Detecting inconsistencies of mixed secrecy models and business policies

Waël Hassan[1], Luigi Logrippo[2]

[1] Université d'Ottawa, École d'ingénierie et de technologie de l'information
[2] Université du Québec en Outaouais, Département d'informatique et ingénierie
wael@acm.org, luigi@uqo.ca

Abstract. Several secrecy models are known in practice, and governance requirements may make it necessary to combine them in order to implement the secrecy policies of an enterprise. However, inconsistencies may arise as a result of implementing multiple secrecy models in an enterprise network, and these inconsistencies may well undermine the intended functioning of the system. We propose a method to detect and report these inconsistencies at the time when the secrecy system is designed. The method is based on specifying the models and their secrecy policies in logic and applying a formal analyzer. A given combination of such models can be analysed for inconsistency, and if found inconsistent this combination of models must be modified before implementation. Our proposed method is demonstrated by using as example a mixed model involving Bell-La Padula (BLP) and Role based access control (RBAC) in addition to separation of concerns (SOC). The logic analyzer Alloy is used to check consistency. The method's principles are conjectured to be generic and hence can apply to any secrecy model.

Keywords: Governance, secrecy models, consistency, formal methods, Alloy.

1 Introduction

Secrecy *requirements* are part of governance requirements in typical enterprises. Such requirements specify who has access to what and when, possibly in what order. In order to satisfy such requirements, secrecy *models* have been developed. These models specify properties such as: multi-level protection, separation of concerns, no-read-up, inheritance of rights, pessimistic mode. As enterprises increasingly integrate networks with various tools and operating systems there is a need for integrating secrecy models with different characteristics, possibly even models that may conflict in some cases. Further, as enterprises specify business policies, situations may arise were these policies cannot be implemented since they may violate other policies or the chosen secrecy model.

Several secrecy models are used in practice and others have been studied in theory. Well-known secrecy models are Bell-La Padula (BLP) and Chinese Wall (CW), other models will be mentioned later. Each model is characterized by a set of fundamental *properties* that represent intrinsic *meta-policies*. As mentioned, practical necessities may motivate the use of some of these models in combination [1][2][3] , and in addition other business *policies* may be necessary, in order to implement the secrecy goals determined by the enterprise.

Combining multiple secrecy models with business policies in the same network may cause inconsistencies. An inconsistency, or logical contradiction, may undermine the proper functioning of a real system, creating situations where some user may be allowed access to an object according to a rule, and not allowed according to another rule. Clearly, it is important that inconsistencies be detected at the design stage of the system.

This paper will show the capability of logic based models to represent an abstraction of secrecy models and business policies. Once this is done, a logic analyzer can detect inconsistencies prior to system deployment.

We differentiate between two types of inconsistencies: model inconsistencies, which are inconsistencies between secrecy models; and system inconsistencies, which are inconsistencies between business policies and secrecy models. A model inconsistency indicates failure to combine tools or networks. On the other hand, a system inconsistency is an indication that some business policies need to be revised.

Combination of models can be done according to one of two modes, that we shall call *mixed* and *hybrid*. In a system that adopts mixed-mode secrecy, it is possible to implement policies following any parent model. Mixed models *combine* the parent model's policies and their properties. Hybrid models, on the other hand, *inherit* properties from parent models or may include additional properties not present in parent models. Schematizing in simple mathematical terms, suppose a mixed mode secrecy system including two parent models with properties P_1 and P_2. The set of properties of the mixed model will be $M = P_1 \cup P_2$. On the other hand, in a hybrid model obtained by combining the same parents, we can only say that the set of properties H obeys the relation $H \cap (P_1 \cup P_2) \neq \varnothing$, meaning that the hybrid model shares some properties with parent models, and it may include others. Mixed models will be the focus of this paper.

Inconsistency in a mixed model can be a consequence of the fact that that the models that have been composed are incompatible and hence cannot be mixed. For example, consider two multilevel secrecy type models. Both models include three classifications and a 'no-read-up' policy, yet one of them has an 'allow-write-down' policy while the other bans it. These two models cannot be mixed given this contradiction.

Policy combinations may be a cause of confusion and concern to system administrators: confusion, because administrators may not understand the implications of adding or removing policies; concern, since these policies may grant rights that threaten secrecy, or revoke needed rights that are instrumental to the business. To illustrate this point, consider an environment that mixes Bell-La Padula (BLP) with Role Based Access Control (RBAC) model in addition to Separation of Concerns (SOC) (some explanations on these models will be given later). A BLP policy may state that client data is classified at the Secret level, while all aggregate calculations (such as revenue and cumulative balances) are at the Top-Secret level. A RBAC policy may state that tellers have access to update client accounts, and managers can authorise withdrawals higher than 10K. A SOC policy may state that there should be no data sharing between the marketing and accounting departments.

The system administrator may have at her disposal tools that allow her to determine the results of these combined policies in specific cases, e.g., can Alice, a teller, access the balance of client Bob? Much more rare are tools that identify problematic cases in general, and one such tool we are proposing in this paper. .

This paper shows how mixed secrecy models can be checked for consistency. We apply our method to commonly known secrecy policies such as those we have mentioned. We use the formal language Alloy to define the models and its associated logic analysis tool to validate consistency. We conjecture that the method is generic and hence it can be applied to other models and model combinations. Our examples show that enterprises, network providers, operating systems, and open source communities can benefit from applying logical analysis tools to their secrecy models, at the time the secrecy system is being defined or modified. Hopefully, in the not too distant future industrial tools to do the analysis we propose will become available.

