E-Commerce Security

A New Methodology for Deriving Effective Countermeasures Design Models

By

Victor D. Sawma

A thesis submitted to

The Faculty of Graduate and Postgraduate Studies

In partial fulfillment of the degree of

Master of Computer Science

School of Information Technology and Engineering

University of Ottawa

(Ottawa-Carleton Institute for Computer Science)

To my father Doumit and my mother Mary

To my brothers Mark and Vany

To my sister Maguy

&

To my fiancée Petty

Acknowledgements

I would like to thank the many individuals who have made this thesis possible. I am especially grateful to my supervisor, Dr. Robert L. Probert, who helped choose the thesis topic and provided guidance, valuable support and encouragement during this research.

My thanks also go to the faculty, staff, and friends of Notre Dame University, especially to Fr. Boutros Tarabay and Mr. Abdallah Abi Aad, for their support and encouragement while pursuing my graduate education.

I would also like to thank Dr. Stan Matwin from the University of Ottawa and Dr. Tony White from the University of Carleton for their valuable suggestions and comments. I would also like to thank Louise Desrochers and Marie-Jo Worral for their unconditional help.

My thanks also go to IBM Canada ltd., especially Joe Wigglesworth, Terry Lau, Paul Sims, Maurus Cappa, Behrad Ghazizadeh, and Philip Day, for their support, comments, suggestions, and encouragement throughout this research.

I would also like to thank Spiro Bittar, Tarek Rahal, Claude Jabre, and Rabih Keserwani for their unconditional support and help while implementing and testing case studies conducted throughout this research. My thanks also go to Wael Hassan and Nour El-Kadri for their unconditional help and valuable suggestions.

My deepest thanks go to my family (my father Doumit, my mother Mary, my brothers Mark and Vany, and my sister Maguy), whose support and encouragement have helped me pursue my education.

Last but not least, my deepest love goes to my fiancée Petty for her love, support, sacrifice, and understanding during the whole period that went into pursuing my graduate studies, conducting this research, and writing this thesis.

Table of contents

ABSTRACT

Designing secure e-commerce systems is one of the most important challenges for security architects. In 1999, the Common Criteria Redbook document was released as a standard for security requirements and evaluation in the Information Technology (IT) domain. Very recently, the National Institute for Standards and Technology (NIST) released a standard for Underlying Technical Models for IT Security based on the Common Criteria Redbook.

Security architects may consider adopting the NIST standard for designing secure e-commerce systems; however, e-commerce is only a sub-domain of IT and the NIST models are too general in nature. Thus, there is a need for specializing the NIST models to directly apply to designing secure e-commerce systems.

The security standards described above emphasize providing proper security features for IT systems by taking proper security measures. However, implementation guidelines for these features are not specified, and testing for the presence of countermeasures against malicious users is postponed to the system testing phase where ethical hacking is applied. This assumes that the system development process includes proper security testing and evaluation before system release.

Moreover, this process of specializing the NIST models and prescribing proper countermeasures against malicious users is ad hoc and relies completely on the expertise of the security architect. This can lead to expensive system development life cycles if defects are discovered during the system testing phase and are traced back to design errors.

As a result, there is a need for a systematic methodology that allows security architects to specialize standard security models (such as the NIST models) for use in e-commerce systems. This methodology must enable security architects to provide proper security

measures for legitimate users and prescribe proper countermeasures against malicious users at system design time.

In this thesis, we address this problem by providing a systematic methodology for deriving countermeasures design models for e-commerce systems based on the NIST security services model. We introduce and illustrate a new term, *malicious user* requirements, which denotes the features which if present enable malicious security attacks to succeed. Our methodology targets security by satisfying legitimate user requirements and blocking *malicious user requirements* at system design time.

Furthermore, we evaluate our methodology by means of a case study. In particular, we derive four security countermeasures design models; namely authentication, authorization, access control enforcement, and transaction privacy models. These derived models are shown to be effective against all known security attacks in the e-commerce domain. Then, we show how to instantiate these models into a SET-integrated existing e-commerce design in order to illustrate the adequacy of our methodology to "design-in" security before implementation, rather than waiting until after implementation and discovering the presence of costly security design errors.

Although we have only demonstrated the value of our approach for four security features, we believe that, in a similar way, our methodology can be applied to the remaining ten features in the NIST standard. We believe that our methodology will be a contribution both to E-commerce security standards, and to better industry practice.

List of Figures

Chapter I

Introduction and Background

1.1. Introduction and background

The use of e-commerce has grown exponentially in recent years [EUC]. This growth requires a comparable growth in security presence (define, design, implement and test a secure environment throughout the e-commerce system). Until recently, best practices for e-commerce (EC) security development were based solely on the expertise of individual designers ([Brinkley, 1995-1] and [Brinkley, 1995-2]). Until 2001, there was no published security design standard by IT standards bodies such as ISO's Standing Committee 27 (SC 27) [ISOSC 27]. These standards bodies are responsible for developing standards for Information Technology and, thus, do not deal with e-commerce security in particular. Best practices for e-commerce security only developed in an ad hoc manner as new security issues were encountered and overcome, and were not applied uniformly across even an organization, let alone the industry. [NIST, 2001]

1.2. Motivation – Need for a New Approach

Recently, some documents have been developed which address the lack of a common security design standard. The pitfalls of using these documents in an ecommerce system design process are described below and are detailed further in chapter 2.

1.2.1. Common Criteria Redbook Pitfalls

The Common Criteria (CC) Redbook [CC, 1999] was one of the first attempts to standardize security assessment/evaluation requirements for Information Technology (IT) systems.

CC is intended for security evaluation in IT systems. [CC, 1999] While it is possible to use CC as a reference for designing security in EC systems, the corresponding design process is not a systematic one and relies completely on the individual security designer expertise. A possible risk is not introducing the proper security countermeasures during the design phase. In this case, defects will either be discovered during the system testing phase, requiring an expensive fix, or not discovered at all.

1.2.2. NIST Security Services Model Pitfalls

The National Institute for Standards and Technology (NIST) released a document entitled "Underlying Technical Models for Information Technology Security" [NIST, 2001] which describes models and recommendations for designing security in IT systems based on the CC Redbook. Still, using the NIST security services model alone does not guarantee the security of an EC system, for example.

While this model is intended as a general model for IT systems, extending it for EC systems is similar to using CC to guide design since the process is not systematic and relies completely on the security designer expertise. Yet, it is much easier to extend the NIST security services model rather than building a security model for ecommerce systems from scratch. This is simply because the NIST security services model is based on the Common Criteria Redbook. Thus, an extension to the NIST security services model has a better chance of satisfying standard security requirements provided in CC. This is our approach in this thesis.

1.2.3. Open Source Security Testing Methodology Manual Pitfalls

The Open Source Security Testing Methodology Manual (OSSTMM) [Herzog, 2001] attempts to set a standard for security testing on a running Internet system. OSSTMM can be used to test the Internet part of a running EC system. Yet, it cannot be used directly during the system design phase.

In our research, we have used OSSTMM as an important, but not unique, reference for some possible security attacks on EC systems.

1.2.4. Current Industrial Practices Pitfalls

Current industrial practices introduce security countermeasures for discovered defects during the system-testing phase. Any defect discovered during this phase can be traced back to an implementation error or a system design error. Design errors are much more expensive to fix during this phase than if they were discovered during the design phase. [Treese, 1998] From our experience, this problem is well recognized in industry.

1.3. Problem Statement

Security requirements can be divided into two categories: *legitimate user requirements* and *malicious user requirements* (part of our new methodology). *Legitimate user requirements* are requirements that allow legitimate users to use the system in a safe manner. Yet, no security is perfect [NIST, 1999]. *Malicious user requirements*, on the other hand, are requirements that allow malicious users to succeed in breaking system security, i.e. security satisfying legitimate user requirements. Thus, a system can be considered secure if it provides enough security measures to satisfy legitimate user requirements and enough countermeasures to block malicious user requirements that lead to successful security attacks against the implemented system security.

At the time of writing this thesis, the process of introducing countermeasures against known security attacks during an EC system design phase relied completely on the security designer expertise. Moreover, the countermeasure selection process is ad hoc and, thus, a prescribed countermeasure at system design time might prove inadequate during the system-testing or post-release service phases. This might result in an expensive life cycle for fixing defects related to the EC system design model.

Thus, our Problem Statement is:

How can appropriate security features, attributes, safeguards, and countermeasures be "designed in" to an e-commerce system under development, rather than waiting until the system is implemented and conducting a costly "test and rework" campaign?

In this thesis, we will describe a systematic methodology for introducing security countermeasures in EC systems at design time. Our methodology is based on the NIST security services model for IT systems. OSSTMM, as well as other resources, are used as points of reference for classifying the different types of security attacks that can be executed on the security features – a feature being a property or service - in the NIST security services model. The result of applying these attacks through our methodology to the security features in the NIST security services model is a specialized set of security countermeasures design models that are directly useful for designing secure EC systems. In the second phase of our methodology, we show how to apply and integrate these models into an existing (SET-integrated) e-commerce system design.

1.4. Contributions of Thesis

The contribution of this thesis can be summarized as follows:

1. We derive a comprehensive matrix mapping all known security attacks to four NIST

security features in e-commerce systems; namely authentication, authorization, access control enforcement, and transaction privacy.

2. We provide four new security models for e-commerce systems that specialize four security features from the NIST security services model; namely authentication, authorization, access control enforcement, and transaction privacy.
3. For the set of features related to security that are investigated in this thesis - namely authentication, authorization, access control enforcement, and transaction privacy – a faithful implementation of a feature model – referred to as a countermeasures design model - is shown to block all known security attacks related to that feature. The main contribution is in avoiding expensive fixes, related to the security feature, during the testing phase. This is achieved by having an effective countermeasures design model that is directly applicable to EC systems. This model specifies detailed requirements for the security feature. In other words, any defect discovered during the testing phase will be related to security implementation. This is relatively cheaper to fix than a defect related to the system security design model.
4. We contribute a cost-effective, systematic methodology for deriving countermeasures design models for the other security features of e-commerce systems.
5. We provide a catalog of all known security attacks related to the four security features discussed in this thesis; namely authentication, authorization, access control enforcement, and transaction privacy.

1.5. Outline of Thesis

This thesis is organized as follows:

In chapter 1 we provide an overview of the literature in this area, and a snapshot of current industrial practices. Chapter 2 shows why we need an EC design methodology and states the problem that our methodology intends to solve. Chapter 3 gives a detailed description of our model-based approach and methodology to derive security countermeasures design models. Chapter 4 applies Phase 1 of our methodology by deriving security countermeasures design models for four security features in the NIST

security services model; namely authentication, authorization, access control enforcement, and transaction privacy. Chapter 5 applies Phase 2 of our methodology to the derived countermeasures design models from Phase 1 through a case study to a SET-integrated e-commerce system that allows its users to select products from a catalog, place orders, submit orders, and pay online. Chapter 6 provides an evaluation of our methodology based on our experience from the two case studies in chapters 4 and 5. Finally, we summarize, conclude, and present future research and work in Chapter 7.

Chapter II

Electronic Commerce Security: Current Standards and Practices Overview

Electronic commerce (EC) is about doing business electronically. It is based on the ability to electronically process and transmit data, including text, sound and video. EC encompasses many diverse activities including electronic trading of goods and services, online delivery of digital content, electronic fund transfers, auctions, direct consumer marketing, and after-sales service. It involves both products (consumer goods) and services (information services), traditional activities (healthcare, education) and new activities (virtual malls).

An example of a generic e-commerce system architecture includes a client communicating with a merchant server via the Internet. The merchant server (also known as business server), usually utilizes a web server to accept client requests, a database management system to manage data and a payment gateway to provide online payment services.

For many years companies have exchanged business data over a variety of communication networks. Now, there is an accelerated expansion driven by the exponential growth of the Internet. EC is now rapidly expanding into a complex web of commercial activities transacted on a global scale between an ever-increasing number of participants, corporate and individual, known and unknown, on global open networks such as the Internet. [EUC]

This explosion requires a comparable growth in security presence. Yet, it is likely that readers of this thesis have heard about successful e-commerce break-ins at some time in the past few years. Such events have made the headlines of newspapers, magazines, and books.

Until 2001, there was no published security design standard encompassing all security features. