Security Requirements for CSCW Systems

Simon N. Foley
Computer Science Department
University College,
Cork, Ireland

Jeremy Jacob
Computer Science Department
University of York,
York, United Kingdom

Abstract

CSCW systems provide computer support to facilitate cooperation between users. This project is concerned with developing an approach for the formal specification of functionality and security requirements of a CSCW application. These requirements give rise to various safety and confidentiality properties that a CSCW system, supporting the application, should uphold. A case study has been developed using this specification technique.

1 Introduction

The purpose of Computer Supported Co-operative Working(CSCW) is to provide computer support that facilitates cooperation between users. This support can be as simple as the provision of an Email application, to a sophisticated concurrent engineering application mediating and resolving conflicts between participants. A great deal of work has been done on the technological aspects of CSCW: the problems of how one actually provides computer and networking support for cooperation. The reader is referred to [3] for an introduction.

We are concerned with what is meant by *confidentiality* in CSCW applications. To do this, we developed a formal definition of what is meant by a CSCW application and the support that we would expect the CSCW system to provide. We are not concerned with, at least at this stage, *how* security should be enforced. But we are concerned with how one might specify and reason about the security requirements for a CSCW application and the resulting properties that a CSCW system, providing support for the application, should uphold.

It is not possible in the limited space available for this report to provide the reader with the technical details of our research. Instead, we outline our approach by using a very simple example. This example is a simplification of a case study that we developed during our investigations. The reader is directed to [1] for further details.

2 Secure Electronic Examinations

A tiny university has one chair, one lecturer and one student. The chair appoints setters of an examination paper, which the student takes. There are two CSCW activities: setting and taking the examination paper.

We have developed a cliché, or idiom, for the formal specification of such CSCW activities: it gives us a guideline on the way an activity can be organized for specification. Our cliché, while abstract, does appear to conform to the criteria that is used in [4] to identify CSCW applications. The safety and security properties that a CSCW system, supporting such activities, is expected to uphold are expressed in the semantic space associated with Hoare's Communicating Sequential Processes and are considered elsewhere[1].

The activity of setting the examination paper is specified as:

Protected Activity `SetPaper` **is**

  start $:= \{announce\}$

  finish $:= \{release\}$

  join $:= \{appoint\text{-}setter\}$

  leave $:= \{resign\text{-}setter\}$

  do $:= \{\ c : TEXT \bullet write.c\ \}$

  conclude$:= \{\ c : TEXT \bullet \langle write.c, release.c \rangle\ \}$

end `SetPaper`

The variable binding **start** $:= \{announce\}$ in specification `SetPaper` specifies that the activity is started or activated once an *announce* command is engaged (by the professor). The activity is finished, and no further participation is possible, when the paper is released to students.

Only members may participate in an activity. A lecturer becomes a member of activity `SetPaper` when the *appoint-setter* command is engaged (by the professor). Once a member, the lecturer participates in the activity by engaging events of the form *write.c*, where c represents the text that makes up an examination paper. Revisions may be made by writing further versions of the paper.

Note that if the paper has not yet been written then it is inappropriate to release the paper, and the activity should not be considered finished. This constraint is specified by the binding for **conclude**, whereby an examination paper may not be released if it has not yet been written.

Activity `SetPaper` is specified as a *protected* activity. This means that a non-participant (student/examinee) in the activity may not learn anything about the commands that are engaged by the participants (lecturer/setter) during the lifetime of the activity. Thus, the examinee may not learn any details of the contents of the examination paper before the start of the examination. The confidentiality property we developed for the the protected activity cliché is based on a version of non-interference. Non-interference and its variants have been well studied since its introduction by Goguen and Meseguer[2].

The examination is started by a *release* event and terminates when the examination is closed with an event of the form *close.a*, where a gives the last version of the examination script written by the student. The activity is specified by

Activity `TakePaper` **is**

$$
\begin{array}{ll}
\textbf{start} & := \texttt{SetPaper}.\textbf{finish} \\
\textbf{finish} & := \{\ a : TEXT \bullet close.a\ \} \\
\textbf{join} & := \{register\} \\
\textbf{leave} & := \{deregister\} \\
\textbf{do} & := \{\ a : TEXT \bullet answer.a\ \} \\
\textbf{conclude} & := \left(\begin{array}{l} \text{if } \texttt{Member}(student) \\ \text{then } \{a : TEXT \bullet \langle answer.a, close.a\rangle\} \\ \text{else } \ \{\langle close.cancel\rangle\} \end{array}\right)
\end{array}
$$

end `TakePaper`

The student must *register* to answer the paper, and may do so before or even during the examination. If the student deregisters before or during the examination, the examination is closed with a cancelled script.

Activity `TakePaper` is not a protected activity: it is reasonable that the lecturer and professor may learn about the progress of the student during the examination.

3 Conclusions

We have developed a notation for describing and specifying CSCW systems where confidentiality is an issue. We believe that this notation is suitable for use by people who are not trained in formal methods. We believe this for the following reason: *The notion of confidentiality used is* very *simple.*

A formal treatment of our approach can be found in [1]. It defines the safety and confidentiality properties that a CSCW system, supporting the application, should uphold. It also provides a detailed case study on the specification of the Secure Electronic Examinations system. This case study defines many activities and their associated functionality and confidentiality requirements. These include multiple examination papers and students (collusion prohibited during examinations), and the reviewing of examination papers (a paper may not be released unless signed off by all reviewers).

Despite its simplicity our approach works well for the SEE case study[1]. A danger is that in other applications than SEE it is too simple. Clearly we must try further case studies before definitely concluding that we have a sophisticated enough notion.

We are reasonably happy with the "outer" form of the language. The concepts to which one must give values are (for the most part) easy to comprehend. But the notation in which the values are described is still "over-mathematical" for the audience we have in mind. This must be studied further before we can claim to have been completely successful.

References

[1] S.N. Foley and J Jacob. Specifying security in CSCW systems. *submitted for publication*, January 1995.

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

[3] J.D. Palmer and N.A. Fields. Special issue on computer supported cooperative work. *IEEE Computer*, 27(5), May 1994.

[4] Reinhard W. et al. CSCW tools: Concepts and architectures. *IEEE Computer*, 27(5):28–36, May 1994.