Section 2 provides the background for this work. Section 3 discussed the motivation detailing some practical examples. Section 4 presents basic definitions needed to understand the approach in the paper. Section 5 introduces the method and providing sample examples. Section 6 shows a logic representation of BLP, RBAC and SOC policies. Section 7 provides a review of related work. Section 8 concludes the paper.

2. Background

The Bell-La Padula (BLP) model is a security model that has its origins in the military, and is based on a system of security classifications and clearances. These represent an object's sensitivity and the degree of trust placed on a subject. Security levels and security clearances are partially ordered. A user has access to an object if his security clearance is higher than the sensitivity level of the object.

Role-Based Access Control (RBAC) is a secrecy mechanism that associates roles with individual users. The essence of RBAC lies in the notion of role as an intermediary between subjects and objects: roles are given access rights to objects while subjects are associated with roles.

Other types of secrecy mechanisms that will not be considered in this paper but that may come into consideration in this type of analysis are the Chinese Wall and the Delegation of Authority. The Chinese Wall (CW) [23] policy was originally aimed at the financial sector where consulting companies assign consultants to provide audit and consultation services to client companies. Insider information creates conflicts of interest for the consultant if she were able to access data of two competing companies. The objective of Delegation of Authority is giving control to another subject, , in other words a subject can delegate to another subject its right of access to objects.

We use Alloy-4[1] for logic analysis. Alloy consists of a formal language and a related logic analyzer and therefore implements a formal method, which can be used to precisely capture and analyze logical specifications of systems. The Alloy language is a structural modeling language based on first-order logic. Alloy features the ability

[1] http://alloy.mit.edu

to express complex structures whether static or dynamic. It is declarative in nature providing full logical ability with conjunction and negation. Using Alloy, a system can be described as a set of constraints.

3 Motivation

There are industrial and research drivers for this work. Industry is interested in building interoperable environments by testing that their integrated tools and operating systems can interconnect given their respective secrecy models. On the other hand, the research world finds in this area the possibility of demonstrating the power of formal methods, particularly relational logic, to model and analyse secrecy models in mixed, hybrid, or single modes.
The need for understanding the consequences of combined secrecy policies from multiple domains has been documented in [4][5][24]. There are others who are keen on developing logic representations of combined access control models[6][7][25]. This paper shows through examples the ability of formal analysers such as Alloy to model and identify system inconsistencies.

Studying combined secrecy models is motivated by the increase in the number of security vulnerabilities as a result of the use of multiple security policies [2]. Management is driven by cost cutting measures towards tool convergence between enterprise content management, enterprise communication and collaboration, and business intelligence. The requirement for integration has prompted Nokia [1] to develop a Multiple Domain Security Tool that can be deployed by an enterprise to interconnect securely multiple internal departments or extranets, or can be deployed by a service provider hosting multiple small customers, each depending on its security policy.

The integration of secrecy models is a characteristic of two areas: Operating system/networks (OS), and tool integration.

For the first area, there are several environments where the network is restricted to OS applications supporting *multilevel secure systems* such as BLP [8]. Military OS are dominated by the use of BLP for highly secure networks [8]. Windows Vista [2][27] on the other hand, allows several types of policies including least privilege user based access[2], process based levels of security 'with inheritance', and multi-level security for user interface communication. Furthermore, Vista suggests classification of 'restricted' processes and 'un-restricted' ones. Our method can be applied to test the ability of integrating Vista's security model with a multilevel security system.

Integrating the Plone/Zope[3] web content management tool with a project management tool such as Clarity© from CA is an industrial example of the trend towards tool integration. In Plone/Zope there are four components to an authorisation : users/user groups, roles (Manager, Author), permissions (rwx[4]), acquisition (containment hierarchy). Their access model supports delegations amongst other mechanisms. Clarity© on the other hand, has the following

2 Also called user account protection
3 Used by the ACM portal
4 rwx: read write execute

components: users, roles, organisational structures, financial structures, process structures, in addition to user interface access control lists. Clarity supports positive authorizations in addition to optional inheritance of rights.

It would be possible to check the ability of these systems to integrate by using logic analysers. Many potential conflicts can be detected sooner and at a lower cost than with manual techniques and empirical tests. The need and the scope of the latter may not be eliminated, but will be reduced.

While this paper does not consider examples of complexity comparable to the one of the systems we have just mentioned, some of the same ideas can be applied.

4 Domain and definitions

Before proceeding we would like to present some important definitions needed to describe our approach. We first present a distinction between security and secrecy. Often one may find that these terms are used interchangeably, however they shouldn't be. In computing systems, a system is said to be *secure* if it is able to guarantee its a) availability, reliability b) authentication including non-repudiation c) secrecy (confidentiality) and e) integrity [9]. *Secrecy* is defined as the ability to hide specific information from certain users according to some policy specifying which users should be forbidden to acquire what kind of information [10]. These policies are generic, involve sets of users and objects, instead of being specific such as: *Alice can read the bulletin board*.

