Believing the Integrity of a System *(Invited Talk)*

Simon N. Foley [1,2]

Department of Computer Science
University College Cork
Ireland

Abstract

An integrity policy defines the situations when modification of information is authorised and is enforced by the protection mechanisms of a system. Traditional models of protection tend to define integrity in terms of ad-hoc authorisation techniques whose effectiveness are justified more on the basis of experience and "best practice" rather than on any theoretical foundation. In a complex application system it is possible that an integrity policy may have been incorrectly configured, or that the protection mechanisms are inadequate, resulting in an unexpected system compromise. This paper examines the meaning of integrity and and describes a simple belief logic approach for analysing the integrity of a system configuration.

Key words: Security, Integrity, Security Protocols, Belief logics, System Configuration.

1 Introduction

The 2001 Computer Crime and Security Survey [9] reported that 196 of the respondents to the survey could quantify their losses due to unauthorised use of computer systems at a total of US$378 million in the previous year. While access-control mechanisms, firewalls and so forth may help counter such losses, we can never be confident about security unless we are provided with some assurance of their effectiveness. Such assurance may be achieved, in part, by analysing whether a formal description of the system upholds certain security properties. These properties include confidentiality (no unauthorised release of information) and integrity (no unauthorised modification of information). The study of integrity as a formal security property has received little attention

[1] Supported by Enterprise Ireland Basic Research Grant Scheme SC/2003/7/ *FITNYSS Foundations for Integrity and Analysis.*

[2] Email: s.foley@cs.ucc.ie

This is a preliminary version. The final version will be published in Electronic Notes in Theoretical Computer Science
URL: www.elsevier.nl/locate/entcs

within the research community; confidentiality has been extensively studied and is the better understood of the two properties.

Early security research [3] characterised integrity in terms of read-write access controls between subjects and objects. This provides for a very coarse interpretation of integrity [30]; for example, once granted access to an account database, a bank clerk can make any change to the customer's account details. Access triples, well-formed transactions, and the principles of encapsulation [8,25], provide finer grained control by constraining the operations that a subject may carry out on an object: the bank clerk may execute only deposit or withdraw operations to access an account database.

Many integrity compromises are a result of 'insiders' executing fraudulent but authorised operations [9]. For example, the bank clerk executes an account deposit without lodging actual funds. Separation of duty [8,13,37,35] controls decrease the potential for fraud by involving at least two individuals at different points in a transaction: for example, by reconciling bank accounts and funds received each day, a supervisor detects and corrects the fraudulent deposit by the clerk. Role Based Access Control models [31,32] and authorisation models [2,21] provide integrity controls based on structures that organise related operations into roles and constrain the way that roles may be assigned and/or inherited by users; separation of duty is expressed within these models using role constraints.

These conventional security models describe *controls* for achieving integrity; they take an operational and/or implementation oriented approach by defining *how* to achieve integrity. No attempt is made to formalise a *property* that defines *what* is meant by integrity. For example, [8] recommends a combination of separation of duties, access-triples and auditing as a strategy for achieving integrity: it does it not attempt to address what is meant by integrity. Confidence is achieved to the extent that good design principles have been applied; there is no assurance that an integrity property is upheld. Thus, when we define a complex separation of duty policy we do not know, for certain, whether a dishonest user can bypass the intent of the separation via some unexpected circuitous, but authorised, route.

Jacob [20] formalises integrity as a functional property. This interpretation of integrity means that an integrity mechanism determines whether the current request for an operation is authorised based on the history of past authorisation requests that led to the current state.

In our research [14,15], we argue that integrity should be regarded as a non-functional property. Non-functionality means that in order to determine authorisation, it is necessary to examine *every possible* history of requests that could have reached the current state. Non-interference is an example of a non-functional property, and while non-interference and its derivatives have been extensively studied [12,11,17,28,29], designing and verifying security mechanisms that uphold non-functional properties is known to be difficult [24,34]. Thus, we argue that building mechanisms that uphold integrity (a

non-functional property) can be expected to be as difficult as building mechanisms that uphold non-interference.

