfo	BankActions = Create	Environment = Any Value	Condition = Any Value	Permit	Originated
10	Bank Teller = teller_2	Personalinfo = CustomerTwo_Personalinfo	BankActions = View	Environment = Any Value	Condition = Any Value	Permit	Originated
11	Bank Teller = teller_2	Personalinfo = CustomerThree_Personalinfo	BankActions = Create	Environment = Any Value	Condition = Any Value	Permit	Originated
12	Bank Teller = teller_2	Personalinfo = CustomerThree_Personalinfo	BankActions = View	Environment = Any Value	Condition = Any Value	Permit	Originated
13	Bank Teller = teller_3	Personalinfo = CustomerOne_Personalinfo	BankActions = Create	Environment = Any Value	Condition = Any Value	Permit	Originated
14	Bank Teller = teller_3	Personalinfo = CustomerOne_Personalinfo	BankActions = View	Environment = Any Value	Condition = Any Value	Permit	Originated
15	Bank Teller = teller_3	Personalinfo = CustomerTwo_Personalinfo	BankActions = Create	Environment = Any Value	Condition = Any Value	Permit	Originated
16	Bank Teller = teller_3	Personalinfo = CustomerTwo_Personalinfo	BankActions = View	Environment = Any Value	Condition = Any Value	Permit	Originated
17	Bank Teller = teller_3	Personalinfo = CustomerThree_Personalinfo	BankActions = Create	Environment = Any Value	Condition = Any Value	Permit	Originated
18	Bank Teller = teller_3	Personalinfo = CustomerThree_Personalinfo	BankActions = View	Environment = Any Value	Condition = Any Value	Permit	Originated
19	Bank Teller = teller_1	Loaninfo = CustomerOne_Loan	BankActions = View	Environment = Any Value	Condition = Any Value	Permit	Originated
20	Bank Teller = teller_1	Loaninfo = CustomerTwo_Loan	BankActions = View	Environment = Any Value	Condition = Any Value	Permit	Originated
21	Bank Teller = teller_2	Loaninfo = CustomerOne_Loan	BankActions = View	Environment = Any Value	Condition = Any Value	Permit	Originated
22	Bank Teller = teller_2	LoanInfo = CustomerTwo_Loan	BankActions = View	Environment = Any Value	Condition = Any Value	Permit	Originated
23	Bank Teller = teller_3	Loaninfo = CustomerOne_Loan	BankActions = View	Environment = Any Value	Condition = Any Value	Permit	Originated
24	Bank Teller = teller_3	LoanInfo = CustomerTwo_Loan	BankActions = View	Environment = Any Value	Condition = Any Value	Permit	Originated
25	Bank Teller = teller_1	Loaninfo = CustomerOne_Loan	BankActions = Approve	Environment = Any Value	Condition = Any Value	Deny	Originated
26	Bank Teller = teller_1	LoanInfo = CustomerTwo_Loan	BankActions = Approve	Environment = Any Value	Condition = Any Value	Deny	Originated
27	Bank Teller = teller_2	LoanInfo = CustomerOne_Loan	BankActions = Approve	Environment = Any Value	Condition = Any Value	Deny	Originated
28	Bank Teller = teller_2	Loaninfo = CustomerTwo_Loan	BankActions = Approve	Environment = Any Value	Condition = Any Value	Deny	Originated
29	Bank Teller = teller_3	Loaninfo = CustomerOne_Loan	BankActions = Approve	Environment = Any Value	Condition = Any Value	Deny	Originated
30	Bank Teller = teller_3	Loaninfo = CustomerTwo_Loan	BankActions = Approve	Environment = Any Value	Condition = Any Value	Deny	Originated

Fig. 11. BankTeller Policy

WWW.SECURITYPOLICYTOOL.COM

oanOfficer Policy	Policy(s) Summary		I rows out of 1				Search	×
Model	Policy Name	Rule Combination Algorithm	Policy Enforcement Algorithm	No. of Rule(s)	Time Created		Last Modified	
ABAC	LoanOfficer Policy	Deny-overrides	Deny Biased	24	July 3, 2018 14:40:4	16	July 3, 2018 14:40:46	6
Rule (s) defined wi	th selected policy (LoanOfficer F	Policy):	24 rows	s out of 24			Search	×
Sequence No	Subject	Resource	Action	Environment	Condition	Decision	Inheritance Relat	tion
1	LoanOfficer = officer_1	Personalinfo = CustomerOne_Personalinfo	BankActions = View	Environment = Anv Value	Condition = Any Value	Permit	Originated	_
2	LoanOfficer = officer_1	Personalinfo = CustomerOne_Personalinfo	BankActions = Create	Environment = Any Value	Condition = Any Value	Permit	Originated	
3	LoanOfficer = officer_1	Personalinfo = CustomerTwo_Personalinfo	BankActions = View	Environment = Anv Value	Condition = Any Value	Permit	Originated	
4	LoanOfficer = officer_1	Personalinfo = CustomerTwo_Personalinfo	BankActions = Create	Environment = Any Value	Condition = Any Value	Permit	Originated	
5	LoanOfficer = officer_1	Personalinfo = CustomerThree_Personalinfo	BankActions = View	Environment = Any Value	Condition = Any Value	Permit	Originated	
6	LoanOfficer = officer_1	Personalinfo = CustomerThree_Personalinfo	BankActions = Create	Environment = Any Value	Condition = Any Value	Permit	Originated	
7	LoanOfficer = officer_1	Loaninfo = CustomerOne_Loan	BankActions = View	Environment = Any Value	Condition = Any Value	Permit	Originated	
8	LoanOfficer = officer_1	Loaninfo = CustomerOne_Loan	BankActions = Create	Environment = Any Value	Condition = Any Value	Permit	Originated	
9	LoanOfficer = officer_1	Loaninfo = CustomerTwo_Loan	BankActions = View	Environment = Any Value	Condition = Any Value	Deny	Originated	
10	LoanOfficer = officer_1	Loaninfo = CustomerTwo_Loan	BankActions = Create	Environment = Any Value	Condition = Any Value	Permit	Originated	
11	LoanOfficer = officer_2	Personalinfo = CustomerOne_Personalinfo	BankActions = View	Environment = Anv Value	Condition = Any Value	Permit	Originated	
12	LoanOfficer = officer_2	Personalinfo = CustomerOne_Personalinfo	BankActions = Create	Environment = Any Value	Condition = Any Value	Permit	Originated	
13	LoanOfficer = officer_2	Personalinfo = CustomerTwo_Personalinfo	BankActions = View	Environment = Any Value	Condition = Any Value	Permit	Originated	
14	LoanOfficer = officer_2	Personalinfo = CustomerTwo_Personalinfo	BankActions = Create	Environment = Any Value	Condition = Any Value	Permit	Originated	
15	LoanOfficer = officer_2	Personalinfo = CustomerThree_Personalinfo	BankActions = View	Environment = Any Value	Condition = Any Value	Permit	Originated	
16	LoanOfficer = officer_2	Personalinfo = CustomerThree_Personalinfo	BankActions = Create	Environment = Any Value	Condition = Any Value	Permit	Originated	
17	LoanOfficer = officer_2	Loaninfo = CustomerOne_Loan	BankActions = View	Environment = Any Value	Condition = Any Value	Permit	Originated	
18	LoanOfficer = officer_2	Loaninfo = CustomerOne_Loan	BankActions = Create	Environment = Any Value	Condition = Any Value	Permit	Originated	
19	LoanOfficer = officer_2	Loaninfo = CustomerTwo_Loan	BankActions = View	Environment = Any Value	Condition = Any Value	Permit	Originated	
20	LoanOfficer = officer_2	Loaninfo = CustomerTwo_Loan	BankActions = Create	Environment = Any Value	Condition = Any Value	Permit	Originated	
21	LoanOfficer = officer_1	Loaninfo = CustomerOne_Loan	BankActions = Approve	Environment = Any Value	Condition = Any Value	Deny	Originated	
22	LoanOfficer = officer_1	Loaninfo = CustomerTwo_Loan	BankActions = Approve	Environment = Any Value	Condition = Any Value	Deny	Originated	
23	LoanOfficer = officer_2	Loaninfo = CustomerOne_Loan	BankActions = Approve	Environment = Any Value	Condition = Any Value	Deny	Originated	
24	LoanOfficer = officer 2	Loaninfo = CustomerTwo_Loan	BankActions = Approve	Environment = Anv Value	Condition = Any Value	Denv	Originated	_

Fig. 12. LoanOfficer Policy

Manager Policy P	Policy(s) Summary		I rows out of 1	S	earch 🚺		
Model	Policy Name	Rule Combination Algorithm P	olicy Enforcement Algorithm	No. of Rule(s)	Time Created		Last Modified
ABAC	FManager Policy	Permit-overrides	Permit Biased	12	July 3, 2018 15:03:03	J	uly 3, 2018 15:03:03
Rule (s) defined wi	ith selected policy (FManager Policy)		12 rows of	ut of 12		Si	earch 🚺
Sequence No	Subject	Resource	Action	Environment	Condition	Decision	Inheritance Relatio
1	Financial Manager = manager	Personalinfo = CustomerOne_Personalinfo	BankActions = Create	Environment = Any Value	Condition = Any Value	Permit	Originated
2	Financial Manager = manager	Personalinfo = CustomerOne_Personalinfo	BankActions = View	Environment = Any Value	Condition = Any Value	Permit	Originated
3	Financial Manager = manager	Personalinfo = CustomerTwo_Personalinfo	BankActions = Create	Environment = Any Value	Condition = Any Value	Permit	Originated
4	Financial Manager = manager	Personalinfo = CustomerTwo_Personalinfo	BankActions = View	Environment = Any Value	Condition = Any Value	Permit	Originated
5	Financial Manager = manager	Personalinfo = CustomerThree_Personalinfo	BankActions = Create	Environment = Any Value	Condition = Any Value	Permit	Originated
6	Financial Manager = manager	Personalinfo = CustomerThree_Personalinfo	BankActions = View	Environment = Any Value	Condition = Any Value	Permit	Originated
7	Financial Manager = manager	Loaninfo = CustomerOne_Loan	BankActions = Create	Environment = Any Value	Condition = Any Value	Permit	Originated
8	Financial Manager = manager	LoanInfo = CustomerOne_Loan	BankActions = View	Environment = Any Value	Condition = Any Value	Permit	Originated
9	Financial Manager = manager	Loaninfo = CustomerTwo_Loan	BankActions = Create	Environment = Any Value	Condition = Any Value	Permit	Originated
10	Financial Manager = manager	LoanInfo = CustomerTwo_Loan	BankActions = View	Environment = Any Value	Condition = Any Value	Permit	Originated
11	Financial Manager = manager	Loaninfo = CustomerOne_Loan	BankActions = Approve	Environment = Any Value	Condition = Any Value	Permit	Originated

Fig. 13. FManager Policy

9 INDIVIDUAL SECURITY REQUIREMENT - TEST CASE 2 (BLOCK PRIVILEGE)

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 requirements below:

Individual Security Requirements:

```
(Loan Officer = officer_1) & (Action = Create) & (LoanInfo = CustomerOne_Loan) \rightarrow decision = Permit
(Loan Officer = officer_1) & (Action = View) & (LoanInfo = CustomerOne_Loan) \rightarrow decision = Permit
(Loan Officer = officer_1) & (Action = Create) & (LoanInfo = CustomerTwo_Loan) \rightarrow decision = Permit
(Loan Officer = officer_1) & (Action = View) & (LoanInfo = CustomerTwo_Loan) \rightarrow decision = Permit
(Loan Officer = officer_2) & (Action = Create) & (LoanInfo = CustomerTwo_Loan) \rightarrow decision = Permit
```

(Loan Officer = officer_2) & (Action = View) & (LoanInfo = CustomerOne_Loan) \rightarrow decision = Permit (Loan Officer = officer_2) & (Action = Create) & (LoanInfo = CustomerTwo_Loan) \rightarrow decision = Permit (Loan Officer = officer 2) & (Action = View) & (LoanInfo = CustomerTwo Loan) \rightarrow 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: BankTestCase2 and these requirements will have been already created for you.

