Chapter 23

A PROCESS ALGEBRAIC APPROACH TO SECURITY POLICIES

Peter Ryan and Ragni Ryvold Arnesen

Abstract We discuss the nature of security policies, particularly those that arise in the context of healthcare informatics, and the kind of mathematical framework needed to describe and reason about them. Various special purpose frameworks for this purpose have been presented over the years, many using bespoke logics and models of computation. We argue that the properties of interest can be expressed cleanly in a mainstream formal method, in particular in the process algebra CSP. This has a number of advantages: we have a well-established, uniform framework with well-defined semantics to work with and access to a number of well-established tools to verify and validate our models and implementations. By way of illustration we describe a CSP formulation of a policy for a clinical trials application drawn for the Framework 5 HARP Project.

Keywords: Security policy, access control policy, process algebra, CSP

1. Introduction

Increasingly there are moves to put health records in electronic form. Often, such records will be distributed and highly networked. There are clear advantages in terms of healthcare and medical research but this introduces a host of new challenges from the point of view of privacy, integrity, trust relationships etc. In this paper we describe a framework in which we can give a formal description of the privacy requirements and model security architectures and mechanisms.

A number of frameworks have been proposed to address such issues, for example the Role Based Access Control (RBAC) models and its many variants. These typically require the construction of special purpose models of computation, special logics etc. In this paper we take a fresh look at these issues and argue that special purpose frameworks are not necessary and a well-established, general purpose formal method is actually more appropriate and effective. In

The original version of this chapter was revised: The copyright line was incorrect. This has been corrected. The Erratum to this chapter is available at DOI: 10.1007/978-0-387-35697-6_26

E. Gudes et al. (eds.), *Research Directions in Data and Applications Security*

© IFIP International Federation for Information Processing 2003

particular we argue that the privacy requirements of interest in a healthcare context can be expressed in an elegant and intuitive way in the process algebra Communicating Sequential Processes (CSP) [3]. A number of advantages follow from this: we have well-defined semantics and well-developed tools and techniques for design, validation and verification at our disposal.

We start with a discussion of the characteristics of security policies in general and healthcare policies in particular. We then give some elements of CSP. This is followed by an outline of a model of a clinical trails example based on a use case developed for the European Framework 5 Project HARP [2]. We present CSP formulations of the associated security requirements. A fuller version of this paper is available at www.cs.ncl.ac.uk/research/trs/index.html.

2. Security Policies

At the most abstract level, a security policy seeks to regulate the possible behaviours of a system. Here the term "system" is used in the broad sense, not just the computer system but the human users, social and legal frameworks, system developers and so on.

In this paper, we will restrict ourselves to just the technical aspects of the system. To deal with issues of accountability, responsibility, duties, obligations, trust etc. we need to go beyond the purely technical notion of system, and deal with socio-technical mechanisms.

Some constraints are simple assertions that certain behaviours are undesirable. These are often referred to as "safety" or "trace" properties and can be formulated as simple predicates over the system behaviours. Some assert that certain behaviours must be possible, i.e. "liveness" properties. These are a little more subtle but are also well studied in the theory of concurrent systems. Some are more subtle still: asserting, for example, that the system as a whole must exhibit some property e.g., fairness or absence of certain information-flows. These are not reducible to predicates over traces and often take the form of assertions that if certain behaviours are possible then related behaviours must also be possible.

Such properties can, leaving aside certain subtleties, be readily expressed in a process algebra like CSP [5]. Their enforcement, in the sense of Schneider [8], is rather more delicate. Schneider points out that information flow is not an enforceable policy. The question of how exactly properties like information flow are enforced by, for example, access control mechanisms, is still not fully understood. This will be dscussed more fully in a future paper. For the purposes of this paper we will restrict our discussions to enforceable policies and show how these at least can be elegantly formulated in a process algebraic framework like CSP.

3. Components of a Healthcare System

Our model will be populated with principals, resources, permissions and roles. By principals we mean identified, active entities that can initiate security relevant actions. This includes users of the system, clinicians, patients etc., but can also include devices, programs, mobile agents etc. Resources include security relevant, passive objects such as records, databases, printers, CPU's and so on.

Permissions can be thought of as security relevant capabilities: for example various kinds of access to patient records. We will regard a role as being a collection of permissions (and also, where appropriate, of duties). It is generally supposed that roles correspond to functions in the corporate structure. Roles will vary from application to application but in a healthcare context we might have: Doctor, Consultant, Nurse, Patient, Administrator, and Security Officer.

