Short Paper: A Modal Framework for Security Properties

Matvey Soloviev Musard Balliu Roberto Guanciale

KTH Royal Institute of Technology
{matvey,musard,robertog}@kth.se

I. Introduction

The study of computer security is concerned with guaranteeing that computer systems maintain some desirable properties in the face of one or more attackers that seek to subvert them by way of some set of actions and observations available to each of them. To formalise and streamline this process, various particular *security properties*, ranging from basic ones such as confidentiality and integrity [4] to more involved ones such as robust declassification [7] and nonmalleability [2], have been defined and studied. These properties can be instantiated as building blocks to define an overall *security policy*, that is, a description of desirable properties that we wish to be maintained against an attacker with well-defined capabilities.

Generally, these properties are defined with respect to particular system models and formalisms, which are often tailor-made to the problem at hand. These different approaches make it hard to compare the differences between various definitions, and often obscure subtle properties that determine their interpretation in counterintuitive scenarios, such as ones involving nontermination, asynchrony or unusual sets of attacker capabilities. To address this problem, we propose a new framework for reasoning about security that is based on *modal logic* [1], [3], an extension of propositional logic that enables reasoning about the interaction of time, choices and possibility with the truth and falsity of formulae. Formulae of modal logic are evaluated with respect to a *Kripke frame*, which encodes all possible scenarios that may be relevant in the domain of discourse.

We introduce a way of encoding the security-relevant behaviour of a system and the potential attackers interacting with it as a special *security Kripke frame*. Trace-based program semantics, as often used in the computer security literature, can be converted into this representation. Having represented programs in this way, we can show how the popular security properties of confidentiality, integrity and robust declassification can be represented as modal logic formulae that must hold for the security Kripke frame representing a system. All these notions can be shown equivalent to their usual runs-and-systems definition under the given conversion. In the case of robust declassification, we find that our approach reveals a subtlety about the way that the accepted definition deals with nontermination and unbounded delays in asynchronous systems, and suggest an alternative modal logic formula that defines a similar property, which may be more appropriate in many scenarios. Another advantage of security Kripke frames is that while program semantics can be converted into them straightforwardly, they also lend themselves to manual representation of intuitive abstract scenarios, which can aid in understanding various security properties.

II. Definitions

We work in a standard multimodal logic as in [3] obtained by completing a propositional logic of system state with two modalities (as in [6]) for each relation R. If φ is a valid formula, then so is $[R]\varphi$ and $\langle R\rangle\varphi$. Modal logics are typically interpreted with respect to a *Kripke frame*, which consists of a set of *possible worlds* W and relations $R \subseteq W \times W$, corresponding to the relations that may occur in the formula. For each world $w \in W$, we assign a truth value to basic propositional statements φ by an entailment relation $w \vDash \varphi$, and we say $w \vDash [R]\varphi$ iff for all w' such that $(w, w') \in R$, $w' \vDash \varphi$, and $w \vDash \langle R\rangle\varphi$ iff $w \vDash \neg[R]\neg\varphi$.

We want to define a special class of Kripke frames in which we can reason about security. As is standard, we take the set of possible worlds to be complete descriptions of the possible states the system could be in at a given point in time. To relate them to each other, we then introduce relations that relate worlds if one can evolve into the other over time, and relations that capture the *capabilities* (what they *can* do) and *permissions* (what we wish to *allow* them to achieve) for each participant agent of the system.

Definition II.1. A *security Kripke frame* over a set of worlds W and agents A is a frame equipped with

- a transitive, reflexive time relation T;
- for each agent A,
  - an equivalence relation K_A, which relates two worlds if A can not distinguish them.
  - a transitive, reflexive relation W_A, which relates two worlds (w_1, w_2) if A can perform an action at w_1 to change it into w_2.
  - a transitive, reflexive relation P_A, which relates two worlds (w_1, w_2) if A would be permitted to let the system be in state w_2 when in reality it is in state w_1.

For the sake of following a common convention, we write $\Box$ for $[T]$ and $\Diamond$ for $\langle T\rangle$. We often care about formulae that say that a particular event occurred in a run, and thus remain true forever if they become true once, so for all worlds w, $w \vDash \varphi$ iff $w \vDash \Box\varphi$. Such formulae are called *temporally sound* (t.s.).

Typical runs-and-systems settings from the program security literature [3], [7], [2] can be embedded in this framework, though the details are beyond the scope of this extended abstract.

Proposition II.2. *A computer system specified by a set of possible initial states, a program and small-step semantics, which operates on memory locations that take labels from a security lattice, can be represented as a security Kripke frame.*

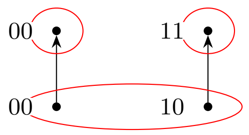
III. Examples of security properties

In this section, we show some examples of security properties that can be captured in our framework. The most basic example is the notion of *confidentiality*, interpreted as saying that no secret information is revealed by the system. In a setting which is deterministic and where what each agent *can* see is identical with what they are *permitted* to see, this is equivalent to saying that no new knowledge is gained: to the extent that an agent can't predict some knowledge of the system that they will have in the future, this must be because the fact they will know depends on something the agent doesn't currently know, and thus equivalently isn't allowed to know. We thus arrive at the following definition:

Definition III.1. A security Kripke frame satisfies *confidentiality* if for all agents A, worlds w and t.s. formulae φ,

$$w \vDash \Diamond[K_A]\varphi \Rightarrow [K_A]\Diamond\varphi.$$

We can illustrate a typical security Kripke frame in which confidentiality is violated. Here, we take there to be two variables, of which the first one is secret (invisible to A) and the second one is public (visible to A). After one step, the system will copy the contents of the first variable into the second one, revealing the secret to A. The red ellipses depict the equivalence classes of worlds under K_A.