t Case 2(s) Summary			1 rows out of 1	Searc	h 🚺		
Д	ccess Control Security Requirement		Requirement Schema		No. of Security Requirement(s)		
	Individual		Test Case 2		8		
curity Requirement (s)	defined under selected Requirement S	ichema (Test Case 2):	@ 8	rows out of 8	Searc	h 🚺	
Sequence No.	Subject	Perource	Action	Environment	Condition	Decision	
Sequence No	Subject	Resource	Action BankActions = Create	Environment	Condition	Decision	
Sequence No	Subject LoanOfficer = officer_1 LoanOfficer = officer_1	Resource Loaninfo = CustomerOne_Loan Loaninfo = CustomerOne_Loan	Action BankActions = Create BankActions = View	Environment = Any Value	Condition = Any Value	Decision Permit Permit	
1	LoanOfficer = officer_1	Loaninfo = CustomerOne_Loan	BankActions = Create	Environment = Any Value Environment = Any Value	Condition = Any Value Condition = Any Value	Permit	
1 2	LoanOfficer = officer_1 LoanOfficer = officer_1	Loaninfo = CustomerOne_Loan Loaninfo = CustomerOne_Loan	BankActions = Create BankActions = View	Environment = Any Value Environment = Any Value Environment = Any Value	Condition = Any Value Condition = Any Value Condition = Any Value	Permit Permit	
1 2	LoanOfficer = officer_1 LoanOfficer = officer_1 LoanOfficer = officer_1	Loaninfo = CustomerOne_Loan Loaninfo = CustomerOne_Loan Loaninfo = CustomerTwo_Loan	BankActions = Create BankActions = View BankActions = Create	Environment = Any Value Environment = Any Value Environment = Any Value Environment = Any Value	Condition = Any Value Condition = Any Value Condition = Any Value Condition = Any Value	Permit Permit Permit	
1 2 3 4	LoanOfficer = officer_1 LoanOfficer = officer_1 LoanOfficer = officer_1 LoanOfficer = officer_1	Loaninfo = CustomerOne_Loan Loaninfo = CustomerOne_Loan Loaninfo = CustomerTwo_Loan Loaninfo = CustomerTwo_Loan	BankActions = Create BankActions = View BankActions = Create BankActions = View	Environment = Any Value Environment = Any Value Environment = Any Value	Condition = Any Value Condition = Any Value Condition = Any Value	Permit Permit Permit Permit	
1 2 3 4 5	LoanOfficer = officer_1 LoanOfficer = officer_1 LoanOfficer = officer_1 LoanOfficer = officer_1 LoanOfficer = officer_2	Loaninfo = CustomerOne_Loan Loaninfo = CustomerOne_Loan Loaninfo = CustomerTwo_Loan Loaninfo = CustomerTwo_Loan Loaninfo = CustomerOne_Loan	BankActions = Create BankActions = View BankActions = Create BankActions = View BankActions = Create	Environment = Any Value Environment = Any Value Environment = Any Value Environment = Any Value Environment = Any Value	Condition = Any Value Condition = Any Value Condition = Any Value Condition = Any Value Condition = Any Value	Permit Permit Permit Permit Permit	

Fig. 14. Individual Security Requirement

10 POLICY VERIFICATION/ANALYZING RESULTS - TEST CASE 2 (BLOCK PRIVI-LEGE)

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 "Block Privilege" error being created. A Block Privilege error occurs when policy rules are blocking a subject's legitimate access to rightful resources. A Block Privilege is created when a policy cannot render a grant or deny decision, no available logic in the AC policy algorithm for evaluating the access request, or by mistaken privilege assignment directly.

For example, when the policy author was designing the logic for these bank policies; the author intended all LoanOfficers to be able to Create all LoanInfo resources. However, to be able to Create all LoanInfo resources, LoanOfficers also need to be able to View all LoanInfo resources. To ensure the policy has been designed correctly let's verify that this is true (e.g., LoanOfficer \rightarrow View or Create \rightarrow LoanInfo Resources).

We will run one "Merged Policy" Verification with all three of our policies to reveal the Block Privilege error that is present in our policies. To do this, we will select Test Case 2 (security requirement) and BankTeller Policy & LoanOfficer Policy & FManager Policy as a Merged Policy Verification and analyze our verification result. Again, this will have already been done for you if you open Project File: BankTestCase2.

Policy Verification (July 3, 20	018 16:16:57)(s) Summa	ry			1 rows out of 1				Search	×
Status	Name	Verification Type	Verification Technique	Number of Policy(s)	Combination Algorithm	Enforcement Algorithm		Policy List	t	
ToDate Policy Verification	(July 3, 2018 16:16:57)	Standard	Merged Policy	3	Deny-overrides	Deny Biased	ABAC:BankTeller Policy,	ABAC:LoanOffice	er Policy, ABAC:FMar	ager Po
		n (July 3, 2018 16:		Action		vs out of 8	Condition	Decision	Search	Result
Requirement Schema	Subject		Resource	Action BankActions	n E	nvironment	Condition	Decision	Verification	Result
		er_1 Loani		BankActions	e Create Environ	nvironment ment = Any Value	Condition = Any Value		Verification TRUE	Result
Requirement Schema Test Case 2	Subject LoanOfficer = office	r_1 Loani r_1 Loani	Resource nfo = CustomerOne_Loan	BankActions BankActions	= Create Environ = View Environ	nvironment		Permit	Verification TRUE TRUE	Result
Requirement Schema Test Case 2 Test Case 2	Subject LoanOfficer = office LoanOfficer = office	r_1 Loani r_1 Loani r_1 Loani	Resource nfo = CustomerOne_Loan nfo = CustomerOne_Loan	BankActions BankActions BankActions	Create Environ View Environ Create Environ Create Environ	nvironment ment = Any Value ment = Any Value	Condition = Any Value Condition = Any Value	Permit Permit	Verification TRUE	Result
Requirement Schema Test Case 2 Test Case 2 Test Case 2 Test Case 2	Subject LoanOfficer = office LoanOfficer = office LoanOfficer = office	r_1 Loani r_1 Loani r_1 Loani r_1 Loani r_1 Loani	Resource nfo = CustomerOne_Loan nfo = CustomerOne_Loan nfo = CustomerTwo_Loan	BankActions BankActions BankActions BankActions BankActions	E Create Environ = View Environ = Create Environ = View Environ	nvironment ment = Any Value ment = Any Value ment = Any Value	Condition = Any Value Condition = Any Value Condition = Any Value	Permit Permit Permit	Verification TRUE TRUE TRUE	Result
Requirement Schema Test Case 2 Test Case 2 Test Case 2 Test Case 2 Test Case 2	Subject LoanOfficer = office LoanOfficer = office LoanOfficer = office LoanOfficer = office	r_1 Loani r_1 Loani r_1 Loani r_1 Loani r_2 Loani	Resource nfo = CustomerOne_Loan nfo = CustomerOne_Loan nfo = CustomerTwo_Loan nfo = CustomerTwo_Loan	BankActions BankActions BankActions BankActions BankActions	E Create Environ = View Environ = View Environ = View Environ = Create Environ	mvironment ment = Any Value ment = Any Value ment = Any Value ment = Any Value	Condition = Any Value Condition = Any Value Condition = Any Value Condition = Any Value	Permit Permit Permit Permit	Verification TRUE TRUE TRUE FALSE	Result
Test Case 2 Test Case 2 Test Case 2 Test Case 2 Test Case 2 Test Case 2	Subject LoanOfficer = office LoanOfficer = office LoanOfficer = office LoanOfficer = office LoanOfficer = office	rr_1 Loani tr_1 Loani tr_1 Loani tr_1 Loani tr_2 Loani tr_2 Loani	Resource nfo = CustomerOne_Loan nfo = CustomerOne_Loan nfo = CustomerTwo_Loan nfo = CustomerTwo_Loan nfo = CustomerOne_Loan	BankActions BankActions BankActions BankActions BankActions BankActions	e Create Environ = View Environ = Create Environ = View Environ = Create Environ = View Environ	ment = Any Value ment = Any Value ment = Any Value ment = Any Value ment = Any Value	Condition = Any Value Condition = Any Value Condition = Any Value Condition = Any Value Condition = Any Value	Permit Permit Permit Permit Permit	Verification TRUE TRUE TRUE FALSE TRUE	Result

Fig. 15. Merged Policy x Test Case 2

Right away we will notice our Verification Results contains a False which is known as Block Privilege. A rule or rules in our policy are incorrectly telling the system to not allow officer_1 to View CustomerTwo_Loan which is required for officer_1 to be able to Create LoanInfo resources. If we click on our Security Requirement (4), we can analyze deeper the reasoning for this result we have received. Here is where we will notice we have mistakenly selected the wrong privilege assignment for this subject's access on this resource.