This difficulty in properly analysing the integrity of a system is illustrated in [14,15], whereby, to provide integrity guarantees it is necessary to model the behaviour of both the system (with its protection mechanisms) and the *infrastructure* in which the system operates. Infrastructure is everything that serves the system requirements: software, hardware, users, and so forth. Even if a system is functionally correct, the infrastructure is likely to fail: software fails, users are dishonest, do not follow procedures, and so forth. The system and its security mechanisms must be designed to be resilient to these infrastructure failures. Only when a system is characterised in this way can it become possible to completely analyse whether a particular system configuration (including security policy) ensures integrity.

The non-functional approach to integrity proposed in [14,15] provides a formal algebraic semantics that requires detailed formal specifications to be provided for the system and its infrastructure. This requires considerable specification effort. The cost of such in-depth specification and subsequent analysis may be justified for small critical security mechanisms. However, we argue that such integrity analysis would not scale well to the configuration of a large and/or complex application system because it would be necessary to formally specify and reason about the potential behaviour of *every* infrastructure component, user and so forth.

2 Analysis of Security Configurations

We are interested in developing shallow and pragmatic security analysis methods for systems. This is achieved through the analysis of how a system is *configured*, rather than an analysis of its underlying mechanisms and protocols. Instead of concentrating on detailed semantics and complete formal verification of components, we are concerned more with with the ability to trace, at a practical level of abstraction, how component security requirements relate to each other and any overall security requirements. We believe that a complete security verification of a system is not achievable in practice; we seek some degree of useful feedback from an analysis that a particular system configuration is reasonable.

In [4] we describe a modelling approach that requires less semantic detail about the operation of the system and its infrastructure. Rather than attempting to model the complete behaviour of the system and infrastructure (as in [15]), only those components that are considered relevant to the security policy and configuration are modelled. This is done by modelling the system and infrastructure in terms of the *constraints* that they impose over security relevant components of the system. This results in a definition of integrity consistency that can be solved as a constraint satisfaction problem [23]. An advantage to expressing integrity analysis as a constraint satisfaction problem

is that there exists a wide body of existing research results on solving this problem for large systems of constraints in a fully mechanised manner. Constraints have been used in many practical analysis tools, such as Concurrent Engineering and Computer Aided Verification.

Another example of a 'shallow' configuration analysis is given in [33]. This approach searches for possible conflicts between separation of duty, user role assignment and role inheritance rules in RBAC models. [4] provides an (albeit abstract) semantics for integrity that may be used to determine whether the separation of duty controls actually achieve external consistency and whether the application uses integrity mechanisms correctly.

3 Towards a Logic of Integrity

We argue that the integrity-algebraic approach [14,15] does not scale well to large complex systems. In this paper we propose a logic-based approach: such an approach facilitates a high-level analysis of a system that is too complex to be amenable to algebraic analysis. The basis for this approach rests on our conjecture that belief logics [19] can be used to provide a complementary approach to integrity/external consistency analysis.

Studying the effects of normal versus abnormal infrastructure behaviour of systems has been successfully applied to cryptographic security protocols, for example [10,26,27]. Analysis is based on a generic behaviour (called "spy" [27] or "adversary" [10]) that characterises the untrusted network (abnormal infrastructure) over which a security protocol operates. Algebraic integrity analysis [15] generalises this by considering a complex adversary that characterises the combined threats from the infrastructure that a protection mechanism must withstand. This viewpoint—that many different adversaries with differing behaviours should be modelled in the environment—is also adopted in [1] in the analysis of security protocols.

Belief logics such as [6,18,36] have also been successfully used to analyse beliefs held between participants in cryptographic security protocols. In these logics the behaviour of the adversary is implicit in the deduction rules and in the stated beliefs of the principals. Therefore, integrity analysis based on belief-logic is possible if one reflects the 'infrastructure adversary' in terms of suitable logic deduction rules and in terms of the beliefs held by principals. We are developing a belief logic that can be used to analyse integrity. The following example sketches how it might be used.

Let P, A, B, C represent principals, ϕ represent a formulae of the logic and X represents a message (which can also be a formulae). The logic has the following formulae constructions.

- $P \models\!\!\equiv X$: P *believes*, or would be entitled to believe X.
- $P \mid\!\sim X$: P at some time *said/sent* a message that indicated X.
- ΥX: X is *consistent* with some real world value. We use this to characterise

integrity in terms of *external consistency*, which is informally described in [8] as *"the correct correspondence between data objects and the real world"*, for example, a bank account balance corresponds to the customer's own belief about the account based on withdrawals and deposits.