At any given time the policy will assign a set of roles, and hence set of permissions, legally available to each user identity. This will typically depend on, amongst other things, the attributes assigned to the user. For a certain class of policy it may be that for any given user this set is essentially static: assigned at time zero, possibly changing occasionally as user's attributes change (promotion, new qualifications etc). For more elaborate policies we might imagine the set of permissions also being dependent on factors such as time, location, roles already adopted in simultaneous or even previous sessions and so on.

In order to interact with the system, a user will start a session by authenticating himself and then selecting a role or roles form those available.

4. Security Requirements

We give informal statements of some of the properties that appear to be relevant for health-care. More formal definitions, given in terms of CSP, will be given in the case study. Further definitions can be found in [6].

4.1 Confidentiality

Confidentiality rules determine what information is available and to whom. Rules will typically be stated in terms of roles and other constraints. For example: "A clinician logged in locally under the consultant role is allowed to read X fields of a patient's records during working hours if he is assigned to that patient."

4.2 Information Flow

Confidentiality rules, as we have defined them above, constrain what information flows are allowed from objects to users. In order to enforce privacy we also need rules constraining what information flows are allowed between

objects, similar to those found in military style policies. Information should not, ordinarily, be allowed to flow from a highly sensitive file to one of lesser sensitivity, or from one patient's records to another's. Exceptions will be allowed in special circumstances, akin to the "downgrades" of the military style policies. This suggests that there might be a useful role for MLS style policies in healthcare applications.

4.3 Integrity

Roughly speaking, integrity is about ensuring the correctness of data. Typically it is formulated in terms of who (which roles) are allowed to write data under what circumstances. We might then have rules like:

> "A clinician assigned to a patient can create a record and append information to that patient's record."

What precisely is meant by "correctness" of data is rather tricky and takes us into semantic as opposed to purely syntactic considerations. The access control approach sidesteps the semantic issues by relying on the authorised agents to maintain the accuracy of the information. The problem is related to the issue of the limits of enforceable policies: a user can commit a misdemeanour whilst remaining strictly within his permissions.

4.4 Privacy

Privacy is a complex notion and is used in many different senses. We will no go into details here but assume, for the purposes of this paper, that it is a suitable mix of confidentiality, intrgrity, anonymity requirements from the patient point of view and so does not introduce fundamentally new concepts.

4.5 Separation of Duty

Sometimes we need to stipulate additional constraints, such as mutual exclusions between permissions: a given user is allowed exercise either permissions A and B but not both. Sometimes this is to avoid conflicts of interest (as with so-called Chinese Wall policies) or to support separation of duties: certain actions can only be performed with the agreement of several roles. We might, for example, require that to alter a patient's record needs the approval of both a consultant and an administrator. By stipulating that no user can adopt both of these roles we ensure that a single user cannot alter patient records.

4.6 Anonymity

Anonymity will occasionally be called for, for example: that a patient name cannot be linked with a certain record under certain circumstances. We may want to provide patients with the possibility of obtaining advice and treatment

under an alias. In general therefore anonymity is a requirement that information and actions not be linkable to individuals.

Anonymity, like confidentiality and integrity, can be viewed a symmetry under an appropriate set of transformations of a suitable system abstraction. Thus, suppose we want to stipulate that the system provides patient anonymity with respect to medical researchers. We require that any medical researcher's view of the system be unchanged under permutations of patient identities. This is readily expressed using the renaming operator of CSP. Further details may be found in [6].

Note that anonymity is relative to a point of view. We might require clinician anonymity when the records are viewed by the public but not when viewed by, say, an auditor.

4.7 Availability

All the properties discussed above are concerned with preventing (or deterring) actions. We will also be concerned with ensuring the availability of certain actions. That is, if an action is allowed by the policy then the system implementation should ensure that it can be carried out in a timely and dependable fashion. We can think of availability as a duty of the system to provide services within a given times of a request.

5. Formalisms for Healthcare Policy

We will use the process algebra CSP in this paper to formulate our policies and requirements. CSP is an event-based formalism that appears to be well suited to our purposes. It is possible that an alternative formalism may be equally suitable, for example a state-based one like Abrial's B-method. An exact semantic equivalence has been established between CSP and B along with tool support that might be useful to move between state-based and event-based frameworks [1].

