Reality vs. Security Model vs. Software – Bridging the Gaps
Extended Abstract

Daniel Pähler

Institute for Information Systems Research, University of Koblenz-Landau, Germany
tulkas@uni-koblenz.de

Introduction

In [4], it was shown that existing formalizations of Digital Rights Management share some common weaknesses, for instance their vendor-centric point of view or the fact that they only distinguish between "legal" and "illegal", without allowing for a "gray area". A list of requirements for a new DRM model was derived, and a formal model was introduced that fulfills these requirements. This formal model, which has in the meanwhile been named "Formosa" (Formal Model for Secure Actions), follows in the tradition of other IT security models, as described by Grimm [1]. It specifies secure system states and rules for allowed state transitions which, if adhered to, aim to fulfill the superior security objective that each actor should be able to subjectively feel secure. Also similar to other IT security models, Formosa is not just meant to describe an existing situation, but to provide a sound basis for implementations that fulfill the same security objective. Therefore, the work on Formosa uses the research method of "Design Science Research" [5].

Reality versus Security Model

This section lists the challenges that have turned up in the process of creating Formosa on the basis of assumptions about the real world.

The general problem of modeling

In IT security models as well as in every other type of model, one central aim is to reduce complexity through abstraction. Inevitably, aspects of the original scenario have to be left out, and it can be difficult to decide which aspects these should be. Particularly during the first steps of creating a formal model, a trade-off is often necessary. On one hand, each feature that is added could turn out to be useful or even necessary later. On the other hand, each additional feature also makes the model more complex and might violate the scientific principle of "Occam's Razor"[1]. In an informal model, e.g. in a UML class diagram, changes

[1] As Heylighen explains in [2], "[Occam's Razor] admonishes us to choose from a set of otherwise equivalent models of a given phenomenon the simplest one."

might be easy to perform later in the development process; classes can be deleted or added, and it is rather easy to see which other classes are influenced by a change. In Formosa, there are numerous definitions that build on each other, and if one of the basic definitions is changed, each other definition has to be checked and possibly changed, too.

At first, ODRL 2.0 could be used as a kind of template for Formosa. It was considered that if entities such as assets, actors, permissions or duties could be used to describe usage control in real-world uses cases, they would provide a good foundation for a formal model. But since ODRL 2.0 itself is not formal and cannot reasonably be formalized (as pointed out in [4]), some features had to be left out. As a concrete example, Formosa was first created without a notion of time to avoid making an already complex model even more complex. Only lately was the decision made to include time so that temporally limited rights expressions and duty deadlines can be expressed. The downside is that Formosa is now even more difficult to understand than it was before.

Choosing an adequate notation

In order for Formosa to be both comprehensible and accurate, an appropriate notation has to be used. The current notation, which mainly uses mathematical set expressions, has the advantage that it allows for precise expressions and is still relatively easy to read for researchers in the area of computer science. But it is unclear if there exist other notations that could allow for expressions which are easier to read and write, or which could even be interpreted by a computer. The latter would not only help unveil errors in the model, it could also help when creating an implementation of the model.

This challenge is currently being worked on in a master's thesis. The student will first try to get a broad overview of modeling methods (and thus, notations) used in the area of computer science and group them into clusters of similar methods. In the next step, he will analyze the aptness of some methods for the creation of IT security models in general and Formosa specifically, pointing out the advantages and disadvantages of each method. Eventually, this thesis' goal is to aid in finding the ideal notation.

Checking the real world assumptions

One fundamental innovation of Formosa, when compared to other IT security models, is the fact that it allows for a "legal gray area". Actors might perform illegal actions and still be in a legal state (if they are still able to pay a fine for their actions). In this regard, Formosa was created to reflect user behavior more realistically. It is assumed that many users behave "a little illegally" without feeling guilty about it, and that a DRM model should therefore distinguish between minor misdemeanors and actual crimes.

But the question arises, "Is this a valid assumption?". The facts that many legal systems also make this distinction, and that trivial offenses are often considered socially acceptable point to "yes", but can this be proven?

