MODEL BASED EVALUATION OF ACCESS CONTROL

Dániel TÓTH
Advisor: András PATARICZA

I. Introduction

Access control is a factor with especially high associated risk in infrastructures where the initial design usually do not take security constraints into account and access control is developed in an evolutionary manner. In such systems further applications are frequently added to the pool of programs acting as the functional blocks of a large scale integrated application without any supervision of the potential dangers and risks of using outdated access control schemes.

Recently both regulatory compliance requirements and the growing need of enterprises to comply to practical system management standards like ITIL, have started enforcing certain access control policy properties over data and their access methods.

The existing formal requirement systems such as Bell-LaPadula [1] for confidentiality, Biba [2], Clark-Wilson[3], or the recently published Shankar-Jaeger-Sailer [4] can be incorporated into high level security policies of an organization. Such formalism can enable automated security auditing, however this is an emerging problem that needs new approaches and tools.

II. Practical challenges of Access Control evaluation

A. Heterogenity

The evolutionary development of the software used in IT infrastructures has lead to the existence of various different, often incompatible access control schemes. Large scale enterprise infrastructures are typically heterogeneous in the software involved. This problem manifests as the heterogenity of operating system as well as the interaction between the application and operating system level access controls. Clearly any comprehensive means of analysis must treat the entire deployment as a whole, otherwise the disregard of possible interactions could result in missed access paths.

B. Scalability

The size of a typical enterprise deployment in the number of both the involved computer systems and the users presents another challenge for comprehensive analysis. The size of the Access Control Matrix is proportional to the number of protected objects by the number of actors. It can be seen that traditional formal analysis algorythms that rely on computing the access matrices become unfeasible when dealing with a large complex deployment.

III. Model-based approach

In this article I propose a model-based approach for the development of access control evaluation tools that deal with heterogeneous systems as well as some techniques that can improve the handling of large data sets.

The basic concepts of OMG's Model Driven Architecture (formulating the requirements over an abstract platform-independent metamodel (PIM) while the exact implementation details specific to a target platform are represented by instances of platform-specific metamodels (PSM)) are key elements of this approach.

However, MDA's top-down design process has to be substituted with a bottom-up procedure in the context of analyzing existing systems, simply, because the existing systems are not engineered with

the model-based concept in mind, moreover they were subject to evolutionary development. Note that the following process outline is generally applicable to any kind of consolidation problem, not just for access controls:

1. Identitfy concepts and evaluation semantics of existing systems to formulate platform specific metamodels.
2. Synthetize the platform metamodels into a common Platform Independent Metamodel.
3. Implement a means of discovery for instantiating a platform specific model precisely representing the actual deployment.
4. Implement a transformation that will abstract the discovered platform specific models into an instance of the PIM.
5. Carry out formal analysis on the PIM.

A. Analysis of existing systems

As part of this research the access control subsystems of the following operating systems were analysed and metamodelled:

- Linux with standard POSIX filesystem level access
- Linux with sudo access control
- Linux with optional ACL extensions
- Windows Server 2003 [5]
- VMWare ESX Server virtualization platform

Each platform implements a more or less complete Role-Based Access Control scheme (RBAC)[6]. This manifests in the ability to assign actors into groups or roles, thereby giving rise to actor hierarchy.

A common feature is ability to "impersonate", i.e. to change the effective user identifier during execution, ultimately operating on behalf of another actor. Each platform implements this differently. This feature is dynamic making it is particularly hard to analyse, as the internal execution flow of the applications making use of this feature needs to be traced. Static analysis can only make either optimistic or pessimistic assumptions.

Another common feature among ACL supporting platfroms is the possibility of permission inheritance, which is very important since it allows the administrator to reduce the number of necessary explicite access control definitions. Each platform has its own complex ACL evaluation algorythm for handling such implied permissions.

B. Defining the Platform Independent Metamodel

Carrying out analyses on installations of complex heterogeneous systems is only possible if the incoherently structured (instances of different metamodels) deployment information is converted to a uniform representation. There is a choice between two substantially different kinds of platform independent metamodels.

In the "vertical" approach, the PIM needs to capture only common, basic concepts (primitives) of the problem domain. This will result in a flattened model instance (similar to the access matrix formalism) that is straightforward to formally analyse but too large to handle. For instance, all implied permissions become explicitely represented in this approach.

The "horizontal" approach on the other hand contains all the platform specific metamodels as a union, thus creating a common metamodel merging the platform specific metamodels into a single notion. In this case the concrete model can be a precise equivalent of the configuration of the system component under analysis, so the exact deployment can be reproduced from the model. Obviously, this