5.1 Elements of CSP

The process algebra CSP (Communicating Sequential Processes) is a well-established formal method with mature tools that has been used highly effectively in the design an analysis of concurrent, distributed systems [3]. In particular it has proved highly effective in information security applications: the analysis of security protocols and the formulation of information flow properties [5, 6].

Here we give a brief overview of the elements of CSP that we will require to construct our model of a clinical trials case study and of its security requirements. Readers wanting a fuller description of the theory underlying CSP are referred to [4, 10].

For the purposes of this paper we only need a small fragment of the syntax and semantics of CSP. A process P has an associated alphabet αP the set of events (actions) available to it. Let Σ denote the universal alphabet. For what follows, the trace model of CSP will suffice. Here, a process P is identified with sets of traces τP. Such a set represents the possible sequences of events that P may be prepared to execute.

Processes can be combined in various ways but we will only need alphabetised parallel composition, denoted by $P||_A Q$, in which the two processes P and Q interact by synchronising over events in the set A. In the special case where $\tau P = \tau Q = A$, the traces of $P||_A Q$ are given by $\tau P \cap \tau Q$, i.e. the intersection of the two trace sets. Intuitively, an event from A can only occur if both processes simultaneously agree to execute it. The model of interaction is thus a synchronised handshake. Asynchronous interactions can be modelled using suitable buffering processes.

We can prefix a process term with an event: $a \rightarrow P$. This process will first perform the event a and then behave as the process P. The environmental choice between P and Q, $P[]Q$, represents a process that may behave as P or as Q, the choice being made by the environment. We will also use the channel notation: events may take the form $c.a$ where c denotes a channel and a an event (value) on that channel drawn from the channel type. We can have compound channels with multiple fields, e.g. $c.a.b$.

We need the notation for hiding events: $P \backslash A$ denotes the process P with all events from the set A hidden from the environment. We also need the process $STOP_A$ which, when composed in parallel with any process, will prevent the occurrence of any event from the set A.

The final concept we require is that of (trace) refinement. P is a trace refinement of Q, denoted by:

$$Q \sqsubseteq_T P \; when \; \tau P \subset \tau Q$$

Thus, any behaviour that P can exhibit must also be possible for Q.

Safety properties can be captured using trace refinement. Let Q encode the specification (τQ denotes acceptable behaviours). If P trace refines Q it is an acceptable implementation of Q, i.e., it will not perform any unacceptable behaviours. However, because the model ignores non-determinism, trace refinement will not provide any guarantees about liveness of the implementation. More elaborate models are available to deal with non-determinism, dead-lock, live-lock, etc. but these are not required for the current presentation.

5.2 Tools

CSP has tool support in the form of the FDR model-checker. This checks whether or not one CSP specification is a refinement of another. One can thus

produce a high level CSP specification of the requirements and a lower level design and check the latter against the former. This can be continued recursively towards an actual implementation. If the check fails, the tool returns a (minimal) counter-example which typically helps identify the cause of the violation. In some cases a counter-example will show up some problem with the specification, a perfectly acceptable behaviour that has been ruled out. This renders model-checking tools like FDR highly effective at debugging designs [6]. The model-checker FDR and the animator tool *Probe* are available from Formal Systems Europe (www.formal.demon.co.uk/). The animator is useful to help explore and validate policies formulated in CSP.

6. Clinical Trails Case Study

We describe the security model for a clinical-trials demo, derived from a case study used in the HARP project, firstly informally and then formally using CSP. Once we have set up a formal model of the system along with statements of the security requirements the model can be verified against the requirements using the model-checking tool FDR.

The clinical trials example illustrates many of the issues and many of the requirements typical of security policies. On the other hand it is sufficiently compact to remain understandable and tractable. To streamline the presentation we have simplified the original model.

The trial is distributed across a number of hospitals. Each hospital has a set of clinicians attached to it that are recognised users of the system. They have unique identities, public/secret key pairs and all the usual paraphernalia to interact securely with the system. For simplicity we assume that clinicians come in a single flavour, i.e., any clinician, subject to certain constraints, can adopt any of the roles described below.

The core of the trial is the database of patient records. This could be thought of as being distributed across the participating hospitals but for the level of abstraction of this presentation this will not be addressed. The database is accessed by clinicians via secure sessions. At the start of a session, a clinician must authenticate herself to the system, establish a secure channel and select from one of two roles: Record Creator (RC) or Record Validator (RV).