As we can see, at the bottom left world, A does not know that the secret equals 0 (as it can't distinguish that world from the bottom-right one, where the secret equals 1), but will eventually come to know. A similarly simple security property says that an untrusted agent may not influence trusted memory by setting the initial values of memory locations they control. Literally, our definition says that if by performing an action A is capable of and waiting for its effects to propagate, an effect φ can be achieved, then A must be permitted to directly bring about φ at some point in the future.

Definition III.2. A security Kripke frame satisfies *integrity* if for all agents A, worlds w and t.s. formulae φ,

$$w \vDash \langle W_A\rangle\Diamond\varphi \Rightarrow \Diamond\langle P_A\rangle\varphi.$$

Both confidentiality and integrity are often considered too restrictive for real applications. A looser security property was therefore introduced in [7] and refined further in [5], [2], which says that a system may release secrets, but an untrusted agent must not be able to influence whether a secret is released. Here, it turns out that the most natural definition that we make happens to differ from the one in [2] in a crucial way when dealing with asynchronous systems: if a system leaks a secret in some cases and loops forever without doing anything in others, but an attacker can influence it to reliably output the secret, then this is considered a violation of the first definition below but not the second. Another similar counterexample arises when the leak may be delayed by an unboundedly large amount of time, so there is an infinite number of potential runs in which the delay is arbitrary long. The reason for this discrepancy is that definitions as in [2] compare infinite traces generated by non-terminating programs, functionally modelling the attacker as having knowledge that it can't actually get at any finite point in time. The easiest way to capture this is to add additional worlds corresponding to *infinite* traces, reachable by a special relation T_ω from any finite trace in the same run, and only K_A-related among themselves. Notions such as "A knows eventually" are thus stronger with $\langle T_\omega\rangle[K_A]$ than with $\Diamond[K_A]$, as they also cover knowledge gained "after an infinite number of steps".

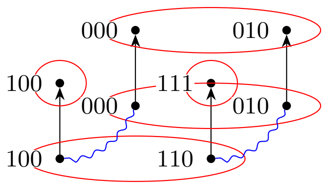
Definition III.3. A Kripke frame satisfies *robust declassification* if and only if for all worlds w, all agents A and all t.s. formulae φ,

$$\langle W_A\rangle(\Diamond[K_A]\varphi \wedge \neg[K_A]\Diamond\varphi) \\ \Rightarrow \Diamond[K_A]\varphi \wedge \neg[K_A]\Diamond\varphi.$$

It satisfies *infinite r.d.* iff

$$\langle W_A\rangle(\langle T_\omega\rangle[K_A]\varphi \wedge \neg[K_A]\langle T_\omega\rangle\varphi) \\ \Rightarrow \langle T_\omega\rangle[K_A]\varphi \wedge \neg[K_A]\langle T_\omega\rangle\varphi.$$

This definition intuitively says that if by performing an action ($\langle W_A\rangle$), A can bring about a violation of confidentiality, then confidentiality of φ is violated regardless. Taking u to be untrusted, p to be public and s secret, while e.g. the program $p := s$ should satisfy this property, the program if $u = 1$ then $p := s$ is an archetypal example of one that violates this definition. We can illustrate it as follows, using wavy lines to denote the relation W_A which allows A to change the initial value of u:



Theorem III.4. *Confidentiality, integrity and infinite r.d. are satisfied for a security Kripke frame derived from a program on security-labelled memory if and only if the original program satisfies their runs-and-systems counterparts. Moreover, for a synchronous program inducing a finite set of possible runs, robust declassification and infinite r.d. are equivalent.*

References

[1] P. Blackburn, M. de Rijke, and Y. Venema. *Modal Logic*. Cambridge Tracts in Theoretical Computer Science, No. 53. Cambridge University Press, Cambridge, U.K., 2001.

[2] E. Cecchetti, A. C. Myers, and O. Arden. Nonmalleable information flow control. In *Proc. 2017 ACM SIGSAC Conference on Computer and Communications Security*, 2017.

[3] Ronald Fagin, Joseph Y. Halpern, Yoram Moses, and Moshe Y. Vardi. *Reasoning About Knowledge*. MIT Press, Cambridge, MA, USA, 2003.

[4] J. A. Goguen and J. Meseguer. Security policies and security models. In *Proc. IEEE Symposium on Security and Privacy*, pages 11–20. 1982.

[5] A. C. Myers, A. Sabelfeld, and S. Zdancewic. Enforcing robust declassification. In *Proc. 17th IEEE Computer Security Foundations Workshop*, pages 172–186, 2004.

[6] Vaughan R. Pratt. Dynamic logic*. In L. Jonathan Cohen, Jerzy Łoś, Helmut Pfeiffer, and Klaus-Peter Podewski, editors, *Logic, Methodology and Philosophy of Science VI*, volume 104 of *Studies in Logic and the Foundations of Mathematics*, pages 251–261. Elsevier, 1982.

[7] S. Zdancewic and A. C. Myers. Robust declassification. In *Proc. 14th IEEE Computer Security Foundations Workshop*, pages 15–23, 2001.