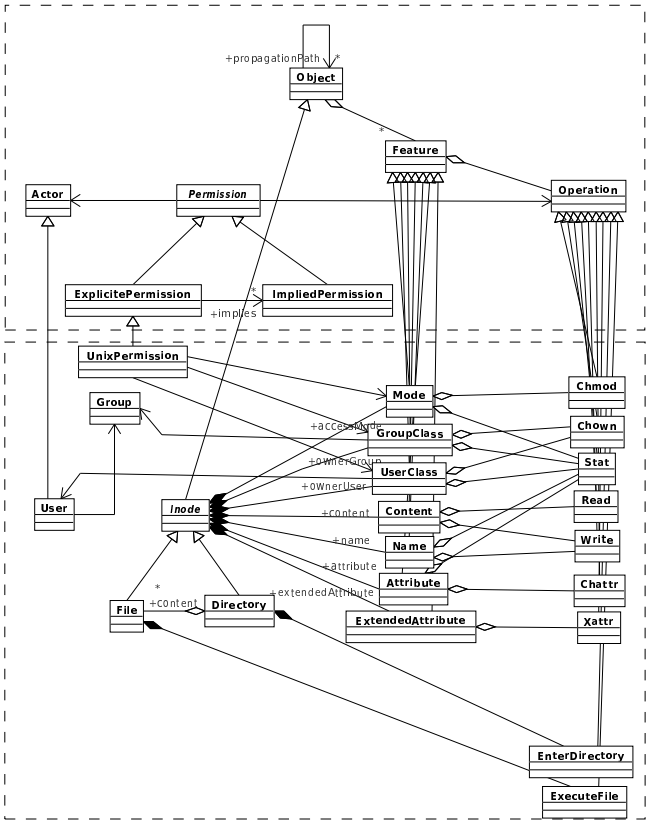

Figure 1: The platform independent metamodel (above) with POSIX platform specific metamodel as an example (below)

extreme will result in a relatively compact model instance, but any formal analysis will be troublesome to develop as the details of each platform will have to be dealt with.

The two approaches can be combined by realizing that the origin of complicated, diverse access modes (read, write, change permission, change ownership etc.) is simply the consequence of the insufficiently detailed modelling of protected objects. File system entries for example are not atomic entities but objects with separate attributes. By modeling the various attributes (including the permission attributes themselves) separately, all access modes can be mapped to simple read, write and execute operation primitives. This way a platform independent core metamodel can be devised that serves a generalization for all platform specific ones.

C. Semantic modeling

Remaining issues of the above described generalization concept are:

- Handling access to attributes that define permissions themselves
- Actor hierarchy
- Efficient handling of implied (inherited) permissions

It is desireable from the scalability standpoint to avoid explicitely instantiating the implicite permissions described. It should also be noted that all three outlined issues are similar in nature, they all describe implied permissions that can only be evaluated by simultaneously testing multiple conditions on different model elements separated by non trivial, recursive navigational patterns.

It is important to point out that each platform uses different semantics for implied permissions. Implementing such complex evaluators for heterogenous systems in traditional imperative programming languages is an error-prone task. An important part of this research is the investigation of using recursive pattern matching in addressing the above semantics. Until now, there were no specific requirements for the modeling language, however this approach proposes the use of a graph transformation

framework with efficient recursive pattern matching capabilities, such as VIATRA2 R3 [7].

This way the actual access evaluation becomes a pattern matching problem. Graph patterns can be considered as a formal way of defining the access evaluation procedure. The development process thus includes not only the platform specific metamodel definition but also the need to define a supplementary pattern library for each platform. These platform specific pattern libraries can then be combined into a generic access evaluation pattern that serves as a common interface for all platforms.

Similarly the formal requirements, such as the no read-up, no write-down rules of Bell-LaPadula and similar rules of Biba integrity model can also be defined as patterns. These patterns are also generic, not directly refering to the actual model elements but calling the generic access permission evaluation patterns instead. This way the same formal analysis becomes available for all platforms without additional effort.

IV. Proof of Concept

An early proof of concept implementation utilized IBM Tivoli CCMDB for discovering and storing the access control and identity information for Linux systems. The configuration data was then parsed and instantiated as EMF models. The platform independent core metamodel was also defined in EMF along with transformation and formal analysis code written Java. The output of the formal analysis is an audit report model that traces all possible violations of the formal requirements. Although this implementation is functional it required significant development effort to support just a single platform, which showed the drawback of this approach.

By utilizing the EMF integration facility of VIATRA, the EMF access control models were adapted to the VIATRA framework. At the time of writing, the integration of multiple target platforms is being developed using the described recursive graph matching approach.

V. Conclusion and further work

In this paper, I outlined the basic development process and key considerations for model-based evaluation of heterogeneous access control deployments. The most important direction of future work is to test the pattern matching-based implementation on actual deployments to find out its practical limits. Incorporating clustering of protected objects with similar permissions will also be investigated to bring further reduction in the resource requirements. Another important area to investigate is the recently developed incremental pattern matching facility of VIATRA2. Finally implementing more fine-grained formal analyses such Clark-Wilson are considered for future work. Such formal requirement systems involve the opertional analysis of the applications which greatly broadens the scope of this research.

References

[1] D. E. Bell and L. J. LaPadula, "Secure computer systems: Mathematical foundations," *MITRE Technical Report 2547*, I., Mar. 1973.

[2] K. J. Biba, "Integrity considerations for secure computer systems," *MTR-3153, The Mitre Corporation*, Apr. 1977.

[3] D. Clark and D. Wilson, "A comparison of commercial and military computer security policies," in *IEEE Symposium on Computer Security and Privacy*, Apr. 1987.

[4] T. J. U. Shankar and R. Sailer, "Toward automated information-flow integrity verification for security-critical applications," in *2006 Network and Distributed Systems Security Symposium*, pp. 267–280, 2006.

[5] *Microsoft Windows Internals, Fourth Edition: Microsoft Windows Server 2003, Windows XP and Windows2000*.

[6] R. K. R. Sandhu, D. Ferraiolo, "The nist model for role-based access control: Towards a unified standard," in *5th ACM Workshop on Role-Based Access Control*, pp. 47–63. ACM Press, 2000.

[7] G. V. Á. Horváth and D. Varró, "Generic search plans for matching advanced graph patterns," in *Sixth International Workshop on Graph Transformation and Visual Modeling Techniques (GT-VMT 2007)*, pp. 57–68, 2007.