Governance Requirements: Informally defined, a *requirement* is a documented need of what a particular product or service should be or do [11] . It is a statement that identifies necessary attributes, capabilities, characteristics, or qualities of a system for it to have value and utility to a user [12]. Governance requirements may specify secrecy models, secrecy properties, and related business policies. For example, a governance requirement may be to have a secrecy model that combines the properties or functionalities of SOC and BLP. An enterprise property can require four levels of security. A business policy can be to prevent network sharing between the human resources department and the information technology department.

Corporate Governance: Informally defined, corporate governance is the system by which business corporations are directed and controlled. Corporate governance has several sub-processes, makes use of tools and methods, and dictates a certain organisational structure. This structure specifies the distribution of rights and responsibilities among different participants in a corporation and spells out the rules and procedures for making decisions on corporate affairs. It also provides the processes through which a company's objectives are set, the means for attaining those objectives and for monitoring the performance. The law and business policies are the sources of governance requirements. Hence *governance requirements* prescribe legal and business attributes, capabilities, or qualities of an enterprise system [13]. They state the models involved along with enterprise properties and business policies. Part of governance requirements are the secrecy policies, models and mechanisms that an enterprise uses.

Secrecy meta policy: A part of governance requirements are secrecy requirements. Secrecy requirements specify *meta policies*. We use this term to identify the high-level rules according to which access control must be regulated [14]. Examples of meta-policies are: access to financial information is based on role participation; there are two classifications for data: private and non-private; subjects who have access to private data cannot delegate their access to subjects who have access restricted to non-private data.

A secrecy model: is a combination of secrecy meta policies. A secrecy model can be represented in a set of data types with their relations. Some of these relations describe policies that are intrinsic to the model, these are the meta-policies of the model. For example, a multilevel-secrecy model can have the following classifications: *TopSecret*, *Secret*, *Classified* and *Unclassified*. *TopSecret* is a higher classification than *Secret*, *Classified*, and *Unclassified* and so on. Two meta policies are usually included in this model: *No-read-up* and *No-write-down*. *No-read-up* implies that no one classified *Secret* can read *TopSecret* information. Examples of secrecy models are the well-known ones presented above: BLP, RBAC, CW.

A business Policy: A business policy, in this context, is one that specifies the modality of access over objects. As mentioned, it explicitly states who has access to what and when, and in what order. A business policy can specify for example that managers only have read access to their subordinates HR files. More specifically, it can say that Lena has access to HR files. A business policy can also say that employees cannot approve their own time-sheet, rather, they are only accessible by their managers.

Relation: A secrecy model is a combination of a set of meta-policies. Each policy suggesting a governance requirement. Meta policies have priority over a business policy. Furthermore, a business policy follows the format specified in secrecy model. For example if the secrecy model specifies role based access then business policies could have the form: users belong to roles and roles have positive or negative modality of access over objects. In addition, meta policies change infrequently compared to business policies. A meta policy is usually changed upon the introduction of a new network or tool that has a different secrecy model. Whereas manging users membership in roles and role access rights, for example, is much more frequent. Finally, meta-policies are usually a select few policies where as business policies are numerous.

Figure 1 shows in schematic form the process of checking governance requirements for consistency. The figure contains two main entities and two processes. The entities are : *Governance Requirements*, and the *Live System* implementing these requirements. The two processes are the *Model* and *System* consistency checks.

