Requirement Engineering meets Security: A Case Study on Modelling Secure Electronic Transactions by VISA and Mastercard*

Paolo Giorgini[1] and Fabio Massacci[1] John Mylopoulos[1,2]

[1] Department of Information and Communication Technology
University of Trento - Italy
{massacci,giorgini}@dit.unitn.it
[2] Deptartment of Computer Science
University of Toronto - Canada
jm@cs.toronto.edu

Abstract. Computer Security is one of today's hot topic and the need for conceptual models of security features have brought up a number of proposals ranging from UML extensions to novel conceptual models. What is still missing, however, are models that focus on high-level security requirements, without forcing the modeler to immediately get down to security mechanisms. The modeling process itself should make it clear why encryption, authentication or access control are necessary, and what are the tradeoffs, if they are selected. In this paper we show that the *i**/Tropos framework lacks the ability to capture these essential features and needs to be augmented. To motivate our proposal, we build upon a substantial case study – the modeling of the Secure Electronic Transactions e-commerce suites by VISA and MasterCard – to identify missing modeling features. In a nutshell, the key missing concept is the separation of the notion of offering a service (of a handling data, performing a task or fulfilling a goal) and ownership of the very same service. This separation is what makes security essential. The ability of the methodology to model a clear dependency relation between those offering a service (the merchant processing a credit card number), those requesting the service (the bank debiting the payment), and those owning the very same data (the cardholder), make security solutions emerge as a natural consequence of the modeling process.

1 Introduction

"... Is there such a thing anymore as a software system that doesn't need to be secure? Almost every software controlled system faces threats from potential adversaries, from Internet-aware client applications running on PCs, to complex telecommunications and power systems accessible over the Internet, to commodity software with copy protection

* Research partly sponsored by the FIRB-ASTRO Project.

> mechanisms. Software engineers must be cognizant of these threats and engineer systems with credible defenses, while still delivering value to customers. ... security concerns must inform every phase of software development, from requirements engineering to design, implementation, testing and deployment..."

In 2000, Devambu and Stubblebine introduced with these words their ICSE article on security and software engineering [5]. The article marked a surfacing need in the IT community: security is not just about securing protocols and communication lines, it is also about software [1,16]. Indeed, the need of securing software is even more pressing than the need of securing communication. Almost no attack has been reported in the literature where hackers harvesting credit card numbers by snooping communication lines, whereas exploits of software security bugs are constantly among the headlines [20].

It has also clearly emerged that security concerns must be tackled from the very beginning because looking at them as an afterthought often leads to problems [1,16]. In their ICSE article, Devambu and Stubblebine posed a challenge of integrating security concerns with requirements engineering, and in particular with UML.

Part of this challenge has been answered, and indeed we have a number of proposals for UML models that incorporate security features [9,8,10,7,13], as well as early requirements models of security concerns [17,22,12]. What is still missing is capturing the high-level security requirements, without getting suddenly bogged down into security solutions or cryptographic algorithms. If we look at the requirement refinement process of many proposals, we find out that at certain stage a leap is made: we have a system with no security features consisting of high-level functionalities, and then the next refinement shows encryption, access control, authentication and the like. The need for these features is indeed explained by the English text but this is hardly satisfactory. The modelling process itself should make it clear why encryption, authentication or access control are necessary.

In this paper we propose a solution that is based on augmenting the i*/Tropos framework [4,21] to take into account security considerations. Our decision to augment the language has been mainly driven by a major case study, the modelling of the Secure Electronic Transactions e-commerce suite[3] by VISA and MasterCard [14,15] that one of us has contributed to formally verify [2,3]. The industrial relevance of the case study is clear but the topic is challenging also for technical reasons. At first because the proposal is accompanied by a massive documentation spanning from an high-level business description to bit-oriented programming guide. However, if we look to the documentation we find out that the business case is described in a totally informal way and the programming guide is fairly operational, in many points a good example of bit-oriented programming. It is not possible to trace back the requirements from the 1000+

[3] A new proposal superseding SET and encompassing also features related to smartcards, the VISA and Mastercard 3-D Secure initiative, has been launched this year.

pages of the system description, except from the English text. In particular, the documentation contains no such thing as a UML process model.