It is the goal of a master's thesis that was recently started to find an answer to the above questions. The student's method is that of a literature analysis. Several studies about users' attitudes toward DRM and DRM-related problems exist and should provide a good data basis for the analysis.

Security Model versus Software

When an IT security model is stable enough, it is desirable to transfer it into an implementation. In terms of the Design Science Research method, this step is needed for (and is already part of) the evaluation of the artifact created before. For Formosa, its similarity to the concept of Usage Rights Management (URM) [3] makes the Java-based URM implementation "TURM" (Toolkit for URM) a particularly attractive platform for an implementation. TURM can already deal with ODRL licenses as well as assets, and it tries to help the user keep an overview of their legal situation. But it turns out that the transition from the model to software is not trivial. The challenges outlined below are currently being worked on in a master's thesis that aims to implement Formosa in TURM.

Features that were left out in Formosa

As was shown above, several features were deliberately left out in Formosa to avoid making it too complex. But in object-oriented software such as TURM, these features can more easily be dealt with, particularly those that are already part of TURM. For instance, Formosa has no notion of count constraints (e.g., "Alice is allowed to send the file xyz.mp3 to at most 3 other persons."), whereas TURM does support them to a certain degree. In how far can the TURM implementation of Formosa make use of these features and still be an accurate representation of the model?

"Open" definitions in Formosa

In order to deal with real-world situations where potentially unlimited amounts of entities can occur, Formosa uses a number of "open" definitions. This means that some sets like *Actors* are defined to include all actors, without listing them explicitly. Other sets like *Actiontypes* are defined with some exemplary elements $(use, copy, buy, ...)$, but are meant to be extended for specific use cases. Finally, many functions are defined as so-called "oracle functions": they cannot actually compute output values for specific inputs, but they have to use lookup tables to find the right output for the respective input (e.g., *cost* returns the cost of a specific action). Obviously, these lookup tables also have to be defined depending on the specific use cases. Since the implementation cannot work with Formosa's open definitions, it has to be configured to use concrete values. But where do these values come from, and who should be able to configure them?

Controllability and observability

As was pointed out in [4], DRM systems often have the problem that in order to be effective, they need to have control over the user's domain. Only if it can be assured that a user has no means of circumventing a restrictive DRMS does it make sense to use this DRMS (hence, assets are usually encrypted).

URM as well as Formosa follow a different approach: they do not use any rights enforcement measures and give the user the freedom to decide for themselves whether they want to behave legally or not. One advantage of this is the fact that users can use assets managed by TURM with the same software that they would use for unmanaged assets. For Formosa's implementation, this turns out to be a problem. Even though Formosa does not necessarily *control* actions, it needs to *observe* them. In the purely theoretical domain of Formosa as an IT security model, it is defined that each action can lead to a state change. But in the implementation, actions that cannot be observed obviously cannot lead to state changes.

A first approach to solve this problem includes a demon process which is always running and which can record certain events in the system; for other events, the user has to "manually inform" Formosa. It is still an open question how useful this approach can be in practice.

References

1. Rüdiger Grimm. A Formal IT-Security Model for a Weak Fair-Exchange Cooperation with Non-Repudiation Proofs. In *SECURWARE 2009, The Third International Conference on Emerging Security Information, Systems and Technologies, Athens, 18-23 June 2009*. IEEE Computer Society Press, 06 2009.
2. Francis Heylighen. Occam's Razor. *Principia cybernetica web*, 07 1997.
3. Helge Hundacker, Daniel Pähler, and Rüdiger Grimm. URM – Usage Rights Management. In Jürgen Nützel and Alapan Arnap, editors, *Virtual goods 2009*, Nancy, France, 09 2009.
4. Daniel Pähler and Rüdiger Grimm. A formal Digital Rights Model without Enforcement. In *Virtual Goods 2011*, 2011.
5. Vijay Vaishnavi and Bill Kuechler. Design Science Research in Information Systems. website, 01 2004. last updated September 30, 2011.