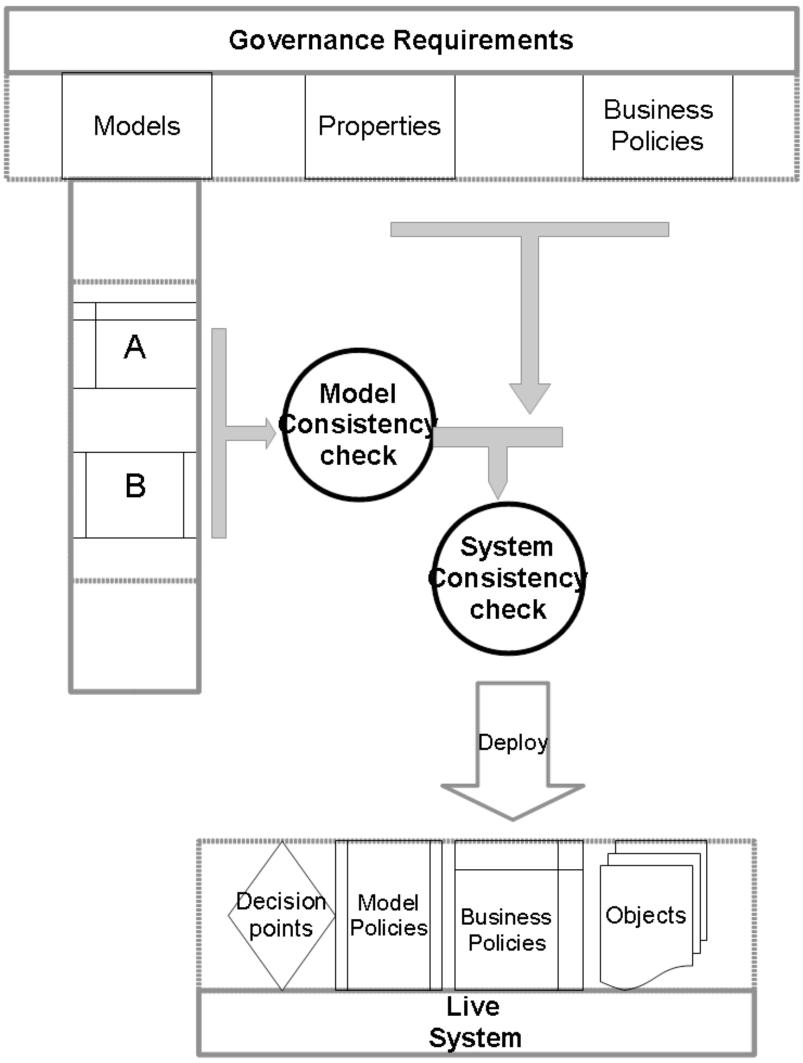

Fig. 1. Phases of formal design

Even though governance requirements from a business perspective may come as a single package, checking them requires a two-step process. As per our method, the mixed model needs to be checked for consistency. This is shown in the figure using the *Model Consistency Check* process. If the check fails, the metapolicies of the combined models are inconsistent and the model needs to be redesigned. Otherwise we proceed to the next step: the governance requirements represented by the business policies need to be integrated with the meta-policies to form a logical model of the system. This analysis checks if the combination of model meta-policies and business policies causes inconsistency.

If a system is free of inconsistencies , it can be deployed in a live environment as shown in the third entity in Fig. 1 labeled *live system*. The figure represents some common live system artifacts such as the decision points, model policies, business policies, and data objects.

This paper is concerned with the processes of checking consistency of models and system.

5. Logic Analysis Method

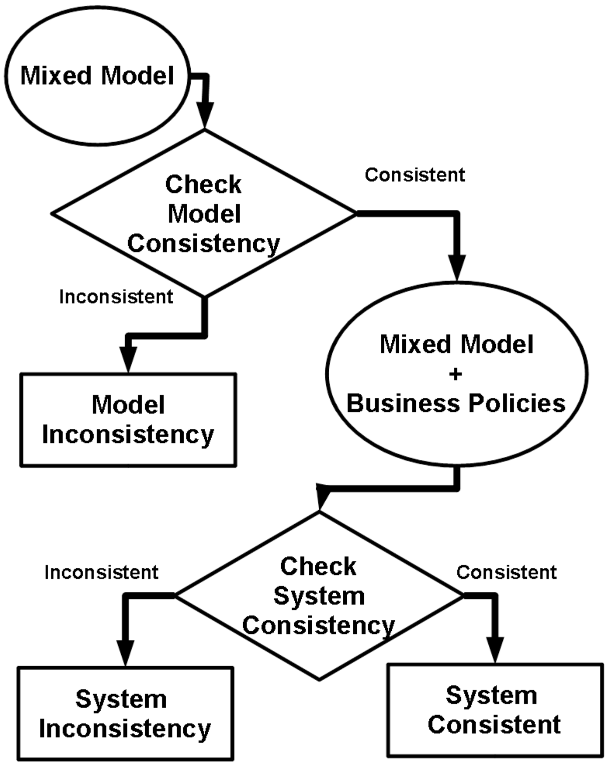

Fig. 2. *Validation Method*

This section shows the steps taken to check a mixed secrecy model for consistency along with business policies.

To recapitulate, in a mixed secrecy model there is always the possibility of inconsistency.

We address two types of inconsistencies: model, and system. Model inconsistency is defined as a logical conflict between properties and meta policies, i.e. the governance requirements, of two models. Such conflicts render the mixed model void and not useful. On the other hand, a system inconsistency is defined as a result of conflict between user policies or between user policies and meta policies.

Figure 2 schematizes the validation procedure using our method. The mixed models are represented using the analyzer language and are presented to the consistency checker. If the consistency checker fails then we can deduce that these two models cannot coexist and hence are deemed incoherent. If no model inconsistency is detected, there remains the possibility that there are inconsistencies when the business policies are added. If there is such interaction, it would be detected in the second validation.

Alloy allows a user to check if a model is inconsistent, in addition to providing sample scenarios showing the reasons for inconsistency. This can be done by appropriate use of the two statements *fact* and *assert*. A *fact* describes a model property. A combination of facts may cause a model to be inconsistent, in which case the tool will state *no instance found*, with no additional information. Inconstent scenarios will be generated by stating the same property by using an *assertion*, as we will see in the examples below.

6. Case Study

Consider two multilevel secrecy models, suggesting levels of classifications and rules for read and write access across levels.

Assume that model A includes two classifications (Classified and Unclassified), in addition to two meta-policies.

Meta Policy-A-1: No-read-up, which means that subjects at the Unclassified-level are not able to read objects at the Classified-level.

Meta Policy-A-2: No-write-down, which prevents subjects at the Classified level to write objects at the Unclassified level.

These two meta-policies can be written in Alloy using the following code:

No Read-up (A1):

```
no ((Classified.~sec).~R  & UNClassified.~level)
```

This statement says that there is no intersection between users at the unclassified level and users who have read rights to objects at the classified level. No intersection implies that it is not possible to be in both sets at the same time.

No Write-down (A2):

```
no ((UNClassified.~sec).~W  & Classified.~level)
```

This statement says that there is no intersection between the set of users who have read rights to objects at the Classified level and the set of users at the Unclassified level. No intersection implies that it is not possible for a user to be in both sets at the same time.