In the RC role, a clinician can request the creation of a new record for a patient along with appropriate data. When a new record is successfully created, a version marked *fresh* is added to the database and a request for the record to be validated is broadcast. A clinician in the RV role can respond to such a request, as long as he isn't the clinician who originally created the record (separation of duty). In the RV role a clinician is supposed to check the data for errors, plausibility etc. He may:

1 Accept: add the record to the data unchanged via the *store* action.

2 Reject: return the record with annotations suggesting changes.

We do not concern ourselves as to how these choices are made. In the CSP model it will be modelled as a non-deterministic choice. If the record is accepted it is marked as *valid* in the database and no further changes to that record are allowed. If it is rejected it is marked *pending* and returned to the originating clinician who can view the suggested changes in RC role. If she now accepts the suggested changes the record is altered accordingly and the database is updated with a version marked *valid.* All records marked *valid* are thenceforth immutable. She may reject the suggested changes in which case the record is returned to the RV who checked it. This process continues until both parties are happy and a valid version added to the database.

The system should satisfy the following properties, informally stated for the moment:

- Integrity: once a record has been marked *valid* and added to the database it is never deleted or altered. (More sophisticated policies might allow subsequent appends.)
- All records should be signed by an appropriate clinician. (Where "appropriate" will be precisely specified by the policy).
- Any addition to the database must have been preceded by a corresponding request from an authorised clinician acting in the appropriate role. Any correction to a record must be signed by an authorised RC and must have been preceded by a RV raising a challenge to that record.
- For any given record identifier (patient Id) there can be at most one record marked *valid* in the database.
- For a given record ID if there is a record marked *valid* this must be the most recent record with this ID in the database.
- A clinician should only be able to view a record whilst in the RC role and furthermore she should only be able to access records marked *valid.*

7. Abstract Model

Due to space constraints we will not give the details of the system model here, they can be found in the full version [2], where the full CSP descriptions can be found. The system model is not terribly informative. What is more interest is the encoding of the security requirements that we come to shortly.

The model described at [2], includes correctly behaving clinicians as well as rogue clinicians. Rogue clinicians attempt to perform actions forbidden by the

policy. The Monitor, if correctly modelled, will enforce the policy. Errors in the Monitor model will allow the rogue clinician to violate the policy. This will be detected when the model is run in FDR against the security requirements as failures of the refinement checks.

8. Formalising Security Properties

Here we give CSP formulations of many of the security properties described earlier.

8.1 Confidentiality

Clinicians should only be able to view records for which they are responsible and whilst in the RC role. Furthermore they should only be able to view records marked *valid*. This can be encoded as follows: first we introduce a set of *breach* events whose occurrence would represent a violation of this requirement:

$$breach = \{view.d.RC.p \mid d \neq \pi_{RC}(Record(p))\} \cup \{view.d.RV.p\}$$

The view channel is a compound one with the following fields: requesting doctor id, role, patient id of the requested record. Doctors use this channel to request sight of records. π_{RC} is a projection operator that returns the identity of the doctor who created the record. This set therefore corresponds to events in which, in the RC role, a doctor gets to view a record for which his ID does not match the doctor id of the record or he gets to view anything in the RV role. The notation has been slightly simplified from the original HARP report for ease of presentation. Hospital identities have been elided for example.

The property is now easily encoded by requiring that the system should never allow the occurrence of an event from *breach*. We can capture this as a refinement assertion:

$$STOP \sqsubseteq_T \ (SYS) \backslash (\Sigma - breach)$$

The Left Hand Side of the refinement is the System (SYS) with all events (Σ) hidden except the breach events. If the implementation ensures that no breach events can occur, this process will refine $STOP$, the process that does nothing. Conversely, should the system model ever allow a breach event the refinement will be violated and a counter-example returned by the tool. The counter-example will detail the sequence of events leading to the breach.

8.2 Separation of Duty

A clinician should never act as the RV for his own record or be able to accept or reject a record whilst in the RC role.