We assume that formulae can be derived using the usual propositional logic operators $\vee$, $\wedge$ and $\rightarrow$, and the usual K-Axiom of modal logics:

$$\frac{P \models (\phi_1 \rightarrow \phi_2), P \models \phi_1}{P \models \phi_2}$$

Consider a customer C making a deposit (via an envelope) to an ATM machine A. The relationship between these principals are described in terms of beliefs with respect to the belief modality of the customer C.

The customer believes that the ATM A registers her deposit correctly:

$$C \models A \mathrel{|\!\sim} C \mathrel{|\!\sim} \Upsilon \mathsf{dep} \tag{1}$$

This reflects the belief of the customer that when the transaction is complete the ATM has said/registered that the customer made (said) a consistent deposit.

The customer believes that once the bank (for simplicity, A) is satisfied with (believes) the consistency of the deposit then it updates the customer account acct.

$$C \models ((A \models \Upsilon \mathsf{dep}) \rightarrow \Upsilon \mathsf{acct}) \tag{2}$$

These two logical formulae represent the operational beliefs held by C concerning a deposit transaction.

The customer believes that the ATM is honest: it says only what it believes [3]:

$$C \models ((A \mathrel{|\!\sim} \phi) \rightarrow (A \models \phi)) \tag{3}$$

Thus, formula 1 reduces to

$$C \models A \models C \mathrel{|\!\sim} \Upsilon \mathsf{dep} \tag{4}$$

The customer also believes that the bank believes she honest, that is, she will only say things that can be believed.

$$C \models A \models ((C \mathrel{|\!\sim} \phi) \rightarrow \phi)$$

If the customer believes that her deposit is correct ($C \models \Upsilon \mathsf{dep}$), then given the nature of the system and infrastructure as described above, the following can be deduced within the logic.

$$C \models \Upsilon \mathsf{acct}$$

[3] Note that, for the sake of simplicity, we do not consider how *recently* A said a message, or whether it is a replay of some earlier message.

That is, the customer believes that her account has integrity as a result of the transaction.

This deduction is possible because the ATM and customer are individually honest and competent. If the customer does not believe that the bank will honestly reconcile the amount in the envelope then it is not possible to deduce this account consistency. For example if the customer did not believe that A is honest (deletion of formula (3)) then the goal cannot be deduced. This result is as expected and it indicates that further operational checks and balances may be needed in the system, if account consistency is to be ensured in the presence of such infrastructure dishonesty.

For example, suppose that a bank clerk B validates the deposit in the ATM. After a transaction, the customer believes the clerk confirms the amount in the envelope

$$C \models B \mid\sim C \mid\sim \Upsilon \mathsf{dep} \tag{5}$$

However, suppose that only one of A and B can be considered honest. Formula (3) is replaced by

$$\begin{aligned} C \models (((A \mid\sim \phi) \rightarrow (A \models \phi)) \\ \vee ((B \mid\sim \phi) \rightarrow (B \models \phi))) \end{aligned} \tag{6}$$

Furthermore, suppose that the software that reconciles what A and B say about a deposit has a way to detect which is honest [4]. Thus, formula (2) is replaced by

$$C \models (((A \models \Upsilon \mathsf{dep}) \vee (B \models \Upsilon \mathsf{dep})) \rightarrow \Upsilon \mathsf{acct}) \tag{7}$$

These last two formulae correspond to a separation of duty-like rule on the transaction and it becomes possible to derive the original goal $C \models \Upsilon \mathsf{acct}$.

Note that a complete analysis should also consider the belief modality with respect to bank: can it be defrauded by dishonest customers?

Our example is grossly simplified. We are developing an expressive integrity logic, making it possible to properly express beliefs about operations and/or techniques that are used to guarantee integrity, such as cryptography, separation of duty, well-formed transactions, auditing and so forth.

4 Discussion and Conclusion

An abstract logic based approach facilitates high-level analysis of a system that is too complex to be amenable to algebraic analysis. The logic approach follows the strategy to make only the needed distinctions and no more and simplifies the specification and analysis of integrity.