Meta Policy-B-1: No-read-up, which means that subjects at the Unclassified-level are not able to read objects at the Classified-level, in addition subjects at the Classified level cannot read objects at the TopSecret level, furthermore, subjects at un-classified level cannot read objects at TopSecret level

Meta Policy-B-2: Allow-write-down for Secret and Unclassified classifications only. B-2 translates to two statements: a) Subjects at the Secret-level can write on objects at the Unclassified-level; b) subjects at the Top-Secret level cannot write objects at the classified level.

No Read-up (B1):

```
no ((Classified.~sec).~R  & UNClassified.~level)
no ((TopSecret.~sec).~R  & Classified.~level)
```

Allow write-down (B-2):

```
no ((TopSecret.~sec).~W  & Classified.~level)
some ((UNClassified.~sec).~W & Classified.~level)
```

6.1 Model consistency

Combining models A and B generates an inconsistency since the no-write-down policy A-2 of model A contradicts the write-down policy B-2 in model B

```
Executing "Run example"
Solver=sat4j Bitwidth=4 MaxSeq=4 SkolemDepth=1 Symmetry=20
0 vars. 0 primary vars. 0 clauses. 11ms.
No instance found. Predicate may be inconsistent. 0ms.
```

Fig. 3. *Alloy output showing inconsistency*

In other words, the logical conjunction of the two models A and B is always false. Figure 3 shows the results produced by the logic analyzer when the inconsistent model is checked.

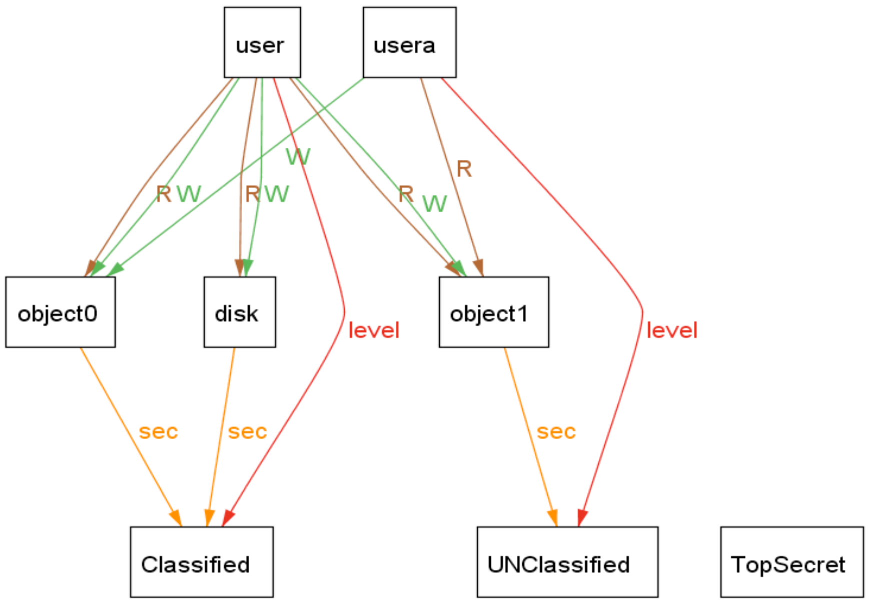

Fig. 4. *Visualized version of the inconsistency*

We then asked the Alloy analyzer to generate an instance of the violation, by using the *assert* statement. The result is shown in Fig. 4. In this figure it is possible to see clearly how a 'user' at the Classified level has write access to object1 which is at the unclassified level, therefore breaking the no-write-down policy.

Note that Alloy shows a possible world where the inconsistency occurs, withoug pinpointing its reasons. These must be found by inspection. To facilitate the task in complex scenarios, the tool provides a filtering facility, which allows selecting different aspects of a scenario.

6.2 System consistency

For studying system inconsistency we create a model instance of per the mixed model shown in Figure 5. The model shows three classifications (*Classified, TopSecret, UNClassified*). A user has a certain level that is attached to a classification. An object has a security relationship with the meta-class classification. A user can have read and write relationships with objects. The two class extensions *usera* and *disk* are instances of the classes *user* and *object* respectively. We define two policies for the model: No-read-up and No-write-down.