Result(s) with selected ve	erification (Poli	cy Verification (July 3, 20:	8 16:16:57))		٤ ۲	8 rows out of 8			S	iearch	8	× I	þ
Requirement Schema		Subject	Resource	Action		Environment	Condition		Decision	Ver	ification R	lesult	Đ
Test Case 2	LoanOt	fficer = officer_1	LoanInfo = CustomerOne_Loan E	BankActions = Creat	e Env	/ironment = Any Valu	e Condition = Any V	alue	Permit		TRUE		1
Test Case 2	LoanOt	fficer = officer_1	LoanInfo = CustomerOne_Loan	BankActions = View	/ Env	vironment = Any Valu	e Condition = Any V	alue	Permit		TRUE		-li
Test Case 2	LoanOt	fficer = officer_1	LoanInfo = CustomerTwo_Loan E	BankActions = Creat	2.22	/ironment = Any Valu			Permit		TRUE		1
Test Case 2	LoanOt	fficer = officer_1	LoanInfo = CustomerTwo_Loan	BankActions = View					Permit		FALSE		
Test Case 2	LoanOt	fficer = officer_2	LoanInfo = CustomerOne_Loan E	BankActions = Creat	e Env	vironment = Any Valu			Permit		TRUE		1
Test Case 2	LoanOt	fficer = officer_2	LoanInfo = CustomerOne_Loan	BankActions = View		vironment = Any Valu			Permit		TRUE		1
Test Case 2	LoanOt	fficer = officer_2	Loaninfo = CustomerTwo_Loan E	BankActions = Creat		vironment = Any Valu			Permit		TRUE		
Test Case 2	LoanOt	fficer = officer 2	LoanInfo = CustomerTwo Loan	BankActions = View		vironment = Any Valu			Permit		TRUE		۰.
		Policy Name		Rule C	Combination A	Algorithm	Policy Enforcement	Algorithm		Merg	ged Result		t
Merred Policy : [J	ABAC : LoapOffi		er Policy, ABAC : BankTeller Policy/	Rule C		170		70.00			×	2	
			er Policy, ABAC : BankTeller Policy]	Rule C	Deny-overrid	170	Policy Enforcement Deny Biase	70.00	s		Deny	× -	
Rule(s) and Matching res		cer Policy, ABAC : FManag Policy against the selcted			Deny-overrid	les	Deny Biase	70.00	S	earch	Deny		•
Rule(s) and Matching res	sult of Selected Sequence No	cer Policy, ABAC : FManag Policy against the selcted Subject	security requirement:	A	Deny-overrid	66 rows out of 66 Environment	Deny Biase	d Decision	Inheritance	iearch : Relation	Deny Match	Result	•
Rule(s) and Matching res Policy Name BAC : LoanOfficer Policy	sult of Selected Sequence No 34	cer Policy, ABAC : FManag Policy against the selcted Subject LoanOfficer = officer_j	security requirement: Resource Personalinfo = CustomerTwo_Perso	A pnalinfo BankActi	Deny-overrid	66 rows out of 66 Environment Environment = Any	Deny Blase Condition Value Condition = Any Value	d Decision e Permit	Inheritance Origina	iearch Relation ated	Deny Match Not App	Result	
Rule(s) and Matching res Policy Name BAC : LoanOfficer Policy BAC : LoanOfficer Policy	sult of Selected Sequence No 34 35	cer Policy, ABAC ; FManag Policy against the selcted Subject LoanOfficer = officer_i LoanOfficer = officer_i	Resource Personalinfo = CustomerTwo_Perso Personalinfo = CustomerThree_Perso	A ponalinfo BankActi onalinfo BankAct	Deny-overrid	66 rows out of 66 Environment Environment = Any Environment = Any	Condition Value Condition = Any Value Value Condition = Any Value	d Decision e Permit e Permit	Inheritance Origina Origina	earch Relation ated ated	Match Not App	Result plicable plicable	
Rule(s) and Matching res Policy Name 3AC : LoanOfficer Policy 3AC : LoanOfficer Policy 3AC : LoanOfficer Policy	sult of Selected Sequence No 34 35 36	cer Policy, ABAC : FManag Policy against the selcted Subject LoanOfficer = officer LoanOfficer = officer	security requirement: Resource Personalinfo = CustomerTwo_Perso Personalinfo = CustomerThree_Personalinfo = CustomerThree_Persona	Analinfo BankActi onalinfo BankActi onalinfo BankActi	Deny-overrid	 66 rows out of 66 Environment Environment = Any Environment = Any Environment = Any 	Deny Blase Condition Value Condition = Any Valu	Decision e Permit e Permit e Permit	Inheritance Origina Origina Origina	earch Relation ated ated ated	Match Not App Not App Not App	Result Result Dicable Dicable	
Rule(s) and Matching res Policy Name BAC : LoanOfficer Policy BAC : LoanOfficer Policy BAC : LoanOfficer Policy BAC : LoanOfficer Policy	sult of Selected Sequence No 34 35 36 37	cer Policy, ABAC : FManag Policy against the selcted Subject LoanOfficer = officer LoanOfficer = officer LoanOfficer = officer LoanOfficer = officer	security requirement: Personalinfo = CustomerTwo_Perso Personalinfo = CustomerThree_Perss Personalinfo = CustomerThree_Perss Laaninfo = CustomerOne_Loa	A panalinfo BankActi onalinfo BankActi onalinfo BankActi in BankAct	Deny-overrid	 66 rows out of 66 Environment = Any Environment = Any Environment = Any Environment = Any 	Deny Bjase Condition Value Condition = Any Valu Value Condition = Any Valu Value Condition = Any Valu Value Condition = Any Valu	e Permit e Permit e Permit	Inheritance Origina Origina Origina Origina	earch Relation ated ated ated ated	Match Not App Not App Not App Not App	Result Result blicable blicable blicable	
Rule(s) and Matching res Policy Name BAC : LoanOfficer Policy BAC : LoanOfficer Policy BAC : LoanOfficer Policy BAC : LoanOfficer Policy BAC : LoanOfficer Policy	sult of Selected Sequence No 34 35 36 37 38	cer Policy, ABAC : FManay Policy against the selcted Subject LoanOfficer = officer LoanOfficer = officer LoanOfficer = officer LoanOfficer = officer	Resource Personalinfo = CustomerTwo_Perso Personalinfo = CustomerThree_Perso Personalinfo = CustomerThree_Perso Loaninfo = CustomerOne_Loa Loaninfo = CustomerOne_Loa	A Analinfo BankActi onalinfo BankActi onalinfo BankActi in BankActi in BankActi	Deny-overrid	66 rows out of 66 Environment Environment = Any	Condition Value Condition = Any Valu Value Condition = Any Valu Value Condition = Any Valu Value Condition = Any Valu Value Condition = Any Valu	Decision e Permit e Permit e Permit e Permit e Permit	Inheritance Origina Origina Origina Origina	earch Relation ated ated ated ated ated ated	Match Not App Not App Not App Not App Not App	Result Dicable Dicable Dicable Dicable	
Policy Name Policy Name AAC : LoanOfficer Policy BAC : LoanOfficer Policy	sult of Selected Sequence No 34 35 36 37 38 38 39	cer Policy, ABAC : FManag Policy against the selcted Subject LoanOfficer = officer LoanOfficer = officer LoanOfficer = officer LoanOfficer = officer LoanOfficer = officer LoanOfficer = officer	Resource Personalinfo = CustomerTwo_Perso Personalinfo = CustomerTree_Perso Personalinfo = CustomerTree_Perso Loaninfo = CustomerOne_Loa Loaninfo = CustomerTwo_Loa	onalinfo BankActi onalinfo BankActi in BankActi in BankActi in BankActi	Deny-overrid	es 66 rows out of 66 Environment = Any Environment = Any Environment = Any Environment = Any Environment = Any Environment = Any	Condition Value Condition = Any Valu	e Permit e Permit e Permit e Permit e Permit e Permit e Permit	Inheritance Origini Origini Origini Origini Origini	earch Relation ated ated ated ated ated ated ated ated	Match Not App Not App Not App Not App Not App Not App	Result Dicable Dicable Dicable Dicable	
Rule(s) and Matching res Policy Name BAC: LoanOfficer Policy BAC: LoanOfficer Policy BAC: LoanOfficer Policy BAC: LoanOfficer Policy BAC: LoanOfficer Policy BAC: LoanOfficer Policy	sult of Selected Sequence No 34 35 36 37 38 38 39 40	cer Policy, ABAC : FManay Policy against the selcted Subject LoanOfficer = officer LoanOfficer = officer LoanOfficer = officer LoanOfficer = officer LoanOfficer = officer LoanOfficer = officer	security requirement: Personalinfo = CustomerTwo_Person Personalinfo = CustomerThree_Person Personalinfo = CustomerThree_Person Loaninfo = CustomerTome_Loa Loaninfo = CustomerTwo_Loa Loaninfo = CustomerTwo_Loa	A onalinfo BankActi onalinfo BankActi in BankActi in BankActi in BankActi in BankActi	Deny-overrid	es 66 rows out of 66 Environment = Any Environment = Any Environment = Any Environment = Any Environment = Any Environment = Any	Deny Bjase Condition Value Condition = Any Valu Value Condition = Any Valu Value Condition = Any Valu Value Condition = Any Valu Value Condition = Any Valu	e Permit e Permit e Permit e Permit e Permit e Permit e Permit e Permit	Inheritance Origin Origin Origin Origin Origin	earch Relation ated ated ated ated ated ated ated ated	Match Not App Not App Not App Not App Not App Not App Not App	Result Dicable Dicable Dicable Dicable Dicable Dicable	
Rule(s) and Matching res Policy Name BAC : LoanOfficer Policy BAC : LoanOfficer Policy	sult of Selected Sequence No 34 35 36 37 38 39 40 41	cer Policy, ABAC : FManag Policy against the selcted Subject LoanOfficer = officer LoanOfficer = officer LoanOfficer = officer LoanOfficer = officer LoanOfficer = officer LoanOfficer = officer LoanOfficer = officer	Resource Personalinfo = CustomerTwo_Perso Personalinfo = CustomerThree_Perso Personalinfo = CustomerThree_Perso Loaninfo = CustomerOne_Loa Loaninfo = CustomerOne_Loa Loaninfo = CustomerTwo_Loa Loaninfo = CustomerTwo_Loa Personalinfo = CustomerTwo_Loa	A nalinfo Bankacti na Bankacti n Bankacti n Bankacti n Bankacti n Bankacti n Bankacti	Deny-overrid	es 66 rows out of 66 Environment = Any Environment = Any Environment = Any Environment = Any Environment = Any Environment = Any Environment = Any	Condition Value Condition = Any Valu Value Condition = Any Value Value Condition =	e Permit e Permit e Permit e Permit e Permit e Permit e Permit e Permit e Permit	Inheritance Origin Origin Origin Origin Origin Origin	earch Relation ated ated ated ated ated ated ated ated	Match Not App Not App Not App Not App De Not App Not App	Result Result blicable blicable blicable blicable blicable blicable blicable	
Rule(s) and Matching res Policy Name BAC : LoanOfficer Policy BAC : LoanOfficer Policy BAC : LoanOfficer Policy BAC : LoanOfficer Policy BAC : LoanOfficer Policy	sult of Selected Sequence No 34 35 36 37 38 38 39 40	cer Policy, ABAC : FManay Policy against the selcted Subject LoanOfficer = officer LoanOfficer = officer LoanOfficer = officer LoanOfficer = officer LoanOfficer = officer LoanOfficer = officer	Resource Personalinfo = CustomerTwo_Perso Personalinfo = CustomerTwo_Perso Personalinfo = CustomerThree_Perso Loaninfo = CustomerOne_Loa Loaninfo = CustomerOne_Loa Loaninfo = CustomerTwo_Loa Personalinfo = CustomerTwo_Loa Personalinfo = CustomerTwo_Loa Personalinfo = CustomerOne_Perso Personalinfo = CustomerOne_Perso	A ponalinfo Bankacti onalinfo Bankacti in Bankacti in Bankacti in Bankacti in Bankacti panlinfo Bankacti	Deny-overrid	es 66 rows out of 66 Environment = Any Environment = Any Environment = Any Environment = Any Environment = Any Environment = Any	Condition Value Condition = Any Valu Value Condition = Any Value Condit	e Permit e Permit	Inheritance Origin Origin Origin Origin Origin	earch Relation ated ated ated ated ated ated ated ated	Match Not App Not App Not App Not App Not App Not App Not App	Result Dicable Dicable Dicable Dicable Dicable Dicable Dicable	

Fig. 16. Merged Policy: Match Results

11 RESOLVING THIS ERROR - TEST CASE 2 (BLOCK PRIVILEGE)

To eliminate a Block Privilege error the policy author would need to go back and either update the related rule that is creating this error or delete it and define a Policy Enforcement Algorithm as Permit Biased because there will be no specific rule for this scenario. However, this could potentially create errors for other rules in our policy so we will modify the exact rule in this situation. For example, if we were to modify this rule below in the LoanOfficer Policy to Permit...

```
LoanOfficer Policy: Modify Rule (9):
(Rule No. = 9) \rightarrow (LoanOfficer = officer_1) \rightarrow (Action = View) \rightarrow (Resource = CustomerTwo_Loan) \rightarrow decision = Permit
```

Fig. 17. LoanOfficer Policy: Modified Rule (9)

LoanOfficer = officer_1 LoanInfo = CustomerTwo_Loan

Then retest using the same Policy Verification selections as last time we will now get a True Verification result showing that we no longer have a Block Privilege error occuring.

BankActions = View

olicy Verification (July 5, 20	018 11:54:51)(s) Summa	ry			1 rows out of 1				Search	×
tatus	Name	Verification Type	Verification Technique	Number of Policy(s)	Combination Algorithm	Enforcement Algorithm		Policy List		
ToDate Policy Verification	(July 5, 2018 11:54:51)	Standard	Merged Policy	3	Deny-overrides	Deny Biased	ABAC:BankTeller Policy	, ABAC:LoanOfficer	r Policy, ABAC:FMana	ager Po
Jacult(r) with calacted varif	ication (Rolicy Verificatio	n (July 5 2019 11-	54.51))		S rov	vs out of 8			Search	X
Result(s) with selected verif		on (July 5, 2018 11:						-	Search	-
Requirement Schema	Subject		Resource	Action	E	nvironment	Condition	Decision	Verification F	_
Requirement Schema Test Case 2	Subject LoanOfficer = office	er_1 Loanii	Resource nfo = CustomerOne_Loan	BankActions =	Create Environ		Condition Condition = Any Value	Decision Permit		_
Requirement Schema	Subject	er_1 Loanii	Resource	BankActions =	Create Environ	nvironment		Decision	Verification F	_
Requirement Schema Test Case 2	Subject LoanOfficer = office	er_1 Loanin er_1 Loanin	Resource nfo = CustomerOne_Loan	BankActions = BankActions	Create Environ = View Environ	nvironment iment = Any Value	Condition = Any Value	Decision Permit	Verification F	_
Requirement Schema Test Case 2 Test Case 2	Subject LoanOfficer = office LoanOfficer = office	er_1 Loanin er_1 Loanin er_1 Loanin	Resource nfo = CustomerOne_Loan nfo = CustomerOne_Loan	BankActions = BankActions BankActions =	Create Environ = View Environ = Create Environ	nvironment iment = Any Value iment = Any Value	Condition = Any Value Condition = Any Value	Decision Permit Permit	Verification F TRUE TRUE	
Requirement Schema Test Case 2 Test Case 2 Test Case 2 Test Case 2	Subject LoanOfficer = office LoanOfficer = office LoanOfficer = office	er_1 Loanii er_1 Loanii er_1 Loanii er_1 Loanii	Resource nfo = CustomerOne_Loan nfo = CustomerOne_Loan nfo = CustomerTwo_Loan	BankActions = BankActions BankActions = BankActions	Create Environ = View Environ = Create Environ = View Environ = View Environ	nvironment Iment = Any Value Iment = Any Value Iment = Any Value Iment = Any Value	Condition = Any Value Condition = Any Value Condition = Any Value	Decision Permit Permit Permit	Verification F TRUE TRUE TRUE	_
Requirement Schema Test Case 2 Test Case 2 Test Case 2 Test Case 2 Test Case 2	Subject LoanOfficer = office LoanOfficer = office LoanOfficer = office LoanOfficer = office	er_1 Loanii rr_1 Loanii rr_1 Loanii rr_1 Loanii rr_2 Loanii	Resource nfo = CustomerOne_Loan nfo = CustomerOne_Loan nfo = CustomerTwo_Loan nfo = CustomerTwo_Loan	BankActions = BankActions BankActions = BankActions BankActions =	Create Environ View Environ Create Environ View Environ View Environ View Environ Create Environ	nvironment iment = Any Value	Condition = Any Value Condition = Any Value Condition = Any Value Condition = Any Value	Decision Permit Permit Permit Permit	Verification R TRUE TRUE TRUE TRUE TRUE	_
Requirement Schema Test Case 2 Test Case 2 Test Case 2 Test Case 2 Test Case 2 Test Case 2	Subject LoanOfficer = office LoanOfficer = office LoanOfficer = office LoanOfficer = office LoanOfficer = office	er_1 Loanin er_1 Loanin er_1 Loanin er_1 Loanin er_2 Loanin er_2 Loanin	Resource nfo = CustomerOne_Loan nfo = CustomerOne_Loan nfo = CustomerTwo_Loan nfo = CustomerTwo_Loan nfo = CustomerOne_Loan	BankActions = BankActions BankActions = BankActions BankActions = BankActions	Créate Environ View Environ Create Environ View Environ Create Environ Create Environ View Environ	nvironment iment = Any Value	Condition = Any Value Condition = Any Value Condition = Any Value Condition = Any Value Condition = Any Value	Decision Permit Permit Permit Permit Permit	Verification R TRUE TRUE TRUE TRUE TRUE TRUE	_

Fig. 18. Updated Policy: No Longer Blocking

12 SETTING UP THE POLICIES – TEST CASE 3 (INHERITANCE LOOP)

This bank example contains three policies (BankTeller Policy & LoanOfficer Policy & FManager Policy). The attributes in this example have not been changed from previous Test Case 1 and Test Case 2. The Attribute/Attribute Values included in these policies are as shown in Figure 19.