Analysis has shown that the key to modelling security features is the separation of the notion of offering a service (of a handling data, performing a task or fulfilling a goal) and ownership of the very same service. Our enhancement of the Tropos/i* methodology is based exactly on this idea.

The ability of the methodology to model a clear dependency relation between those offering a service (for example, the merchant processing a credit card number), those requesting the service (e.g., the bank debiting the payment), and those owning the very same data (the cardholder), is what make security solutions emerge as a natural consequence of the modelling process. Indeed, our proposal ensures that security mechanisms are not introduced early on into a software system design. We believe that this early introduction is what creates gaps in the requirements analysis process and makes unclear the reason behind the introduction of these security mechanisms.

In the rest of the paper we give a quick introduction to the SET e-commerce suite (Section

2 A SET Primer

With the boom of internet shopping in the mid 90s, a consortium led by credit card companies (VISA and Mastercard) and major software vendors (Netscape and Microsoft among them) have put forward an architecture for securing electronic payments.

The need for a secure solution payment solution was spurred by the current unsatisfactory protection offered to customers and merchants dealing on-line. Indeed, people normally pay for goods purchased over the Internet using a credit card. Customers give their card number to the merchant, who claims the cost of the goods against it. To prevent eavesdroppers on the net from stealing the card number, the transaction is encrypted using the SSL protocol. Basically, this is what happen when the browser shows the closed lock on the screen. However this arrangement has many serious limitations:

- The cardholder is protected from eavesdroppers but not from dishonest merchants (pornographers have charged more than the advertised price, expecting their customers to be too embarrassed to complain), nor incompetent ones (a million credit card numbers have been stolen from Internet sites whose managers had not applied security patches) [20].
- The merchant has no protection against dishonest customers who supply an invalid credit card number or who claim a refund from their bank without cause. In most countries, legislation shields customers rather than merchants. So, upon receiving a claim, the banks will go to the merchant asking for a signed receipt and if it does not exist (as it is obviously the case), tough luck for the merchant.

The proposal of the consortium, called SET (Secure Electronic Transactions) aims to “provide confidentiality of information, ensure payment integrity and

authenticate both merchants and cardholders", according to the *Business Description* [14, p. 3].

SET's participants are *cardholders* and *merchants*. Their financial institutions are called *issuers* and *acquirers* respectively. Other participants are *payment gateways* (PG), who play the traditional role of clearing-houses in settling the payment requests made by merchants and cardholders during a purchase. There is also a hierarchy of *certificate authorities* (CA), rooted in a trusted *root certificate authority* (RCA).

Cardholder Registration. This is the initial phase for cardholders. It lets each cardholder register by providing him with a certificate for his signature key. A cardholder begins a session by sending his credit card details to a CA, which replies with a registration form. The cardholder completes the form and choose an asymmetric key pair and send the public part to a CA. The CA checks whether the request is valid (the protocol does not define how) and issues a certificate, signed by the CA, that binds the public key to the cardholder's name. He will have to use the corresponding private key to sign purchase requests. Few points are noteworthy:

- A merchant should not be able to verify a cardholder's account details from his certificate [14, pp. 7, 12 and 25]. Indeed, the cardholder's certificate does not store the account detail. The name of the certificate holder in the X.509 certificate standard [15, pp. 210, 213] is replaced by the hash of his *primary account number* (PAN), loosely speaking the credit card number, and of a secret nonce (PANSecret).
- Still the certificates must assure the merchant during the payment phase (without his having to see the PAN) that a link exists between a cardholder, and a valid PAN, and that the link was validated by the card issuer [14, pp. 8 and 25].

Merchant Registration. This phase performs the analogous function for merchants. Unlike cardholders, merchants register both a signature key and an encryption key. Each merchant gets two certificates.