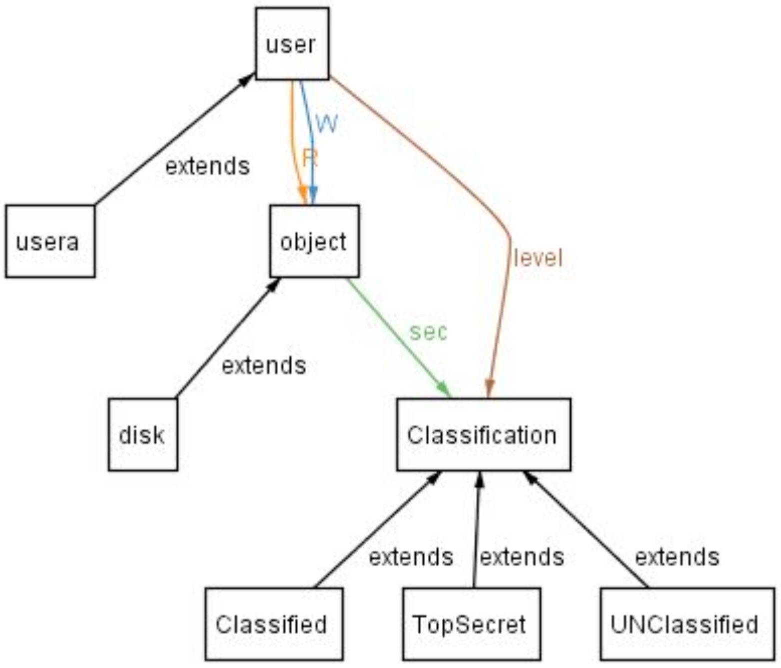

Fig. 5. Meta-model of mixed models A & B

In this example we show how a business policy can interact with model meta-policies. Assume that we have a business policy S stating that *usera* is at the unclassified level. S also states that *usera* has access to a certain network component labeled *disk* which is at the secret classification. We can check if the combined model including the newly added policy S can cause logical conflicts using the Alloy code shown below:

```
fact businessPolicyS {
usera.level =   UNClassified
some usera.R
some usera.W
Classified in disk.sec
disk in usera.R
}
```

This code says that userA is at the unclassified level, there are some objects that are accessible for read by userA, there are some objects that are accessible for write by userA, the object disk is at the classified level of secrecy, finally it says that userA is able to access the disk for read.

We checked policy S with the combined policy meta model. The result was an inconsistency resulting in the impossibility of implementing such a combination of policies, see Figure 5.

```
Executing "Run example"
Solver=sat4j Bitwidth=4 MaxSeq=4 SkolemDepth=1 Symmetry=20, 259 vars.
40 primary vars. 379 clauses. 6ms.
No instance found. Predicate may be inconsistent. 1ms.
```

Fig. 6. *System inconsistency detected*

If we *assert* the policy, many counterexamples can be produced. Each counterexample will show that it is not possible to allow *usera* to have access to *disk* when *usera* is at the UNClassified level and *disk* at the Classified level.

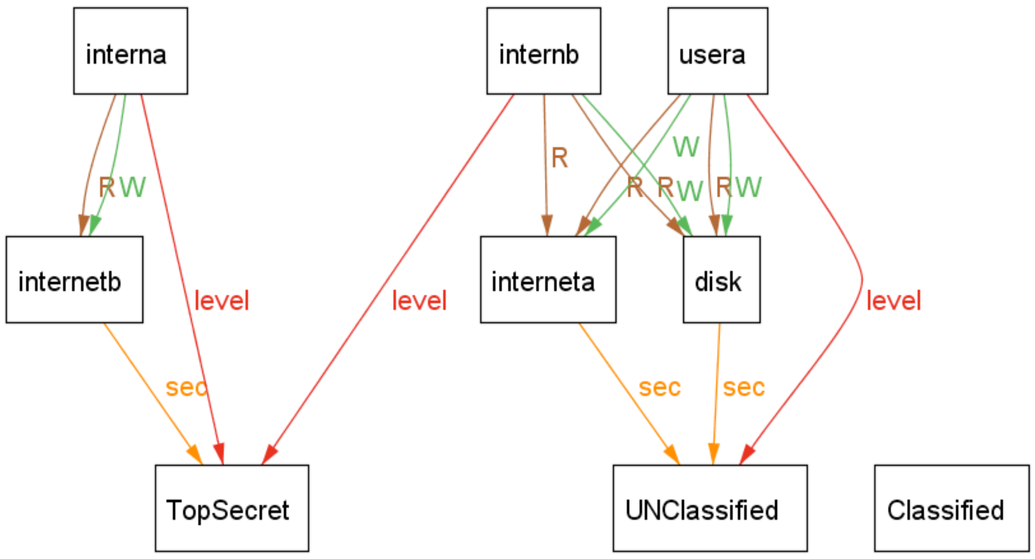

Fig. *7. Counterexample to the policy*

6.3. Separation of Concerns

Prior to implementing separation of concerns policies, we have added to our idealized system an Internet resource that can be shared amongst users. In addition, we have allowed multiple instances of disk resources.

We extend the running example to include certain separation rules. Such rules prevent users from concurrent access to certain resources. For example, interns who have *read* disk access may be denied from having *write* rights to the Internet. In our method, all resources are represented as objects, and the Internet is no exception.

```
fact SOCPolicy {
 internet in intern.W          //write access to Internet
  no disk & intern.R          // read access from disk
}
```

The code listed above specifies the separation of concerns policy. It reads that an i*ntern* cannot have read access to disk while having write access to the internet.

6.4. Role Based Access

Now we add the concept of roles to our existing mix of models. Hence, we will have BLP, SOC, and RBAC meta policies in the same model. We also extend the model by adding several business policies.

The combined model including the BLP, SOC, and RBAC properties in addition to the business policies defines the following roles: *employee, manager, director, VP, CTO, CFO, CPO, and President* in an enterprise. Three classifications of secrecy remain: *UNClassified, Classified, TopSecret.* A *User* can have only one role. In addition we add several user policies. HR files are a disk resource at the Classified level. Client files are another disk resource at the Classified level as well.

The security and privacy objects are at the Top-secret level. Furthermore, the Internet resource is at the Unclassified level.