A potential weakness of taking the logic approach is that, like existing authentication logics, it is possible that an integrity logic will miss classes

[4] For simplicity, we have abstracted away the other bank principals that determine this when they resolve conflicts that arise between the Clerk and ATM.

of attacks that could be identified by the algebraic approach in [15]. In the integrity logic we can model the infrastructure in terms of beliefs about the honesty and competency of the components. While limited in expression, this results in specifications that are relatively easy to develop and analyse. This is contrasted with the algebraic approach [15] which requires the behaviour of each principal/infrastructure to be fully specified, leading to unwieldy specifications that are difficult to develop and require a high degree of expertise to verify. The logic approach trades off completeness of analysis against ease of use and the ability to conduct fully automatic analysis of complex systems [5,22].

The logic of integrity outlined in this paper is an adaptation of the Simple Logic [7] which uses an expressive but very simple logical system. We are currently exploring automated analysis techniques; we have implemented an automatic verification tool for the Simple Logic using Theory Generation [22]. In [38] an automatic protocol generator is described that uses synthesis rules to compute protocols within the Simple Logic. Future research will explore how such techniques can be used to automatically synthesis acceptable security (integrity) policy configurations that uphold external consistency.

It is evident from the example above that if a configuration policy (of the system/infrastructure) is to be specified in terms of low-level beliefs then the exercise has the potential to be tedious and error prone (although, we believe less problematic than the algebraic approach). A topic of future research is to investigate the potential of using requirements elicitation techniques in the specification of security policy configurations. In particular, [16] takes a Safety Engineering approach in which Hazard Analysis is used for the elicitation of security protocol requirements.

Acknowledgements

Thanks to Stefano Bistarelli for many useful discussions on the nature of Integrity. Development and implementation of the Theory Generation encoding of the Simple Logic verifier was by Daithi O'Crualaoich and its adaptation to an automatic protocol generator by Hongbin Zhou.

References

[1] G. Bella and S. Bistarelli. Soft Constraints for Security Protocol Analysis: Confidentiality. In *Proc. of the 3rd International Symposium on Practical Aspects of Declarative Languages (PADL'01)*, LNCS 1990, pages 108–122. Springer-Verlag, 2001.

[2] E. Bertino et al. An authorization model and its formal semantics. In *Proceedings of the European Symposium on Research in Computer Security*, pages 127–142. Springer LNCS 1485, 1998.

[3] K.J. Biba. Integrity considerations for secure computer systems. Technical Report MTR-3153 Rev 1 (ESD-TR-76-372), MITRE Corp Bedford MA, 1976.

[4] S. Bistarelli and S.N. Foley. Analysis of integrity policies using soft constraints. In *Proceedings of IEEE Workshop Policies for Distributed Systems and Networks*, pages 77–80, June 2003.

[5] Stephen H. Brackin. A HOL extension of GNY for automatically analyzing cryptographic protocols. In *PCSFW: Proceedings of The 9th Computer Security Foundations Workshop*. IEEE Computer Society Press, 1996.

[6] M. Burrows, M. Abadi, and R. M. Needham. A logic of authentication. Technical Report Report number 39, Digital Systems Research Center, February 1989.

[7] L. Buttyán, S. Staamann, and U. Wilhelm. A simple logic for authentication protocol design. In *Proceedings of the 11th IEEE Computer Security Foundations Workshop (CSFW '98)*, pages 153–163, Washington - Brussels - Tokyo, June 1998. IEEE.

[8] D. D. Clark and D. R. Wilson. A comparison of commercial and military computer security models. In *Proceedings Symposium on Security and Privacy*, pages 184–194. IEEE Computer Society Press, April 1987.

[9] Computer Security Institute/US Federal Bureau of Investigation. *Computer Crime and Security Survey*, 2001.

[10] D. Dolev and A.C. Yao. On the security of public key protocols. *IEEE Transactions on Information Theory*, 29(2):198–208, 1983.

[11] R. Focardi and R. Gorrieri. A classification of security properties for process algebras. *Journal of Computer Security*, 3(1):5–33, 1995.

[12] S.N. Foley. A universal theory of information flow. In *Proceedings 1987 IEEE Symposium on Security and Privacy*, pages 116–121. IEEE Computer Society Press, 1987.

[13] S.N. Foley. The specification and implementation of commercial security requirements including dynamic segregation of duties. In *ACM Conference on Computer and Communications Security*, pages 125–134, 1997.