Originated

Permit

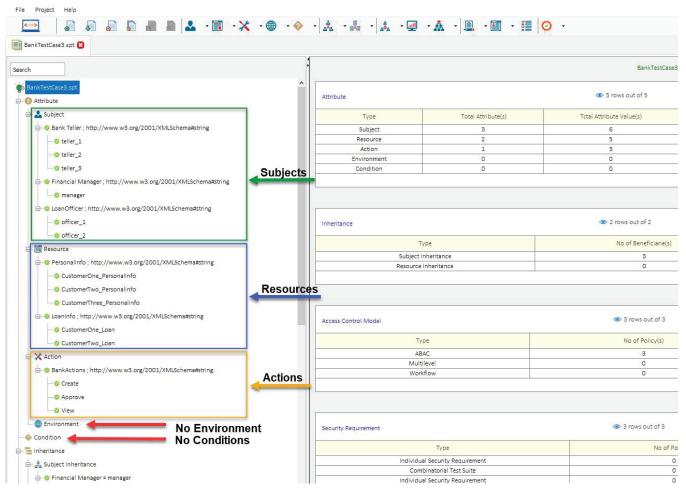


Fig. 19. Test Case 3

13 SUBJECT INHERITANCE – TEST CASE 3 (INHERITANCE LOOP)

Depending on your security needs or organizational structure you may decide to define Inheritance relationships to help you generate policy Rules more quickly. For this bank policy example we will define (3) Subject Inheritance Relationships. If you are creating your own Blank Project enter the relationship rules below. If you did not create your own Project File, you can simply open Security Policy Tool – Project File: BankTestCase3 and these relationships will have been already created for you.

```
Subject Inheritance:

Beneficiary Values \rightarrow Financial Manager = manager

Inherited Values \rightarrow Loan Officer = officer_1 & officer_2

Beneficiary Values \rightarrow Loan Officer = officer_1

Inherited Values \rightarrow Bank Teller = teller_1 & teller_2 & teller_3

Beneficiary Values \rightarrow Loan Officer = officer_2

Inherited Values \rightarrow Bank Teller = teller_1 & teller_2 & teller_3
```

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

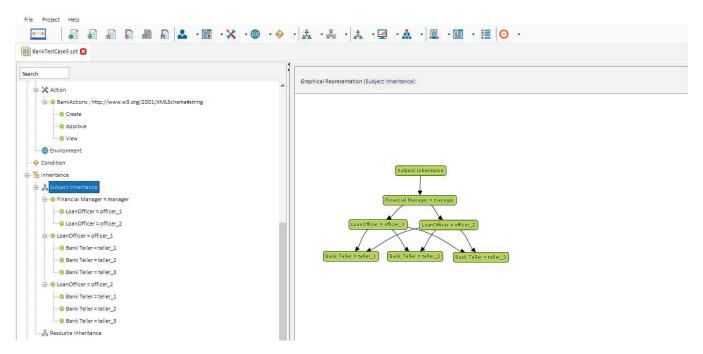


Fig. 20. Subject Inheritance Relationships

By defining these relationships, any (Originated) Rules with Decision = Permit given to Bank Teller subject attributes will now also be given to officer_1 & officer_2 as (Inherited) Rules in our policies. Additionally any Rules with Decision = Permit given to Loan Officers subject attributes will now also be given to the manager subject attribute. 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., bank teller/loan officer 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.

14 UNDERSTANDING THIS ERROR - TEST CASE 3 (INHERITANCE LOOP)

Unlike all other Test Case examples, to demonstrate this error we do not need to run any policy tests. Thus we do not have sections for modeling our policies, creating security requirements, and running verification tests for this example. An Inheritance Loop is an error that occurs when the policy author defines inheritance relations that gives a subject both recursive and subsequent inheritance.

For example, an inheritance loop or sometimes called "'cyclical inheritance" could look like this..

Person A can inherit \rightarrow Person B Privileges Person B can inherit \rightarrow Person C Privileges Person C can inherit \rightarrow Person A Privileges

In our example an Inheritance Loop would look like the screenshot below...

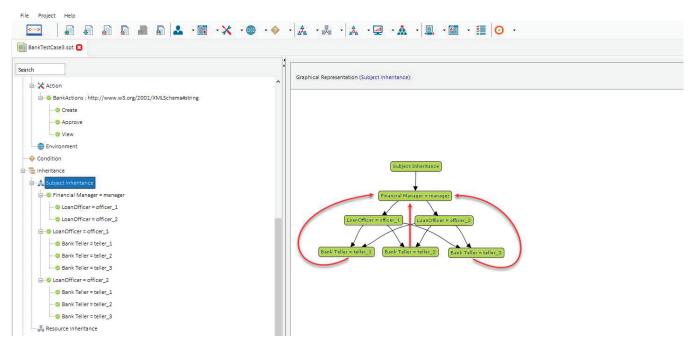


Fig. 21. Inheritance Loop

However, in Security Policy Tool it is not possible to create this type of error. Security Policy Tool automatically detects which attribute values are being selected as Beneficiaries/Inherited Values to prevent Inheritance Loops. It will not allow attributes that are already allocated as Inherited Values to a Beneficiary Value to then also be allocated as a Beneficiary Value to the Beneficiary Value they are already giving their (inheritance) rules to.

See the screenshot below, values that would create an Inheritance Loop are unavailable to be selected...

BankActions ; http://www.w3.org/2001/XMLSchema#string Oreate Oreate Opprove	Graphical Representation (Subject Inheritance):
O View	
Environment	
Condition	
Inheritance	
📩 Subject Inheritance	Subject Inheritance
Financial Manager = manager	
	🖌 📩 Add Subject Inheritance Inherited Value 🛛 🗙
LoanOfficer = officer_2	Subject Inheritance
- O LoanOfficer = officer_1	LoanOfficer = o Beneficiary : Bank Teller = teller_1
🗿 Bank Teller = teller_1	Inherited Value :
Bank Teller = teller_2	Bank Teller = teller 2
Bank Teller = teller_3	Bank Teller_3
□ O LoanOfficer = officer_2	Bank Teller = teller_1
O Bank Teller = teller_1	Add Cancel
Bank Teller = teller_2	

Fig. 22. Subject Inheritance: Cannot Add Any Value

In our example, allowing Bank Teller = teller_1, teller_2, or teller_3 to be able to be a Beneficiary for LoanOfficer = officer_1 or officer_2 or Manager = manager would create an Inheritance Loop. Security Policy Tool detects this issue and only allows Bank Tellers to be able to be defined as a Beneficiary to other Bank Tellers since they are not Inheriting Values from each other at this moment.

If we continue with our example and make teller_1 the beneficiary to teller_2 and teller_3 and attempt to define another Inherited Value for teller_1 we will get this message below and avoid creating an Inheritance Loop. See screenshot...

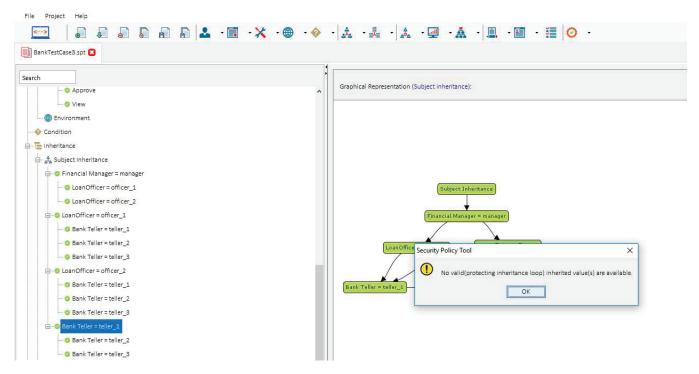


Fig. 23. Subject Inheritance: Loop Error Prevented

15 SETTING UP THE POLICIES – TEST CASE 4 (SEPARATION OF DUTY)

This bank example contains three policies (BankTeller Policy & LoanOfficer Policy & FManager Policy). The attributes in this example have not been changed from previous Bank Test Cases. The Attribute/Attribute Values included in these policies are as shown in Figure 24.

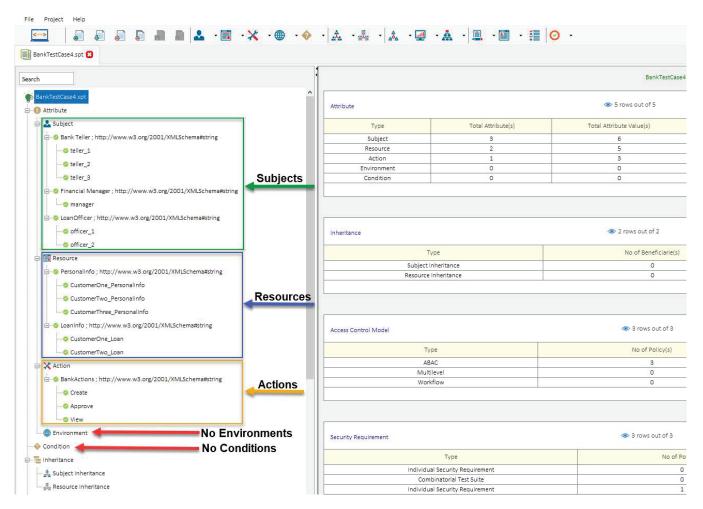


Fig. 24. Test Case 4

16 MODELING YOUR POLICY – TEST CASE 4 (SEPERATION OF DUTY)

Now that we have entered our attributes we can model our three policies (BankTeller Policy & LoanOfficer Policy & FManager 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:

BankTeller Policy:

Buikrener roney.
(Bank Teller = teller_1, CustomerOne_PersonalInfo, Create) \rightarrow Permit
(Bank Teller = teller_1, CustomerOne_PersonalInfo, View) \rightarrow Permit
(Bank Teller = teller_1, CustomerTwo_PersonalInfo, Create) \rightarrow Permit
(Bank Teller = teller_1, CustomerTwo_PersonalInfo, View) \rightarrow Permit
(Bank Teller = teller_1, CustomerThree_PersonalInfo, Create) \rightarrow Permit
(Bank Teller = teller_1, CustomerThree_PersonalInfo, View) \rightarrow Permit
(Bank Teller = teller_2, CustomerOne_PersonalInfo, Create) \rightarrow Permit
(Bank Teller = teller_2, CustomerOne_PersonalInfo, View) \rightarrow Permit
(Bank Teller = teller_2, CustomerTwo_PersonalInfo, Create) \rightarrow Permit
(Bank Teller = teller_2, CustomerTwo_PersonalInfo, View) \rightarrow Permit
(Bank Teller = teller_2, CustomerThree_PersonalInfo, Create) \rightarrow Permit
(Bank Teller = teller_2, CustomerThree_PersonalInfo, View) \rightarrow Permit
(Bank Teller = teller_3, CustomerOne_PersonalInfo, Create) \rightarrow Permit
(Bank Teller = teller_3, CustomerOne_PersonalInfo, View) \rightarrow Permit
(Bank Teller = teller_3, CustomerTwo_PersonalInfo, Create) \rightarrow Permit
(Bank Teller = teller_3, CustomerTwo_PersonalInfo, View) \rightarrow Permit
(Bank Teller = teller_3, CustomerThree_PersonalInfo, Create) \rightarrow Permit
(Bank Teller = teller_3, CustomerThree_PersonalInfo, View) \rightarrow Permit

(Bank Teller = teller_1, CustomerOne_Loan, View) →Permit (Bank Teller = teller_1, CustomerTwo_Loan, View) →Permit (Bank Teller = teller_2, CustomerOne_Loan, View) →Permit (Bank Teller = teller_3, CustomerTwo_Loan, View) →Permit (Bank Teller = teller_3, CustomerTwo_Loan, View) →Permit (Bank Teller = teller_1, CustomerOne_Loan, Approve) →Deny (Bank Teller = teller_1, CustomerTwo_Loan, Approve) →Deny (Bank Teller = teller_2, CustomerOne_Loan, Approve) →Deny (Bank Teller = teller_2, CustomerOne_Loan, Approve) →Deny (Bank Teller = teller_2, CustomerTwo_Loan, Approve) →Deny (Bank Teller = teller_3, CustomerOne_Loan, Approve) →Deny (Bank Teller = teller_3, CustomerTwo_Loan, Approve) →Deny (Bank Teller = teller_3, CustomerTwo_Loan, Approve) →Deny

LoanOfficer Policy:

FManager Policy:

(Manager = manager, CustomerOne_PersonalInfo, Create) \rightarrow Permit
(Manager = manager, CustomerOne_PersonalInfo, View) \rightarrow Permit
(Manager = manager, CustomerTwo_PersonalInfo, Create) →Permit
(Manager = manager, CustomerTwo_PersonalInfo, View) →Permit
(Manager = manager, CustomerThree_PersonalInfo, Create) \rightarrow Permit
(Manager = manager, CustomerThree_PersonalInfo, View) \rightarrow Permit
(Manager = manager, CustomerOne_Loan, Create) \rightarrow Permit
(Manager = manager, CustomerOne_Loan, View) \rightarrow Permit
(Manager = manager, CustomerTwo_Loan, Create) \rightarrow Permit
(Manager = manager, CustomerTwo_Loan, View) \rightarrow Permit
(Manager = manager, CustomerOne_Loan, Approve) →Permit
(Manager = manager, CustomerTwo_Loan, Approve) →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: BankTestCase4 and these policies will have been already created for you.