To test if the meta-policies prevail over potential business policies, we try to assert certain rules that are known to be a violation of the meta-model. For example the policy specified in Alloy below stipulates that it should be possible for an intern to have read access to security object, an object at the secret level in a potential instance model.

```
assert businesspolicy3{
   !security in intern.R
}
```

Running the above assertion shows that it is not possible to have such a business policy. The analyser rejects the above assertion, as shown in Figure 8.

```
Executing "Check businesspolicy3 for 6"
Solver=sat4j Bitwidth=4 MaxSeq=6 SkolemDepth=1 Symmetry=20, 1873 vars.
256 primary vars. 3031 clauses. 19ms.
No instance found. 7ms.
```

***Fig. 8.** Consistency analyser response*

In the Alloy code below, we check if an intern can assume a president's role.

```
assert userpolicy4{
!interna.A = President
}
```

There is no rule preventing an intern from assuming the role of president.
Hence, Figure 8 shows that Alloy was not able to produce a counterexample.

```
Executing "Check userpolicy4"
Solver=sat4j Bitwidth=4 MaxSeq=4 SkolemDepth=1 Symmetry=20, 619 vars.
100 primary vars. 958 clauses. 8ms.
 No counterexample found. Assertion may be valid. 14ms.
```

***Fig. 9**. Consistency analyser response*

Notice that the speed of checking these assertions is quite high. The speed of execution is in the order of milliseconds, however these models are not representative of potentially complex enterprise networks.

7. Related Work

Integration of models is not a new research topic. Existing work on combining secrecy models concentrated on re-implementing models is presented in [26]. Re-implementing or re-writing models is not the same problem as integration. In the former approach, i.e. re-writing, one of the models ceases to exist, whereas in the latter both models are translated into a common representation. We propose first order logic for the common representation.

Formal methods are commonly used for verification of security systems as well as for interaction detection [16]. For example in [15], the authors suggest a requirement driven approach to the design and verification of web services. They also suggest the use of model checkers to verify system constraints. In the security and privacy domain [17] studies a methodology for reasoning and verifying policies.

This paper is a continuation of [18] where we proposed the use of logic for feature interaction detection. In other previous work we have shown the ability of validating access control policies using Alloy in [19]. Address modeling and verification of XACML policies by using Alloy was discussed in [20]. In [21] we presented the ability of to represent legal requirements using logic, this work was inspired by the position argued in [22].

8. Conclusions and Future Work

Governance requirements representing legal and enterprise specifications may require the integration of secrecy models and business policies. This integration may be necessary because of the need to integrate networks including tools and operating systems. Unfortunately, these tools may have multiple secrecy models and business policies that can be found conflicting. This paper shows a method for representing mixed secrecy models including enterprise business policies. It shows how a secrecy system designer can validate consistency using a two step verification process: Model and System consistency checking. The paper provides a method including the definitions, the process, and the logic representation that are needed to understand the method.Technically, we have demonstrated the use of a logic based approach for the detection of dangerous inconsistencies in mixed model security systems.

The paper starts by presenting the formal process of defining an abstract model of governance requirements in a given environment. We then show several examples showing inconsistencies between two types of multi-level secrecy policies. In addition we show inconsistencies between business policies and the mixed model policies.

We have used a first order representation of BLP, RBAC, and Separation of concerns policies, and show some results that can be obtained by analyzing combined policies using our approach. The policy completeness problem was not addressed in this paper, and is the subject of future work.

We believe that formal consistency checking at the design stage is of interest to organisations intending to build systems with multiple secrecy policies. We have shown that this formal approach can detect potential policy interactions and simulate policy violations prior to empirical testing done after the implementation. Faulty model and policy combinations can be detected in this way.

Another equally important aspect, also the subject for future research, is to explore a language meant to represent secrecy models. A security designer may specify a particular security model and test its potential inconsistencies with other models or interactions with user policies. The language can be translated to a logic analyser language for formal analysis. This future language can be benefit from the logic models used in this paper.

Acknowledgment: This research was funded in part by grants of the Natural Sciences and Engineering Research Council of Canada.

References

[1] Nokia Multiple security domains, Nokia, http://www.nokia-asia.com/NOKIA_BUSINESS_26/Europe/Products/Security_Products/sidebars/pdfs/nokiamds_datasheet_emea.pdf , Accessed April 2009.

[2] Windows Vista Security and Data Protection Improvements. Microsoft. http://technet.microsoft.com/en-us/library/cc507844.aspx, Accessed April 2009.

[3] P. Chandrasiri et al, "Personal Security Domains." Contribution to the 10th Wireless World Research Forum(WWRF), New York, October 27-28, 2003.

[4] Matushima, R., Venturini, Y. R., Sakuragui, R. R., Carvalho, T. C., Ruggiero, W. V., Naslund, M., and Pourzandi, M. 2006. Multiple personal security domains. In Proceedings of the 2006 international Conference on Wireless Communications and Mobile Computing (Vancouver, British Columbia, Canada, July 03 - 06, 2006). IWCMC '06. ACM, New York, NY, 361-366. DOI= http://doi.acm.org/10.1145/1143549.1143621

[5] W. R. Ford. 1995. Administration in a multiple policy/domain environment: the administration and melding of disparate policies. In Proceedings of the 1995 Workshop on

New Security Paradigms (La Jolla, California, United States, August 22 - 25, 1995). New Security Paradigms Workshop. IEEE Computer Society, Washington, DC, 42-52.