Let *cheat* represent the set of events in which a clinician violates the policy:

$$\begin{aligned} cheat = \quad & \{accept.d.RC.p\} \cup \{reject.d.RC.p\} \cup \\ & \{accept.d.RV.p \mid d = \pi_{RC}(Record(p))\} \cup \\ & \{reject.d.RV.p \mid d = \pi_{RC}(Record(p))\} \end{aligned}$$

The *accept* and *reject* channels are used to accept of reject records respectively. Thus, the set *cheat* represents any accept or reject actions performed in the RC role or any action performed in the RV role for which the clinician's identity matches the doctor id of the record. The separation of duty property can now be encoded as:

$$STOP \sqsubseteq_T (SYS) \setminus (\Sigma - cheat)$$

8.3 Integrity

A record should only be added to the data-base if preceded by the appropriate action by a clinician. More precisely: adding a record marked *fresh* should be preceded by the appropriate open action. We encode these as Schneider style authentication properties [9]:

$$SYS ||_{\{open.p, store.p.fresh\}} STOP \sqsubseteq_T SYS ||_{open.p} STOP$$

On the RHS of the refinement we have the system with both the *open.p* and *store.p.fresh* events blocked. On the LHS only the *open.p* events are blocked. Thus, if the system is able to perform a *store.p.fresh* event without an *open.p* event it would violate this check and violate the policy.

Similarly, storing a record p with its validity flag set to "pending" should be preceded by a validate action:

$$SYS ||_{\{validate.p, store.p.pending\}} STOP \sqsubseteq_T SYS ||_{validate.p} STOP$$

For a record to be stored marked *valid* it should be preceded by an appropriate check action:

$$SYS ||_{\{check.p, store.p.valid\}} STOP \sqsubseteq_T SYS ||_{check.p} STOP$$

8.4 Anonymity

Anonymity can be elegantly stated in CSP, see for example reference [6]. The essence of the idea is to stipulate that an appropriate view of the system is left unchanged by arbitrary shuffling of the patient identifiers. The shuffling of identifiers is modelled in CSP by use of the renaming operator. The appropriate view here would be that obtained from abstracting all channels except those available for public access.

9. Future Directions

We plan to go on to examine some more elaborate case studies that take account of the requirements of other stakeholders such as clinical researchers, administrators, insurance firms etc. Future work will seek to include exceptions and delegation in the CSP framework.

We will also investigate the use of generalised non-interference to capture richer classes of information flow, in particular the idea of representing confidentiality, integrity and anonymity requirements as system symmetries. We also intend to investigate the theoretical and practical limits of what aspects of security policy may be enforced by purely technical means and clarify the relationship between the information flow requirements and access control mechanisms.

10. Conclusions

We have argued that the security requirements of concern in the context of healthcare can be formulated in a mainstream formal method and do not require bespoke frameworks. We have presented formulations of many of the typical policy requirements in the process algebra CSP. An example policy based on a case study that arose in the Framework 5 Project HARP has been presented and we have discussed how various tools can be used to assist in the verification and validation of such policies.

Acknowledgments

This work was initially funded under the Framework 5 Project HARP whilst the first author was visiting NR Oslo and continued during a period as a visiting scientist at the SEI, Pittsburgh. It has benefited from discussions with Latanya Sweeney and John Dobson.

References

[1] M.J. Butler, csp2B: A practical approach to combining CSP and B (www.ecs.soton. ac.uk/ mjb/csp2B/).

[2] HARP Project (http://telecom.ntua.gr/ HARP/HARP/HARP.htm).

[3] C.A.R. Hoare, *Communicating Sequential Processes*, Prentice Hall, 1985.

[4] A.W. Roscoe, *The Theory and Practice of Concurrency*, Prentice-Hall, 1997.

[5] P.Y.A. Ryan, Mathematical models of computer security, *Proceedings of the FOSAD Summer School, LNCS 2171*, R. Gorrieri (ed.), Springer, 2000.

[6] P.Y.A. Ryan et al., *Modelling and Analysis of Security Protocols*, Pearson Scientific, 2001.

[7] P.Y.A. Ryan and S.A. Schneider, Process algebra and non-interference, *Proceedings of the Computer Security Foundations Workshop*, extended version in *Journal of Computer Security*, vol. 9(1/2), 2001.

[8] F. Schneider, Enforceable policies, *ACM Transactions on Information and System Security* vol. 3(1), pp. 30-50.

[9] S.A. Schneider, Security properties and CSP, *Proceedings of the IEEE Symposium on Security and Privacy*, 1996.

[10] S.A. Schneider, *Concurrent and Real-Time Systems: The CSP Approach*, Wiley, 2000.