WWW.SECURITYPOLICYTOOL.COM

BankTeller Policy P	olicy(s) Summary		I rows out of 1				Search	×
Model	Policy Name	Rule Combination Algorithm	Policy Enforcement Algorithm	No. of Rule(s)	Time Created		Last Modified	
ABAC	BankTeller Policy	Deny-overrides	Deny Biased	30	July 3, 2018 14:28:3	2	July 3, 2018 14:28:3	2
Rule (s) defined wit	th selected policy (BankTeller I	Policy):	30 rows	out of 30			Search	×
Sequence No	Subject	Resource	Action	Environment	Condition	Decision	Inheritance Rela	atio
1	Bank Teller = teller 1	Personalinfo = CustomerOne Personalinfo	BankActions = Create	Environment = Any Value	Condition = Any Value	Permit	Originated	-
2	Bank Teller = teller 1	Personalinfo = CustomerOne Personalinfo	BankActions = View	Environment = Any Value	Condition = Any Value	Permit	Originated	
3	Bank Teller = teller 1	Personalinfo = CustomerTwo Personalinfo	BankActions = Create	Environment = Any Value	Condition = Any Value	Permit	Originated	-
4	Bank Teller = teller 1	Personalinfo = CustomerTwo Personalinfo	BankActions = View	Environment = Any Value	Condition = Any Value	Permit	Originated	_
5	Bank Teller = teller 1	Personalinfo = CustomerThree Personalinfo	BankActions = Create	Environment = Any Value	Condition = Any Value	Permit	Originated	_
6	Bank Teller = teller 1	Personalinfo = CustomerThree Personalinfo	BankActions = View	Environment = Any Value	Condition = Any Value	Permit	Originated	_
7	Bank Teller = teller 2	Personalinfo = CustomerOne_Personalinfo	BankActions = Create	Environment = Any Value	Condition = Any Value	Permit	Originated	_
8	Bank Teller = teller 2	Personalinfo = CustomerOne Personalinfo	BankActions = View	Environment = Any Value	Condition = Any Value	Permit	Originated	_
9	Bank Teller = teller_2	Personalinfo = CustomerTwo Personalinfo	BankActions = Create	Environment = Any Value	Condition = Any Value	Permit	Originated	1
10	Bank Teller = teller 2	Personalinfo = CustomerTwo Personalinfo	BankActions = View	Environment = Any Value	Condition = Any Value	Permit	Originated	-
11	Bank Teller = teller 2	Personalinfo = CustomerThree Personalinfo	BankActions = Create	Environment = Any Value	Condition = Any Value	Permit	Originated	-
12	Bank Teller = teller_2	Personalinfo = CustomerThree_Personalinfo	BankActions = View	Environment = Any Value	Condition = Any Value	Permit	Originated	8
13	Bank Teller = teller_3	Personalinfo = CustomerOne_Personalinfo	BankActions = Create	Environment = Any Value	Condition = Any Value	Permit	Originated	8
14	Bank Teller = teller_3	Personalinfo = CustomerOne_Personalinfo	BankActions = View	Environment = Any Value	Condition = Any Value	Permit	Originated	
15	Bank Teller = teller_3	Personalinfo = CustomerTwo_Personalinfo	BankActions = Create	Environment = Anv Value	Condition = Any Value	Permit	Originated	8
16	Bank Teller = teller_3	Personalinfo = CustomerTwo_Personalinfo	BankActions = View	Environment = Anv Value	Condition = Any Value	Permit	Originated	1
17	Bank Teller = teller_3	Personalinfo = CustomerThree_Personalinfo	BankActions = Create	Environment = Any Value	Condition = Any Value	Permit	Originated	T
18	Bank Teller = teller_3	PersonalInfo = CustomerThree_PersonalInfo	BankActions = View	Environment = Any Value	Condition = Any Value	Permit	Originated	8
19	Bank Teller = teller_1	LoanInfo = CustomerOne_Loan	BankActions = View	Environment = Any Value	Condition = Any Value	Permit	Originated	-
20	Bank Teller = teller_1	Loaninfo = CustomerTwo_Loan	BankActions = View	Environment = Anv Value	Condition = Any Value	Permit	Originated	8
21	Bank Teller = teller_2	Loaninfo = CustomerOne_Loan	BankActions = View	Environment = Any Value	Condition = Any Value	Permit	Originated	5
22	Bank Teller = teller_2	Loaninfo = CustomerTwo_Loan	BankActions = View	Environment = Any Value	Condition = Any Value	Permit	Originated	
23	Bank Teller = teller_3	Loaninfo = CustomerOne_Loan	BankActions = View	Environment = Any Value	Condition = Any Value	Permit	Originated	
24	Bank Teller = teller_3	LoanInfo = CustomerTwo_Loan	BankActions = View	Environment = Any Value	Condition = Any Value	Permit	Originated	5
25	Bank Teller = teller_1	Loaninfo = CustomerOne_Loan	BankActions = Approve	Environment = Any Value	Condition = Any Value	Deny	Originated	8
26	Bank Teller = teller_1	LoanInfo = CustomerTwo_Loan	BankActions = Approve	Environment = Any Value	Condition = Any Value	Deny	Originated	8
27	Bank Teller = teller_2	LoanInfo = CustomerOne_Loan	BankActions = Approve	Environment = Any Value	Condition = Any Value	Deny	Originated	6
28	Bank Teller = teller_2	LoanInfo = CustomerTwo_Loan	BankActions = Approve	Environment = Any Value	Condition = Any Value	Deny	Originated	
29	Bank Teller = teller_3	LoanInfo = CustomerOne_Loan	BankActions = Approve	Environment = Any Value	Condition = Any Value	Deny	Originated	1
30	Bank Teller = teller 3	Loaninfo = CustomerTwo Loan	BankActions = Approve	Environment = Any Value	Condition = Any Value	Denv	Originated	_

Fig. 25. BankTeller Policy

oanOfficer 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	LoanOfficer Policy	Deny-overrides	Permit Biased	-23	July 3, 2018 14:40:4	16	July 3, 2018 14:40:	46
ule (s) defined wit	th selected policy (LoanOfficer F	Policy):	@ 23 rows	out of 23			Search	×
Sequence No	Subject	Resource	Action	Environment	Condition	Decision	Inheritance Re	atio
1	LoanOfficer = officer_1	Personalinfo = CustomerOne_Personalinfo	BankActions = View	Environment = Any Value	Condition = Any Value	Permit	Originated	Н
2	LoanOfficer = officer_1	Personalinfo = CustomerOne_Personalinfo	BankActions = Create	Environment = Any Value	Condition = Any Value	Permit	Originated	Ŀ
3	LoanOfficer = officer_1	Personalinfo = CustomerTwo_Personalinfo	BankActions = View	Environment = Any Value	Condition = Any Value	Permit	Originated	Ł
4	LoanOfficer = officer_1	Personalinfo = CustomerTwo_Personalinfo	BankActions = Create	Environment = Any Value	Condition = Any Value	Permit	Originated	d l
5	LoanOfficer = officer_1	Personalinfo = CustomerThree_Personalinfo	BankActions = View	Environment = Any Value	Condition = Any Value	Permit	Originated	b
6	LoanOfficer = officer_1	Personalinfo = CustomerThree_Personalinfo	BankActions = Create	Environment = Any Value	Condition = Any Value	Permit	Originated	ł
7	LoanOfficer = officer_1	Loaninfo = CustomerOne_Loan	BankActions = View	Environment = Any Value	Condition = Any Value	Permit	Originated	d
8	LoanOfficer = officer_1	Loaninfo = CustomerOne_Loan	BankActions = Create	Environment = Anv Value	Condition = Any Value	Permit	Originated	Ь
9	LoanOfficer = officer_1	Loaninfo = CustomerTwo_Loan	BankActions = View	Environment = Anv Value	Condition = Any Value	Permit	Originated	ł
10	LoanOfficer = officer_1	LoanInfo = CustomerTwo_Loan	BankActions = Create	Environment = Anv Value	Condition = Any Value	Permit	Originated	Ь
11	LoanOfficer = officer_2	Personalinfo = CustomerOne_Personalinfo	BankActions = View	Environment = Anv Value	Condition = Any Value	Permit	Originated	Ь
12	LoanOfficer = officer_2	Personalinfo = CustomerOne_Personalinfo	BankActions = Create	Environment = Any Value	Condition = Any Value	Permit	Originated	Ь
13	LoanOfficer = officer_2	Personalinfo = CustomerTwo_Personalinfo	BankActions = View	Environment = Any Value	Condition = Any Value	Permit	Originated	Ь
14	LoanOfficer = officer_2	Personalinfo = CustomerTwo_Personalinfo	BankActions = Create	Environment = Any Value	Condition = Any Value	Permit	Originated	Ŀ
15	LoanOfficer = officer_2	Personalinfo = CustomerThree_Personalinfo	BankActions = View	Environment = Any Value	Condition = Any Value	Permit	Originated	d
16	LoanOfficer = officer_2	Personalinfo = CustomerThree_Personalinfo	BankActions = Create	Environment = Any Value	Condition = Any Value	Permit	Originated	Ь
17	LoanOfficer = officer_2	Loaninfo = CustomerOne_Loan	BankActions = View	Environment = Any Value	Condition = Any Value	Permit	Originated	Ł
18	LoanOfficer = officer_2	Loaninfo = CustomerOne_Loan	BankActions = Create	Environment = Any Value	Condition = Any Value	Permit	Originated	b
19	LoanOfficer = officer_2	LoanInfo = CustomerTwo_Loan	BankActions = View	Environment = Anv Value	Condition = Any Value	Permit	Originated	Ь
20	LoanOfficer = officer_2	Loaninfo = CustomerTwo_Loan	BankActions = Create	Environment = Any Value	Condition = Any Value	Permit	Originated	4
21	LoanOfficer = officer_1	Loaninfo = CustomerTwo_Loan	BankActions = Approve	Environment = Any Value	Condition = Any Value	Deny	Originated	Ь
22	LoanOfficer = officer_2	Loaninfo = CustomerOne_Loan	BankActions = Approve	Environment = Any Value	Condition = Any Value	Deny	Originated	ł
23	LoanOfficer = officer_2	Loaninfo = CustomerTwo_Loan	BankActions = Approve	Environment = Any Value	Condition = Any Value	Denv	Originated	d

Fig. 26. LoanOfficer Policy

FManager Policy P	olicy(s) Summary		1 rows out of 1			3	Search
Model	Policy Name	Rule Combination Algorithm F	Policy Enforcement Algorithm	No. of Rule(s)	Time Created		Last Modified
ABAC	FManager Policy	Permit-overrides	Permit Biased	10	July 3, 2018 15:03:03		July 3, 2018 15:03:03
२ule (s) defined wi	ith selected policy (FManager Policy)	1	10 rows out	ut of 10		3	Search
Sequence No	Subject	Resource	Action	Environment	Condition	Decision	Inheritance Relation
1	Financial Manager = manager	Personalinfo = CustomerOne_Personalinfo	BankActions = Create	Environment = Any Value	Condition = Any Value	Permit	Originated
2	Financial Manager = manager	PersonalInfo = CustomerOne_PersonalInfo	BankActions = View	Environment = Any Value	Condition = Any Value	Permit	Originated
3	Financial Manager = manager	Personalinfo = CustomerTwo_Personalinfo	BankActions = Create	Environment = Any Value	Condition = Any Value	Permit	Originated
4	Financial Manager = manager	Personalinfo = CustomerTwo_Personalinfo	BankActions = View	Environment = Any Value	Condition = Any Value	Permit	Originated
5	Financial Manager = manager	Personalinfo = CustomerThree_Personalinfo	BankActions = Create	Environment = Any Value	Condition = Any Value	Permit	Originated
6	Financial Manager = manager	Personalinfo = CustomerThree_Personalinfo	BankActions = View	Environment = Any Value	Condition = Any Value	Permit	Originated
7	Financial Manager = manager	Loaninfo = CustomerOne_Loan	BankActions = View	Environment = Any Value	Condition = Any Value	Permit	Originated
8	Financial Manager = manager	LoanInfo = CustomerTwo_Loan	BankActions = View	Environment = Any Value	Condition = Any Value	Permit	Originated
	and the second sec	Loaninfo = CustomerOne Loan	BankActions = Approve	Environment = Any Value	Condition = Any Value	Permit	Originated
9	Financial Manager = manager	Loannio - Costonicione_Loan	burno tectorio "ripprove	Environment – Any value	Condition - Any value		

Fig. 27. FManager Policy

17 SOD SECURITY REQUIREMENTS - TEST CASE 4 (SEPARATION OF DUTY)

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

SOD Security Requirements:

(Loan Officer = officer_1) & (Action = Approve) & (LoanInfo = CustomerOne_Loan) \rightarrow decision = Permit

 $(Loan Officer = officer_1) \& (Action = Create) \& (LoanInfo = CustomerOne_Loan) \rightarrow decision = Permited (LoanInfo = CustomerOne_Loan) \\ \rightarrow decision = Permited$

After entering the rules above your SOD 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: BankTestCase4 and these requirements will have been already created for you.

OD 1(s) Summary		. <	I rows out of 1		Search	
ž	Access Control Security Requirement		Requirement Schema		No. of Security Requirement(s)	
	SOD		SOD 1		2	
ecurity Requirement (s)) defined under selected Requirement	Schema (SOD 1):	2 rov	ws out of 2	Search	XI 4
ecurity Requirement (s)) defined under selected Requirement Subject	Schema (SOD 1): Resource	⊛ 2 rov Action	ws out of 2	Search	Decision.
			22.02	See an office		Decision Permit

Fig. 28. SOD Security Requirements

18 POLICY VERIFICATION/ANALYZING RESULTS - TEST CASE 4 (SEPARATION OF DUTY)

Now that we are ready to test our policies let's discuss the error we will be looking for in this fourth example. When policies are designed there is potential for a "Separation of Duty" error being created. A Separation of Duty error occurs when two or more rules cause competing interests among subjects, resources, or actions. For example, giving one subject too much privileges to the point that they could misuse the system.

For example, when the policy author was designing the logic for these bank policies; the author intended all LoanOfficers to be able to Create all LoanInfo resources. However, they should not be able to Approve loans as way to ensure Loan Officers do not misuse their privileges to create

improper loans. To ensure the policy has been designed correctly let's verify that this is false (e.g., LoanOfficer \rightarrow Approve \rightarrow LoanInfo Resources).

We will run one "Single Policy" Separation of Duty Verification with our LoanOfficer Policy to reveal the Separation of Duty error that is present in our policies. To do this, we will select SOD 1 (security requirement) and LoanOfficer Policy and analyze our verification results. Again, this will have already been done for you if you open Project File: BankTestCase4.

OD Verification(Ju	ly 3, 2018 18:22:35)(s) Summary			1 rows out of 1			Search	×
Status	Name	Verification Type	Verification Technique	Number of Policy(s)	Combination Algorithm	Enforcement Algorithm	Poli	cy List
UpToDate	SOD Verification(July 3, 2018 18:22:35)	SOD	Single Policy	1	Deny-overrides	Deny Biased	ABAC:Loan	Officer Policy
Result(s) with selec	cted verification (SOD Verification(July 3, 2	018 18:22:35))		I rows out	it of 1		Search	×
	SOD Nan	ne			SC	DD Result		
	SOD : SOI	01			All ca	n be granted		
Result(s) with selee	cted SOD		⊗ 2 rov	vs out of 2			Search	×
	nema Subject	Resource	Actio	n Envi	ronment	Condition Decisi	on Verific	ation Result
Requirement Scl			new Construction of the second s	NATION AND AND A COMPANY AND A C	74 10 10 10 10 10 10 10 10			Children Ballows
SOD 1	LoanOfficer = officer_1	Loaninfo = CustomerOne_Lo	an BankActions =	Approve Environme	nt = Any Value Cond	tion = Any Value Perm	it	TRUE

Fig. 29. LoanOfficer Policy x SOD 1

Right away we will notice our test came back with two True Results which is not what was intended. LoanOfficers should be able to Create loans but should not be able to Approve loans. A rule or rules in our policy are incorrectly telling the system to allow officer_1 to Approve CustomerTwo_Loan. If we click on our Security Requirement related to Approving the resource, we can analyze deeper the reasoning for this result we have received.

WWW.SECURITYPOLICYTOOL.COM

Result(s) with se	elected SOD		2 rows o	ut of 2			Sear	:h	×
Requirement	Schema Sub	ject Resource	Action	Environm	ent	Condition	Decision	Verification	n Result
SOD 1	LoanOfficer	= officer 1 LoanInfo = CustomerOne Loa	n BankActions = Ap	prove Environment =	Any Value Cond	ition = Any Value	Permit	TRU	F
SOD 1	LoanOfficer	= officer 1 LoanInfo = CustomerOne Loa	n BankActions = Ci			ition = Any Value	Permit	TRU	-
Policy(s) and M	atching result against the se	Inted security requirement:		1 rows out of 1		CARLOUTER	Sear	-h.	
ency(of and in	arean Brezer against the se						Jean	.0.	
	Policy Name	Rule Combination	Algorithm	Policy Enfo	rcement Algorithm		Combine	d Result	
, A	ABAC : LoanOfficer Policy	Deny-over	rides	Per	mit Biased		Pen	nit	_
Rule(s) and Mat	ching result of Selected Poli	icy against the selcted security requirement:		23 rows or	ut of 23		Sear	:h	
equence No	Subject	Resource	Action	Environment	Condition	Decision	Inheritance Relation	Match	n Result
1	LoanOfficer = officer_1	Personalinfo = CustomerOne_Personalinfo	BankActions = View	Environment = Any Value	Condition = Any Valu	e Permit	Originated	Not Ap	plicable
2	LoanOfficer = officer_1	Personalinfo = CustomerOne_Personalinfo	BankActions = Create	Environment = Any Value	Condition = Any Valu	Permit	Originated	Not Ap	plicable
3	LoanOfficer = officer_1	Personalinfo = CustomerTwo_Personalinfo	BankActions = View	Environment = Any Value	Condition = Any Valu	Permit	Originated	Not Ap	plicable
4	LoanOfficer = officer_1	Personalinfo = CustomerTwo_Personalinfo	BankActions = Create	Environment = Any Value	Condition = Any Valu	Je Permit	Originated	Not Ap	plicable
5	LoanOfficer = officer_1	Personalinfo = CustomerThree_Personalinfo	BankActions = View	Environment = Any Value	Condition = Any Valu	Permit	Originated	Not Ap	plicable
6	LoanOfficer = officer_1	Personalinfo = CustomerThree_Personalinfo	BankActions = Create	Environment = Any Value	Condition = Any Valu	Permit	Originated	Not Ap	plicable
7	LoanOfficer = officer_1	Loaninfo = CustomerOne_Loan	BankActions = View	Environment = Any Value	Condition = Any Valu	Permit	Originated	Not Ap	plicable
8	LoanOfficer = officer_1	Loaninfo = CustomerOne_Loan	BankActions = Create	Environment = Any Value	Condition = Any Valu	le Permit	Originated	Not Ap	plicable
9	LoanOfficer = officer_1	LoanInfo = CustomerTwo_Loan	BankActions = View	Environment = Any Value	Condition = Any Valu	e Permit	Originated	Not Ap	plicable
10	LoanOfficer = officer_1	Loaninfo = CustomerTwo_Loan	BankActions = Create	Environment = Any Value	Condition = Any Valu	Permit	Originated	Not Ap	plicable
11	LoanOfficer = officer_2	Personalinfo = CustomerOne_Personalinfo	BankActions = View	Environment = Any Value	Condition = Any Valu	Permit	Originated	Not Ap	plicable
12	LoanOfficer = officer_2	Personalinfo = CustomerOne_Personalinfo	BankActions = Create	Environment = Any Value	Condition = Any Valu	Je Permit	Originated	Not Ap	plicable
13	LoanOfficer = officer_2	Personalinfo = CustomerTwo_Personalinfo	BankActions = View	Environment = Any Value	Condition = Any Valu	Permit	Originated	Not Ap	plicable
14	LoanOfficer = officer_2	Personalinfo = CustomerTwo_Personalinfo	BankActions = Create	Environment = Any Value	Condition = Any Valu	le Permit	Originated	Not Ap	plicable
15	LoanOfficer = officer_2	Personalinfo = CustomerThree_Personalinfo	BankActions = View	Environment = Any Value	Condition = Any Valu	Permit	Originated	Not App	plicable
16	LoanOfficer = officer_2	Personalinfo = CustomerThree_Personalinfo	BankActions = Create	Environment = Any Value	Condition = Any Valu	Permit	Originated	Not Ap	plicable
17	LoanOfficer = officer_2	LoanInfo = CustomerOne_Loan	BankActions = View	Environment = Any Value	Condition = Any Valu	Je Permit	Originated	Not Ap	plicable
18	LoanOfficer = officer_2	Loaninfo = CustomerOne_Loan	BankActions = Create	Environment = Any Value	Condition = Any Valu	Permit	Originated	Not Ap	plicable
19	LoanOfficer = officer_2	LoanInfo = CustomerTwo_Loan	BankActions = View	Environment = Any Value	Condition = Any Valu	Permit	Originated	Not Ap	plicable
20	LoanOfficer = officer_2	Loaninfo = CustomerTwo_Loan	BankActions = Create	Environment = Any Value	Condition = Anv Valu	Permit	Originated	Not Ap	olicable
21	LoanOfficer = officer_1	Loaninfo = CustomerTwo_Loan	BankActions = Approve	Environment = Any Value	Condition = Any Valu	Je Deny	Originated	Not Ap	plicable
22	LoanOfficer = officer_2	LoanInfo = CustomerOne_Loan	BankActions = Approve	Environment = Any Value	Condition = Any Valu	Je Deny	Originated	Not Ap	plicable
23	LoanOfficer = officer 2	Loaninfo = CustomerTwo Loan	BankActions = Approve	Environment = Any Value	Condition = Any Valu	Denv	Originated		plicable

Fig. 30. LoanOfficer Policy: Match Results

19 Resolving This Error - Test Case 4 (Separation of Duty)

Looking at the Match Results we will see all rules in the LoanOfficer have come back as "'Not Applicable"' which can only mean that we are missing a rule to cover this particular access request. Upon further inspection we will see that in the LoanOfficer Policy all Loan Officers are Denied to Approve LoanInfo resources except for the individual (officer_one) on this specific resource (CustomerOne_Loan). In addition to missing this rule the Policy Enforcement Algorithm has been selected as Permit-Biased thus why the result is Permit with no rules defined related to the request.

To eliminate this error the policy author would need to go back and either update the Policy Enforcement Algorithm to Deny Biased or go back to the LoanOfficer Policy and create a rule for this scenario.

For this example, let's create a new rule in the LoanOfficer Policy...

LoanOfficer Policy: New Rule (24):

 $(Rule No. = 24) \rightarrow (LoanOfficer = officer_1) \rightarrow (Action = Approve) \rightarrow (Resource = CustomerOne_Loan) \rightarrow decision = Deny$

Fig. 31. LoanOfficer Policy: New Rule (24)

LoanOfficer = officer 1

Originated

Then retest using the same SOD Verification selections as last time we will now get a False Verification result showing that we no longer have a Seperation of Duty error occurring.

OD Verificatio	on(July 5, 2018 17:55:06)(s) Summary			1 rows out of 1			Search	1	×
Status	Name	Verification Type	Verification Technique	Number of Policy(s)	Combination Algorithm	Enforcement Algorithm	6	Policy List	2
UpToDate	SOD Verification(July 5, 2018 17:55:06)	SOD	Single Policy	1	Deny-overrides	Deny Biased	ABA	C:LoanOfficer	Policy
Result(s) with	selected verification (SOD Verification(July 5, 2			👁 1 rows ou			Search		× 1
	SOD Nan SOD : SOI				1.254	D Result be granted			
Result(s) with	selected SOD		💿 2 rov	ws out of 2			Search		
Result(s) with Requiremen		Resource	2 rov Action		ronment (Condition Decis		Verification F	
	nt Schema Subject	Resource Loaninfo = CustomerOne_Lo	Actio	n Envi		Condition Decision = Any Value Perr	sion		

Fig. 32. Updated Policy: No Seperation of Duty

20 SETTING UP THE POLICIES – TEST CASE 5 (INCONSISTENT ASSIGNMENT)

This bank example contains three policies (BankTeller Policy & LoanOfficer Policy & FManager Policy). The attributes in this example have not been changed from previous Test Case examples. The Attribute/Attribute Values included in these policies are as shown in Figure 33.

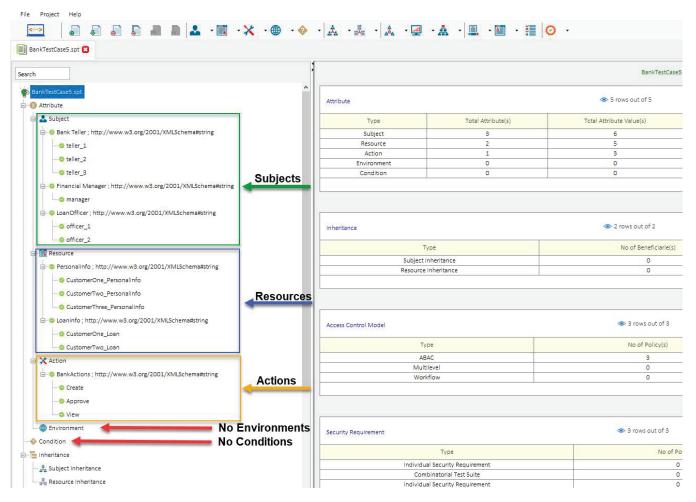


Fig. 33. Test Case 5

21 MODELING YOUR POLICY - TEST CASE 5 (INCONSISTENT ASSIGNMENT)

Now that we have entered our attributes we can model our three policies (BankTeller Policy & LoanOfficer Policy & FManager 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:

BankTeller Policy:

Durik Terrer Torrey.
(Bank Teller = teller_1, CustomerOne_PersonalInfo, Create) \rightarrow Permit
(Bank Teller = teller_1, CustomerOne_PersonalInfo, View) \rightarrow Permit
(Bank Teller = teller_1, CustomerTwo_PersonalInfo, Create) \rightarrow Permit
(Bank Teller = teller_1, CustomerTwo_PersonalInfo, View) \rightarrow Permit
(Bank Teller = teller_1, CustomerThree_PersonalInfo, Create) \rightarrow Permit
(Bank Teller = teller_1, CustomerThree_PersonalInfo, View) \rightarrow Permit
(Bank Teller = teller_2, CustomerOne_PersonalInfo, Create) \rightarrow Permit
(Bank Teller = teller_2, CustomerOne_PersonalInfo, View) \rightarrow Permit
(Bank Teller = teller_2, CustomerTwo_PersonalInfo, Create) →Permit
(Bank Teller = teller_2, CustomerTwo_PersonalInfo, View) \rightarrow Permit
(Bank Teller = teller_2, CustomerThree_PersonalInfo, Create) \rightarrow Permit
(Bank Teller = teller_2, CustomerThree_PersonalInfo, View) \rightarrow Permit
(Bank Teller = teller_3, CustomerOne_PersonalInfo, Create) \rightarrow Permit
(Bank Teller = teller_3, CustomerOne_PersonalInfo, View) \rightarrow Permit
(Bank Teller = teller_3, CustomerTwo_PersonalInfo, Create) →Permit
(Bank Teller = teller_3, CustomerTwo_PersonalInfo, View) →Permit
(Bank Teller = teller_3, CustomerThree_PersonalInfo, Create) \rightarrow Permit
(Bank Teller = teller_3, CustomerThree_PersonalInfo, View) \rightarrow Permit

(Bank Teller = teller_1, CustomerOne_Loan, View) →Permit (Bank Teller = teller_1, CustomerTwo_Loan, View) →Permit (Bank Teller = teller_2, CustomerOne_Loan, View) →Permit (Bank Teller = teller_3, CustomerTwo_Loan, View) →Permit (Bank Teller = teller_3, CustomerTwo_Loan, View) →Permit (Bank Teller = teller_1, CustomerOne_Loan, Approve) →Deny (Bank Teller = teller_1, CustomerTwo_Loan, Approve) →Deny (Bank Teller = teller_2, CustomerOne_Loan, Approve) →Deny (Bank Teller = teller_2, CustomerOne_Loan, Approve) →Deny (Bank Teller = teller_2, CustomerTwo_Loan, Approve) →Deny (Bank Teller = teller_3, CustomerTwo_Loan, Approve) →Deny (Bank Teller = teller_3, CustomerTwo_Loan, Approve) →Deny (Bank Teller = teller_3, CustomerTwo_Loan, Approve) →Deny

LoanOfficer Policy:

(Loan Officer = officer_1, CustomerOne_PersonalInfo, View) \rightarrow Permit
(Loan Officer = officer_1, CustomerOne_PersonalInfo, Create) \rightarrow Permit
(Loan Officer = officer_1, CustomerTwo_PersonalInfo, View) →Permit
(Loan Officer = officer_1, CustomerTwo_PersonalInfo, Create) \rightarrow Permit
(Loan Officer = officer_1, CustomerThree_PersonalInfo, View) \rightarrow Permit
(Loan Officer = officer_1, CustomerThree_PersonalInfo, Create) \rightarrow Permit
(Loan Officer = officer_1, CustomerOne_Loan, View) \rightarrow Permit
(Loan Officer = officer_1, CustomerOne_Loan, Create) \rightarrow Permit
(Loan Officer = officer_1, CustomerTwo_Loan, View) \rightarrow Permit
(Loan Officer = officer_1, CustomerTwo_Loan, Create) \rightarrow Permit
(Loan Officer = officer_2, CustomerOne_PersonalInfo, View) →Permit
(Loan Officer = officer_2, CustomerOne_PersonalInfo, Create) \rightarrow Permit
(Loan Officer = officer_2, CustomerTwo_PersonalInfo, View) \rightarrow Permit
(Loan Officer = officer_2, CustomerTwo_PersonalInfo, Create) \rightarrow Permit
(Loan Officer = officer_2, CustomerThree_PersonalInfo, View) \rightarrow Permit
(Loan Officer = officer_2, CustomerThree_PersonalInfo, Create) \rightarrow Permit
(Loan Officer = officer_2, CustomerOne_Loan, View) \rightarrow Permit
(Loan Officer = officer_2, CustomerOne_Loan, Create) \rightarrow Permit
(Loan Officer = officer_2, CustomerTwo_Loan, View) \rightarrow Permit
(Loan Officer = officer_2, CustomerTwo_Loan, Create) \rightarrow Permit
(Loan Officer = officer_1, CustomerOne_Loan, Approve) →Deny
(Loan Officer = officer_1, CustomerTwo_Loan, Approve) →Deny
(Loan Officer = officer_2, CustomerOne_Loan, Approve) \rightarrow Deny
(Loan Officer = officer_2, CustomerTwo_Loan, Approve) \rightarrow Deny

FManager Policy:

(Manager = manager, CustomerOne_PersonalInfo, Create) →Permit
(Manager = manager, CustomerOne_PersonalInfo, View) \rightarrow Permit
(Manager = manager, CustomerTwo_PersonalInfo, Create) →Permit
(Manager = manager, CustomerTwo_PersonalInfo, View) →Permit
(Manager = manager, CustomerThree_PersonalInfo, Create) \rightarrow Permit
(Manager = manager, CustomerThree_PersonalInfo, View) →Permit
(Manager = manager, CustomerOne_Loan, Create) \rightarrow Permit
(Manager = manager, CustomerOne_Loan, View) \rightarrow Permit
(Manager = manager, CustomerTwo_Loan, Create) \rightarrow Permit
(Manager = manager, CustomerTwo_Loan, View) \rightarrow Permit
(Manager = manager, CustomerOne_Loan, Approve) →Permit
(Manager = manager, CustomerTwo_Loan, Approve) →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: BankTestCase5 and these policies will have been already created for you.

WWW.SECURITYPOLICYTOOL.COM

			Poncy.eps					
BankTeller Policy P	olicy(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	BankTeller Policy	Deny-overrides	Deny Biased	30	July 3, 2018 14:28:32		July 3, 2018 14:28:3	
ule (s) defined wit	h selected policy (BankTeller	Policy1:	@ 30 row	s out of 30			Search	×
Sequence No	Subject	Resource	Action	Environment	Condition	Decision	Inheritance Re	lati
17								
2	Bank Teller = teller_1	Personalinfo = CustomerOne_Personalinfo	BankActions = Create	Environment = Any Value	Condition = Any Value	Permit	Originate	
3	Bank Teller = teller_1 Bank Teller = teller 1	Personalinfo = CustomerOne_Personalinfo	BankActions = View	Environment = Any Value	Condition = Any Value	Permit	Originate	
3	-	Personalinfo = CustomerTwo_Personalinfo	BankActions = Create	Environment = Any Value	Condition = Any Value	Permit	Originate	_
5	Bank Teller = teller_1	Personalinfo = CustomerTwo_Personalinfo	BankActions = View	Environment = Any Value	Condition = Any Value	Permit	Originate	
	Bank Teller = teller_1	Personalinfo = CustomerThree_Personalinfo	BankActions = Create	Environment = Any Value	Condition = Any Value	Permit	Originate	
6	Bank Teller = teller_1	Personalinfo = CustomerThree_Personalinfo	BankActions = View	Environment = Any Value	Condition = Any Value	Permit	Originate	_
7	Bank Teller = teller_2	Personalinfo = CustomerOne_Personalinfo	BankActions = Create	Environment = Any Value	Condition = Any Value	Permit	Originate	
8	Bank Teller = teller_2	Personalinfo = CustomerOne_Personalinfo	BankActions = View	Environment = Any Value	Condition = Any Value	Permit	Originates	-
9	Bank Teller = teller_2	Personalinfo = CustomerTwo_Personalinfo	BankActions = Create	Environment = Any Value	Condition = Any Value	Permit	Originates	_
10	Bank Teller = teller_2	Personalinfo = CustomerTwo_Personalinfo	BankActions = View	Environment = Any Value	Condition = Any Value	Permit	Originater	
11	Bank Teller = teller_2	Personalinfo = CustomerThree_Personalinfo	BankActions = Create	Environment = Any Value	Condition = Any Value	Permit	Originates	_
12	Bank Teller = teller_2	Personalinfo = CustomerThree_Personalinfo	BankActions = View	Environment = Any Value	Condition = Any Value	Permit	Originate	
13	Bank Teller = teller_3	Personalinfo = CustomerOne_Personalinfo	BankActions = Create	Environment = Any Value	Condition = Any Value	Permit	Originates	-
14	Bank Teller = teller_3	Personalinfo = CustomerOne_Personalinfo	BankActions = View	Environment = Any Value	Condition = Any Value	Permit	Originates	_
15	Bank Teller = teller_3	Personalinfo = CustomerTwo_Personalinfo	BankActions = Create	Environment = Any Value	Condition = Any Value	Permit	Originate	
16	Bank Teller = teller_3	Personalinfo = CustomerTwo_Personalinfo	BankActions = View	Environment = Any Value	Condition = Any Value	Permit	Originates	-
17	Bank Teller = teller_3	Personalinfo = CustomerThree_Personalinfo	BankActions = Create	Environment = Any Value	Condition = Any Value	Permit	Originates	_
18	Bank Teller = teller_3	Personalinfo = CustomerThree_Personalinfo	BankActions = View	Environment = Any Value	Condition = Any Value	Permit	Originater	
19	Bank Teller = teller_1	LoanInfo = CustomerOne_Loan	BankActions = View	Environment = Any Value	Condition = Any Value	Permit	Originates	-
20	Bank Teller = teller_1	Loaninfo = CustomerTwo_Loan	BankActions = View	Environment = Any Value	Condition = Any Value	Permit	Originate	
21	Bank Teller = teller_2	Loaninfo = CustomerOne_Loan	BankActions = View	Environment = Any Value	Condition = Any Value	Permit	Originates	
22	Bank Teller = teller_2	Loaninfo = CustomerTwo_Loan	BankActions = View	Environment = Any Value	Condition = Any Value	Permit	Originates	
23	Bank Teller = teller_3	LoanInfo = CustomerOne_Loan	BankActions = View	Environment = Any Value	Condition = Any Value	Permit	Originate	
24	Bank Teller = teller_3	LoanInfo = CustomerTwo_Loan	BankActions = View	Environment = Any Value	Condition = Any Value	Permit	Originates	
25	Bank Teller = teller_1	LoanInfo = CustomerOne_Loan	BankActions = Approve	Environment = Any Value	Condition = Any Value	Deny	Originate	
26	Bank Teller = teller_1	Loaninfo = CustomerTwo_Loan	BankActions = Approve	Environment = Any Value	Condition = Any Value	Deny	Originate	
27	Bank Teller = teller_2	Loaninfo = CustomerOne_Loan	BankActions = Approve	Environment = Any Value	Condition = Any Value	Deny	Originate	-
28	Bank Teller = teller_2	LoanInfo = CustomerTwo_Loan	BankActions = Approve	Environment = Any Value	Condition = Any Value	Deny	Originate	
29	Bank Teller = teller_3	LoanInfo = CustomerOne_Loan	BankActions = Approve	Environment = Any Value	Condition = Any Value	Deny	Originate	
30	Bank Teller = teller_3	LoanInfo = CustomerTwo_Loan	BankActions = Approve	Environment = Any Value	Condition = Any Value	Deny	Originates	d

Policy.eps

Fig. 34. BankTeller Policy

oanOfficer Policy	Policy(s) Summary		I rows out of 1				Search	×
Model	Policy Name	Rule Combination Algorithm	Policy Enforcement Algorithm	No. of Rule(s)	Time Created		Last Modified	
ABAC	LoanOfficer Policy	Deny-overrides	Deny Biased	24	July 3, 2018 14:40:4	46	July 3, 2018 14:40:4	16
tule (s) defined wit	th selected policy (LoanOfficer F	Policy):	@ 24 row	s out of 24			Search	×
Sequence No	Subject	Resource	Action	Environment	Condition	Decision	Inheritance Rela	ation
1	LoanOfficer = officer_1	Personalinfo = CustomerOne_Personalinfo	BankActions = View	Environment = Anv Value	Condition = Any Value	Permit	Originated	
2	LoanOfficer = officer_1	Personalinfo = CustomerOne_Personalinfo	BankActions = Create	Environment = Any Value	Condition = Any Value	Permit	Originated	0
3	LoanOfficer = officer_1	Personalinfo = CustomerTwo_Personalinfo	BankActions = View	Environment = Anv Value	Condition = Any Value	Permit	Originated	
4	LoanOfficer = officer_1	Personalinfo = CustomerTwo_Personalinfo	BankActions = Create	Environment = Any Value	Condition = Any Value	Permit	Originated	5
5	LoanOfficer = officer_1	Personalinfo = CustomerThree_Personalinfo	BankActions = View	Environment = Any Value	Condition = Any Value	Permit	Originated	
6	LoanOfficer = officer_1	Personalinfo = CustomerThree_Personalinfo	BankActions = Create	Environment = Any Value	Condition = Any Value	Permit	Originated	
7	LoanOfficer = officer_1	Loaninfo = CustomerOne_Loan	BankActions = View	Environment = Any Value	Condition = Any Value	Permit	Originated	8
8	LoanOfficer = officer_1	Loaninfo = CustomerOne_Loan	BankActions = Create	Environment = Any Value	Condition = Any Value	Permit	Originated	8
9	LoanOfficer = officer_1	LoanInfo = CustomerTwo_Loan	BankActions = View	Environment = Any Value	Condition = Any Value	Permit	Originated	8
10	LoanOfficer = officer_1	Loaninfo = CustomerTwo_Loan	BankActions = Create	Environment = Any Value	Condition = Any Value	Permit	Originated	8
11	LoanOfficer = officer_2	Personalinfo = CustomerOne_Personalinfo	BankActions = View	Environment = Any Value	Condition = Any Value	Permit	Originated	5
12	LoanOfficer = officer_2	Personalinfo = CustomerOne_Personalinfo	BankActions = Create	Environment = Any Value	Condition = Any Value	Permit	Originated	R
13	LoanOfficer = officer_2	Personalinfo = CustomerTwo_Personalinfo	BankActions = View	Environment = Any Value	Condition = Any Value	Permit	Originated	
14	LoanOfficer = officer_2	Personalinfo = CustomerTwo_Personalinfo	BankActions = Create	Environment = Any Value	Condition = Any Value	Permit	Originated	į.
15	LoanOfficer = officer_2	Personalinfo = CustomerThree_Personalinfo	BankActions = View	Environment = Any Value	Condition = Any Value	Permit	Originated	8
16	LoanOfficer = officer_2	Personalinfo = CustomerThree_Personalinfo	BankActions = Create	Environment = Any Value	Condition = Any Value	Permit	Originated	8
17	LoanOfficer = officer_2	Loaninfo = CustomerOne_Loan	BankActions = View	Environment = Any Value	Condition = Any Value	Permit	Originated	Å.
18	LoanOfficer = officer_2	LoanInfo = CustomerOne_Loan	BankActions = Create	Environment = Any Value	Condition = Any Value	Permit	Originated	ŝ.
19	LoanOfficer = officer_2	Loaninfo = CustomerTwo_Loan	BankActions = View	Environment = Any Value	Condition = Any Value	Permit	Originated	Ś.
20	LoanOfficer = officer_2	LoanInfo = CustomerTwo_Loan	BankActions = Create	Environment = Any Value	Condition = Any Value	Permit	Originated	8
21	LoanOfficer = officer_1	Loaninfo = CustomerOne_Loan	BankActions = Approve	Environment = Any Value	Condition = Any Value	Deny	Originated	8
22	LoanOfficer = officer_1	LoanInfo = CustomerTwo_Loan	BankActions = Approve	Environment = Any Value	Condition = Any Value	Deny	Originated	8
23	LoanOfficer = officer_2	LoanInfo = CustomerOne_Loan	BankActions = Approve	Environment = Any Value	Condition = Any Value	Deny	Originated	8
24	LoanOfficer = officer_2	Loaninfo = CustomerTwo_Loan	BankActions = Approve	Environment = Anv Value	Condition = Any Value	Deny	Originated	

Fig. 35. LoanOfficer Policy

FManager Policy P	olicy(s) Summary		I rows out of 1	Se	arch 🚺		
Model	Policy Name	Rule Combination Algorithm	Policy Enforcement Algorithm	No. of Rule(s)	Time Created		Last Modified
ABAC	FManager Policy	Permit-overrides	Permit Biased	12	July 3, 2018 15:03:03	ıl	uly 3, 2018 15:03:03
Rule (s) defined wi	ith selected policy (FManager Policy):		12 rows out	ut of 12		Se	earch
Sequence No	Subject	Resource	Action	Environment	Condition	Decision	Inheritance Relation
1	Financial Manager = manager	Personalinfo = CustomerOne_Personalinfo	BankActions = Create	Environment = Anv Value	Condition = Any Value	Permit	Originated
2	Financial Manager = manager	Personalinfo = CustomerOne_Personalinfo	BankActions = View	Environment = Any Value	Condition = Any Value	Permit	Originated
3	Financial Manager = manager	Personalinfo = CustomerTwo_Personalinfo	BankActions = Create	Environment = Any Value	Condition = Any Value	Permit	Originated
4	Financial Manager = manager	Personalinfo = CustomerTwo_Personalinfo	BankActions = View	Environment = Any Value	Condition = Any Value	Permit	Originated
5	Financial Manager = manager	Personalinfo = CustomerThree_Personalinfo	BankActions = Create	Environment = Any Value	Condition = Any Value	Permit	Originated
6	Financial Manager = manager	Personalinfo = CustomerThree_Personalinfo	BankActions = View	Environment = Any Value	Condition = Any Value	Permit	Originated
7	Financial Manager = manager	Loaninfo = CustomerOne_Loan	BankActions = Create	Environment = Any Value	Condition = Any Value	Permit	Originated
8	Financial Manager = manager	Loaninfo = CustomerOne_Loan	BankActions = View	Environment = Any Value	Condition = Any Value	Permit	Originated
9	Financial Manager = manager	Loaninfo = CustomerTwo_Loan	BankActions = Create	Environment = Any Value	Condition = Any Value	Permit	Originated
10	Financial Manager = manager	LoanInfo = CustomerTwo_Loan	BankActions = View	Environment = Any Value	Condition = Any Value	Permit	Originated
11	Financial Manager = manager	Loaninfo = CustomerOne_Loan	BankActions = Approve	Environment = Any Value	Condition = Any Value	Permit	Originated

Fig. 36. FManager Policy

22 UNDERSTANDING THIS ERROR - TEST CASE 5 (INCONSISTENT ASSIGNMENT)

Now that we have our policies set up let's discuss the error we will be looking at in this fifth example. When XACML policies are being constructed there is potential for "Inconsistent Assignment" errors being created. This error occurs when a policy author has unintentionally assigned attributes, conditions, rule or other policy variables/values incorrectly in different policies. For example, Attribute LoanOfficer could be incorrectly termed as LoanOfficr or LoanOfficer in different policy documents.

Doing so, could result in a significant security vulnerability due to the system providing unintended access decisions as consequence of the incorrectly defined policy value(s). Organizations with very large and complex policies are especially at risk for this type of error as small inconsistencies could very easily go unnoticed if they do not have a rigorous method for modeling and testing their policies.

23 RESOLVING THIS ERROR - TEST CASE 5 (INCONSISTENT ASSIGNMENT)

Security Policy Tool by default prevents this error from occurring. After modeling your policies you can automatically convert your policies into XACML 3.0. The XACML Editor included with Security Policy Tool contains intuitive features to help you create secure and accurate XACML documents. Several robust features that prevent inconsistent assignment errors include Integrity Verification, Syntax Error Detection, Assistive XACML Code Completion, and others.

rch			
Personalinto ; http://www.w3.org/2001/XMLSchema#string	A BankTeller Policy Po	oliny(s) Summany	
😳 CustomerOne_PersonalInfo	bunktenet roney r	oney(s) commony	
😋 CustomerTwo_PersonalInfo	Model	Policy Name	Rule Combination Algorithm
CustomerThree_PersonalInfo	ABAC	BankTeller Policy	Deny-overrides
□ _ O LoanInfo ; http://www.w3.org/2001/XMLSchema#string			
CustomerOne Loan			
- O CustomerTwo Loan			
-	Rule (s) defined wit	h selected policy (BankTeller I	Policy):
∋- X Action			
BankActions ; http://www.w3.org/2001/XMLSchema#string	Sequence No	Subject	Resource
Create	1	Bank Teller = teller_1	PersonalInfo = CustomerOne_PersonalInf
	2	Bank Teller = teller_1	Personalinfo = CustomerOne_Personalinf
🥝 Approve	3	Bank Teller = teller_1	Personalinfo = CustomerTwo_Personalinf
View	4	Bank Teller = teller_1	Personalinfo = CustomerTwo_Personalinf
-	5	Bank Teller = teller_1	Personalinfo = CustomerThree_Personalin
	6	Bank Teller = teller_1	Personalinfo = CustomerThree_Personalin
Condition	7	Bank Teller = teller_2	Personalinfo = CustomerOne_Personalinf
Inheritance	8	Bank Teller = teller_2	Personalinfo = CustomerOne_Personalinf
	9	Bank Teller = teller_2	PersonalInfo = CustomerTwo_PersonalInf
	10	Bank Teller = teller_2	PersonalInfo = CustomerTwo_PersonalInf
💑 Resource Inheritance	11	Bank Teller = teller_2	Personalinfo = CustomerThree_Personalin
Model	12	Bank Teller = teller_2	Personalinfo = CustomerThree_Personalin
ABAC	13	Bank Teller = teller_3	Personalinfo = CustomerOne_Personalinf
	14	Bank Teller = teller_3	Personalinfo = CustomerOne_Personalinf
BankTeller Policy: Denv-overrides & Denv Riased	15	Bank Teller = teller_3	Personalinfo = CustomerTwo_Personalinf
Add a New ABAC Rule	16	Bank Teller = teller_3	Personalinfo = CustomerTwo_Personalinf
Paste Rule(s)	17	Bank Teller = teller_3	Personalinfo = CustomerThree_Personalin
FManager Police Update ABAC Policy	18	Bank Teller = teller_3	Personalinfo = CustomerThree_Personalin
	19	Bank Teller = teller_1	Loaninfo = CustomerOne_Loan
Workflow Convert Policy into XACML 3.0 Policy	20	Bank Teller = teller_1	LoanInfo = CustomerTwo_Loan
Conterest only into some did to help	21	Bank Teller = teller_2	Loaninfo = CustomerOne_Loan
Access Control Securi Convert Policy into XACML 3.0 Policy	22	Bank Teller = teller_2	LoanInfo = CustomerTwo_Loan
🖳 🛄 Individual Security Requirement	23	Bank Teller = teller_3	Loaninfo = CustomerOne_Loan
Separation of Duty Security Requirement	24	Bank Teller = teller_3	Loaninfo = CustomerTwo_Loan
	25	Bank Teller = teller_1	Loaninfo = CustomerOne_Loan
Combinatorial Test Suite	26	Bank Teller = teller_1	LoanInfo = CustomerTwo_Loan

Fig. 37. Convert Modeled Policies into XACML

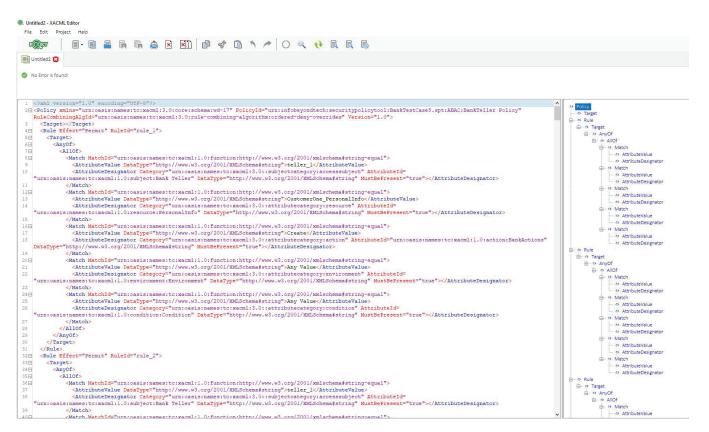


Fig. 38. Converted BankTeller Policy: No Errors

24 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 \rightarrow <u>Lite Version</u>.



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.