[14] S.N. Foley. Evaluating system integrity. In *Proceedings of the ACM New Security Paradigms Workshop*, 1998.

[15] S.N. Foley. A non-functional approach to system integrity. *Journal on Selected Areas in Communications*, 21(1), Jan 2003.

[16] Nathalie Foster and Jeremy Jacob. Hazard analysis for security protocol requirements. In *Advances in Network and Distributed Systems Security, IFIP TC11 WG11.4 First Annual Working Conference on Network Security*, IFIP Conference Proceedings, 2001.

[17] J. A. Goguen and J. Meseguer. Unwinding and inference control. In *Proceedings 1984 IEEE Symposium on Security and Privacy*, pages 75–86. IEEE Computer Society, 1984.

[18] Li Gong, R. M. Needham, and R. Yahalom. Reasoning about belief in cryptographic protocols. In D. Cooper and T. Lunt, editors, *Proceedings 1990 IEEE Symposium on Security and Privacy*, pages 234–248, May 1990.

[19] G.E. Hughes and Cresswell M.J. *An Introduction to Modal Logic*. University Paperbacks, Methuen Press, 1968.

[20] J.L. Jacob. The basic integrity theorem. In *Proceedings of the Computer Security Foundations Workshop IV*, pages 89–97. IEEE Computer Society Press, June 1991.

[21] S. Jajodia et al. A logical language for expressing authorizations. In *Symposium on Security and Privacy*, pages 31–42. IEEE Press, 1997.

[22] D. Kindred and J.M. Wing. Theory generation for security protocols. *ACM TOPLAS*, July 1999.

[23] A.K. Mackworth. Constraint satisfaction. In S.C. Shapiro, editor, *Encyclopedia of AI (second edition)*, pages 285–293. John Wiley & Sons, 1992.

[24] J. McLean. A general theory of composition for trace sets closed under selective interleaving functions. In *Proc. IEEE Symposium on Research in Security and Privacy*, pages 79–93, 1994.

[25] R. Needham. Later developments at cambridge: Titan, cap and the cambridge ring. *Annals of the History of Computing*, 14(4):57, 1992.

[26] L.C. Paulson. The inductive approach to verifying cryptographic protocols. *Journal of Computer Security*, 6:85–128, 1998.

[27] A.W. Roscoe. Using intensional specifications of security protocols. In *Proceedings of the Computer Security Foundations Workshop*, pages 28–38. IEEE Press, 1996.

[28] A.W. Roscoe, J.C.P. Woodcock, and L. Wulf. Non-interference through determinism. *Journal of Computer Security*, 4(1), 1995.

[29] P.Y.A. Ryan and S.A. Schneider. Process algebra and non-interference. In *IEEE Computer Security Foundations Workshop*, pages 214–227, 1999.

[30] R. Sandhu. A perspective on integrity mechanisms. In *Proceedings of the Computer Security Applications Conference*, page 279, 1989. Panel Discussion paper.

[31] R Sandhu et al. Role based access control models. *IEEE Computer*, 29(2):38–47, 1996.

[32] R. Sandhu, D. Ferraiolo, and R. Kuhn. The nist model for fole-based access-control. In *Proceedings of the ACM Workshop on Role-Based Access Control*, 2000.

[33] A. Schaad and D. Moffett. The incorportation of control principles into access control policies. In *Workshop on Policies for Distributed Systems and Networks*, Bristol, UK, 2001.

[34] F.B. Schneider. Enforcable security policies. *ACM Transactions on Information and Systems Security*, 3(1):30–50, February 2000.

[35] R. Simon and M. Zurko. Separation of duty in role based environments. In *Proceedings of the Computer Security Foundations Workshop*, pages 183–194. IEEE Press, 1997.

[36] P. Syverson and P.C. van Oorschot. On unifying some cryptographic protocol logics. In *Proceedings of IEEE Computer Security Foudations Workshop*, pages 14–28, 1994.

[37] U. S. Department of Defense. Integrity-oriented control objectives: Proposed revisions to the trusted computer system evaluation criteria (TCSEC). Technical Report DOD 5200.28-STD, 1991.

[38] H. Zhou and S.N. Foley. Fast automatic synthesis of security protocols using backward search. In *ACM Workshop on Formal Methods for Security Engineering*, Washington, DC, USA, 2003.