Purchase Request. The cardholder sends the order information and the payment instructions to a merchant, who may run Payment Authorization (see below) before accepting the order. A trusting merchant can batch the payment authorizations and run them later. It is worth noticing that the cardholder invokes Purchase Request *after* he has agreed with the merchant to buy certain goods or services (the Order Description) for a certain amount. SET is concerned with payment, not with shopping.

Payment Authorization. When a merchant receives an order, he does not receive the cardholder's PAN. So, he cannot just use that number, as done with credit card transactions conducted via the telephone [19], to settle directly with the card issuer. Instead, the merchant forwards the payment instructions to a PG. The PG, in cooperation with the card issuer, checks that everything is fine, and sends the payment authorization to the merchant.

Payment Capture. The merchant sends to the PG one or more payment requests and the corresponding "capture tokens" obtained during the previous steps. The PG checks that everything is satisfactory and replies to the merchant. The actual funds transfer from the cardholder to the merchant is done outside the protocol. This step, for the finalization of the trasnfer of money from the bank to the merchant is explained by a business need: payments can be authorized on-line but captured (finalised) much later, such as at time of delivery. Another example comes from "Buy on Christmas, Pay on Easter" schemes or the like. All we expect is that payment is captured only if previously authorized.

3 The Tropos/i* Methodology for Requirement Analysis

We start by selecting the appropriate methodology for analysis. Among the competing alternatives we have chosen the Tropos/i* methodology, which has been already applied to model some security properties [12,22].

Tropos [4,21] is an agent-oriented software system development methodology, tailored to describe both the organisational environment of a system and the system itself, employing the same concepts throughout the development process. Tropos adopts the i* modelling framework, which uses the concepts of actor, goal, soft goal, task, resource and social dependency for defining the obligations of actors (dependees) to other actors (dependers). Actors have strategic goals and intentions within the system or the organisation and represent (social) agents (organisational, human or software), roles or positions (that represent a set of roles). A goal represents the strategic interests of an actor. In Tropos, we differentiate between hard (hereafter just "goals") and soft goals. The latter have no clear definition or criteria for deciding whether they are satisfied or not. A task represents a way of doing something. Thus, for example a task can be executed in order to satisfy a goal. A resource represents a physical or an informational entity. finally, a dependency between two actors indicates that one actor depends on another to accomplish a goal, execute a task, or deliver a resource.

These modelling concepts are particularly well suited to model business security requirements, which are usually expressed in natural language using notions such as agents and high level goals such confidentiality and authentication. Tropos features make also possible an explicit modelling of security as requirements without requiring an immediate discussion of sizes of cryptographic keys or deny vs access configuration of files.

The distinctive characteristic of Tropos (covering the very early phases of requirements analysis) allows for a deeper understanding of the environment where the software must operate, and of the kind of interactions that should occur between software and human users. This understanding is often essential for having security features right [1,16]: technically perfect security solutions are typically hampered by lack of understanding of interactions. By considering early phases of the requirements analysis process, we can capture not only the *what* or the *how*, but also the *why* a piece of software is developed. This, in turn, supports

Fig. 1. (a) Merchant-Cardhodler Basic Dependencies; (b) Certification actor diagram

Fig. 2. Rationale diagram

a more refined analysis of system dependencies, covering both functional and non-functional requirements. Security requirements can, of course, be seen as a particular form of non-functional requirements.

4 Some Examples of Modelling Features

Let's now start from a simple example of a dependency in SET using plain Tropos.

Example 1 (Fig.
The Cardholder depends on the Merchant for obtaining some goods and the Merchant depends on the Cardholder for obtaining the payment.

The above example is fairly high level: a sort of first principle. We may want to model more complicated examples on interdepencies looking at some subsystems, for example Cardholder Registration.

Example 2 (Fig.
A Cardholder depends on the Certification Authority for the emission of certificates.The Authority itself depends on the Cardholder's credit card Issuer for authorization data. The Issuer of the credit card delegates to the local bank the management of customer relations, so that the Cardholder also depends on the local bank for the desired credit card services.

Another interesting feature of Tropos is the refinement analysis and the usage of rationale diagrams that explains dependencies.