[6] L. Ninghui, J. Feigenbaum , B. Grosof, A Logic-based Knowledge Representation for Authorization with Delegation, Proceedings of the 1999 IEEE Computer Security Foundations Workshop, p.162, June 28-30, 1999.

[7] P. Bonatti, S. De Capitani di Vimercati, and P. Samarati. 2002. An algebra for composing access control policies. ACM Trans. Inf. Syst. Secur. 5, 1 (Feb. 2002), 1-35. DOI= http://doi.acm.org/10.1145/504909.504910.

[8] C. E. Landwehr. Formal models for computer security. ACM Computing Surveys, 13(3):247–278, 1981.

[9] Information networking: networking technologies for broadband and mobile networks : international conference ICOIN 2004, Busan, Korea, February 18-20, 2004 : revised selected papers By Hyun-Kook Kahng, Shigeki Goto, Han'guk Chŏngbo Kwahakhoe, Jōhō Shori Gakkai (Japan) Edition: illustrated Published by Springer, 2004.

[10] Computer security--ESORICS 2002: 7th European Symposium on Research in Computer Security, Zurich, Switzerland, October 14-16, 2002 : proceedings By Dieter Gollmann, Günter Karjoth, Michael Waidner Edition: illustrated Published by Springer, 2002 ISBN 3540443452, 9783540443452

[11] J. Fjardo, E. Dustin. Testing SAP R/3: A Manager's Step-by-Step Guide. Published by Wiley-Interscience, 2007 ISBN 0470135484, 9780470135488.

[12] M. Evers . An analysis of the requirements for DSS on integrated river basin management. , Management of Environmental Quality: An International Journal, Year: 2008 Volume: 19 Issue: 1 Page: 37 - 53 DOI: 10.1108/14777830810840354.

[13] W. Hassan. Validating Legal Compliance- Governance Analysis Method. PhD. Thesis. Submitted 2009.

[14] P. Samarati, and S. De Capitani di Vimercati, " Access Control: Policies, Models, and Mechanisms," in Foundations of Security Analysis and Design, R. Focardi, and R. Gorrieri (eds.), Springer-Verlag, 2001

[15] M. Pistore, M. Roveri, P. Busetta. Requirements-Driven Verification of Web services. Electronic Notes in Theoretical Computer Science. To appear.

[16] W. Hassan, L. Logrippo. Governance Policies for Privacy Access Control and their Interactions. In: S. Reiff-Marganiec, M. Ryan. Feature Interactions in Telecommunications and Software Systems VIII, ICFI'05, IOS Press, 114-130.

[17] L. Cholvy, F. Cuppens. Analyzing Consistency of Security Policies, 18th IEEE Computer Society Symposium on Research in Security and Privacy.

[18] W. Hassan, L. Logrippo. Interactions among secrecy models. ICFI 2009 10th International Conference on Feature Interactions in Telecommunications and Software Systems. To Appear

[19] W. Hassan, L. Logrippo, M. Mankai: Validating Access Control Policies with Alloy. Appeared in: K.Adi, L.Logrippo, M. Mejri. Proceedings of a Workshop on Practice and Theory of Access Control Technologies (WPTACT 2005), Montréal, Jan. 2005, 17-22.

[20] M. Mankai, L. Logrippo. Access Control Policies: Modeling and Validation. Appeared in: K. Adi, D. Amyot, L. Logrippo - Proceedings of the 5th NOTERE Conference, Gatineau, Canada, August 2005. p 85-91.

[21] W. Hassan, L. Logrippo. Requirements and Compliance in Legal Systems: a Logic Approach. In Proc. IEEE 16th International Requirements Engineering Conference (RE'08): RELAW Workshop. Barcelona, Spain. Sep. 2008. (Electronic proceedings, 5 pages).

[22] L. Logrippo. Normative Systems: the Meeting Point between Jurisprudence and Information Technology? In: H. Fujita, D. Pisanelli (Eds.): New Trends in Software Methodologies, Tools and Techniques – Proc. of the 6th SoMeT_07. IOS Press, 2007, 343-354.

[23] D. F. C. Brewer and M. J. Nash. The Chinese Wall security policy. In Proc. of the IEEE Symposium on Research in Security and Privacy, 206-214, Oakland, California, May 1989. Also available at http://www.gammassl.co.uk/topics/chinesewall.html.

[24] Ford, W. R. 1995. Administration in a multiple policy/domain environment: the administration and melding of disparate policies. In Proceedings of the 1995 Workshop on New Security Paradigms (La Jolla, California, United States, August 22 - 25, 1995). New Security Paradigms Workshop. IEEE Computer Society, Washington, DC, 42-52

[25] Abadi, M. and Lamport, L. 1993. Composing specifications. ACM Trans. Program. Lang. Syst. 15, 1 (Jan. 1993), 73-132. DOI= http://doi.acm.org/10.1145/151646.151649

[26] J. Zao, H. Wee, Jonathan Chu, Daniel Jackson, RBAC Schema Verification Using Lightweight, Formal Model and Constraint Analysis, SACMAT 2003.

[27] M. Conover. Analysis of the Windows Vista security model. Available at www.symantec.com/avcenter/reference/Windows_Vista_Security_Model_Analysis.pdf