Example 3 (Fig.
Analyzing the goal of the emission of the certificate from example

5 First Attempt at Modelling SET with Tropos/i*

From an ideal perspective, we would like to start from the high level goal of electronic payments and show why the protocol itself is a possible solution and to what goals. So we start by refining the example

Fig. 3. Payment-2 actor diagram

We can further refine this example as follows:

Example 4. The Cardholder depends on the local Bank for banking services and the Merchant for obtaining some goods. The local Bank depends on the Issuer for credit card transaction and the Issuer depends on the Bank for customer relations. The Merchant depends on his Bank for obtaining the payment and the Bank depends on the Acquirer for payment of credit card transaction and the Acquirer depends on the Bank for customer service relations.

The next step clarifies the relations between various actors in the financial setting of the transaction.

Example 5 (Fig.

The Cardholder depends on the local Bank for banking services and the Merchant for obtaining some goods. The local Bank depends on the Issuer for authorizing billing credit card transaction and the Issuer depends on the Bank for customer relations. The Merchant depends on his Bank for obtaining the payment and the Bank depends on the Acquirer for authorizing crediting of card transaction and the Acquirer depends on the Bank for customer service relations. The Issuer depends on the Acquirer for obtaining financial data for the transaction and viceversa.

Fig. 4. Payment-3 diagram

Further analysis of the financial data between Acquirer and Issuers reveals that this is the client credit card number, the merchant account and the payment information.

At this stage one may jump to the conclusion that now we need to encrypt the credit card number so that the merchant cannot see it. Though possible, this conclusion, would not be immediately nor easily derivable from the present analysis. The distance between the analysis and the proposed solution is far too great. In Tropos, we could precisely model the protocol, but without explaining *why* encrypting a credit card number is a good solution and especially a solution for *which* problem.

6 Security-Enhanced Tropos

We now focus on what is missing in the "vanilla" version for Tropos. Looking at the above example, it is clear that we need to distinguish between the *servers* that manipulate some data (e.g., the Merchant) and the *owners* to whom the data ultimately belongs (e.g., the Cardholder). Indeed, inasmuch as the owner of a resource is also disposing of its use we have no need of security analysis. We need security as soon as the owner of some data must use the services of some other agent to use his own data.Looking at operating systems, we need operating system security because the user owns a file but the operating system, an agent distinct from the user, serves access to this file. So, we first extend the Tropos notion of *Actor* as follows:

Fig. 5. Extendend dependency diagram

Fig. 6. Payment Dependency with Ownership of Data

Actor:= **Actor** *name [attributes] [creation-properties] [invar-properties] [actor-goals] [ownership] [services]*

We extend the notion of actor introducing the possibility that an actor can own resources, tasks and goals (*ownership*) and can provide *services* (task, goals and resources). Then we extend the notion of dependency introducing the notion of owner, as follows:

*Dependency:=***Dependency** *name type mode* **Depender** *name* **Dependee** *name* *[***Owner** *name] [attributes] [creation-properties] [invar-properties] [fulfill-properties]*

We can now model situations in which an agent (depender) depends on an another agent (dependee) for a resource (task and goal) and a third agent (owner) gives the permission to use such a resource (task and goal). Of course this includes also the fact that the dependee is able to provide the resource (task and goal), namely the resource is included in its service list, and that the owner owns the resource, namely the resource is included in the owned-resource list.

Figure

The new model makes it possible to analyze the *trust relationship between clients, servers, and owners* and the consequent need for security solution. This is the missing gap between the high-level model and the low-level usage of cryptographic primitives. For example, suppose that we modelled some archival system where the owner of some data must give the data to somebody for storing it. The model will show the dependency relation. If the owner does not trust the storage server, we can change the model and encrypt the data. The trusted relation of the revised model can better address our desired trust relationships.

7 Modelling SET with Security Enhanced Tropos

Using this new ternary dependency we can refine the model presented in Fig.

In the new model, the problem is becoming clear: to obtain the money the Merchant must get a piece of data that belongs to somebody else. Since the Merchant and the Cardholder are not phisically close, the data must be trasmitted from the Cardholder to the Merchant via the network infrastructure. This is modelled in Fig.

Fig. 7. Internet Shopping - Old Style

The analysis of the trust relationships in Fig.

It is worth noting that the same analysis of the trust relationship also shows a potential vulnerability of the system that is not usually considered when discussing the security of internet payment systems. The cardholder must also trust

the phone/network provider between the bank and merchant. This explains the need for additional protection on that side or the usage of a dedicated financial network. So we can devise a refinement alternative to Fig.

This solution can be still unsatisfactory for the cardholder. His data must still be processed by the merchant and there might be cases where this trust relationship is undesirable. Now we can define yet a different refinement in Fig.

Fig. 8. Internet Shopping á la TLS/HTTPS

Fig. 9. Internete Shopping - SET-style

We now have a stepwise refinement process that makes it possible to understand why there is the need of enrypting and protecting credit card details. The refinement steps shows clearly how the trust relation is build, and what trust relations are created. The usage of encrytpion is now clearly motivated by the model itself: we have a owner of data that is forced to use the services of some other entity. Authentication and access control can be similarly introduced during the analysis of the depender-dependee model when the goal require some form of identification.

8 Related Work and Conclusions

A number of articles have discussed the usage of Tropos/i* methodology for modelling security concerns. For example, in [12] the authors show how modelling relationships among strategic actors can be used in order to elicit, identify and analyze security requirements. In particular, how actor dependency analysis helps in the identification of attackers and their potential threats, while actor goal analysis helps to elicit the dynamic decision making process of system players for security issues. [22] shows how goal models can be used to model privacy concerns and the different alternatives for operationalizing it. Using an example in the health care domain, the authors propose a framework based on a catalogue, to guide the software engineer through alternatives for achieving privacy. However, both proposals remain at an abstract level and stops before filling the missing gap that we noted.

A preliminary modification of the Tropos methodology to enable it to model security concerns throughout the whole software development process has been proposed in [18,17]. In particular, this extension proposes the use of security constraints and secure capabilities as basic concepts to be used in order to integrate security concerns throughout all phases of the software development process. Here, we find an operational description but still the notion of constraints is not sufficient to capture the trust relationship that we have been able to model in our enhanced Tropos model.

Down the line in the modelling paradigm, a number of proposals have proposed enhancements to UML to cope with security constraints. [9,10,11] propose an extension of UML where cryptographic and authentication features are explicitly modelled. The model is rich enough to allow for a detailed analysis and indeed has been driven by a similar case study of electronic payment systems [8]. In comparison to ours, the system is fairly low level and is therefore suited to more operational analysis. A challenging line of research may involve the integration of the two methodologies, so that one can start from a high level analysis of the system with our security-enhanced Tropos and then continue down the line to an operational specification using UML.

Another proposal of enhancing UML with security feature is the SecureUML language [7,13] which, however, is geared towards access control. The proposal is focused on providing concrete syntax for representing access control notions in UML so that access control policies can be directly modelled in UML and formal properties derived from that models. These modelling features are essential, but from our perspective only at the end of the system modelling process.

In this paper we have shown an enhancement of Tropos/i* that is based on the clear separation of roles in a dependency relation between those offering a service (the merchant processing a credit card number), those requesting the service (the bank debiting the payment), and those owning the very same data (the cardholder). This distinction makes it possible to capture the high-level security requirements of an industrial case study, without getting immediatley bogged down into cryptographic algorithms or security mechanisms, where purpose is obscured in a morass of details. The modelling process we envision makes it clear why encryption, authentication or access control are necessary and which trust relationships or requirements they address. The challenge is now to integrate this framework with other development phases, to ensure truly secure designs for software systems.

References

1. R. Anderson. *Security Engineering - a Guide to Building Dependable Distributed Systems*. Wiley and Sons, 2003.
2. G. Bella, F. Massacci, and L. C. Paulson. The verification of an industrial payment protocol: The SET purchase phase. In V. Atluri, editor, *9th ACM Conference on Computer and Communications Security*, pages 12–20. ACM Press, 2002.
3. G. Bella, F. Massacci, and L. C. Paulson. Verifying the SET registration protocols. *IEEE Journal on Selected Areas on Communications*, 21(1), 2003. in press.
4. J. Castro, M. Kolp, and J. Mylopoulos. Towards Requirements-Driven Information Systems Engineering: The Tropos Project. *Information Systems*, 2003. Elsevier, Amsterdam, the Netherlands, (to appear).
5. P. T. Devambu and S. Stubbelbine. Software engineering for security: a roadmap. In *Future of Software Engineering. Special volume of the proceedings of the 22nd International Conference on Software Engineering (ICSE 2000)*, pages 227–239, 2000.

6. J.-M. Jézéquel, H. Hußmann, and S. Cook, editors. *5th International Conference on the Unified Modeling Language (UML 2002)*, volume 2460 of *Lecture Notes in Computer Science*. Springer, 2002.
7. J.-M. Jézéquel, H. Hußmann, and S. Cook, editors. *SecureUML: A UML-Based Modeling Language for Model-Driven Security*, volume 2460 of *Lecture Notes in Computer Science*. Springer, 2002.
8. J. Jürjens. Modelling audit security for smart-card payment schemes with UMLsec. In *16th International Conference on Information Security (IFIP/SEC 2001)*. Kluwer AP, 2001.
9. J. Jürjens. Towards secure systems development with umlsec. In *Fundamental Approaches to Software Engineering (FASE/ETAPS 2001)*, LNCS. Springer-Verlag, 2001.
10. J. Jürjens. UMLsec: Extending UML for secure systems development. In Jézéquel et al. [6].
11. J. Jürjens. Using UMLsec and Goal-Trees for secure systems development. In *Symposium of Applied Computing (SAC 2002)*. ACM Press, 2002.
12. L. Liu, E. Yu, and J. Mylopoulos. Analyzing Security Requirements as Relationships Among Strategic Actors. In *Proceedings of the 2nd Symposium on Requirements Engineering for Information Security (SREIS-02)*, Raleigh, North Carolina, 2002.
13. T. Lodderstedt, D. A. Basin, and J. Doser. Model driven security for process-oriented systems. In *8th ACM Symposium on Access Control Models and Technologies*, 2003.
14. Mastercard & VISA. *SET Secure Electronic Transaction Specification: Business Description*, May 1997. Available electronically at `http://www.setco.org/set_specifications.html`.
15. Mastercard & VISA. *SET Secure Electronic Transaction Specification: Programmer's Guide*, May 1997. Available electronically at `http://www.setco.org/set_specifications.html`.
16. G. McGraw and J. Viega. *Building Secure Software*. Addison Wesley Professional computing, 2001.
17. H. Mouratidis, P. Giorgini, and G. Manson. Integrating security and systems engineering: Towards the modelling of secure information systems. In *Proceedings of the 15th Conference On Advanced Information Systems Engineering (CAiSE 2003)*, 2003.
18. H. Mouratidis, P. Giorgini, and G. Manson. Modelling secure multiagent systems. In *Proceedings of the 2nd International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 2003.
19. D. O'Mahony, M. Peirce, and H. Tewari. *Electronic payment systems*. The Artech House computer science library. Artech House, 1997.
20. A. Paller. Alert: Large criminal hacker attack on Windows NTE-banking and E-commerce sites. On the Internet at `http://www.sans.org/newlook/alerts/NTE-bank.htm`, Mar. 2001. SANS Institute.
21. A. Perini, P. Bresciani, F. Giunchiglia, P. Giorgini, and J. Mylopoulos. A Knowledge Level Software Engineering Methodology for Agent Oriented Programming. In *Proc. of the 5th Int. Conference on Autonomous Agents*, Montreal CA, May 2001. ACM.
22. E. Yu and L. Cysneiros. Designing for Privacy and Other Competing Requirements. In *Proceedings of the 2nd Symposium on Requirements Engineering for Information Security (SREIS-02)*, Raleigh, North Carolina, 2002.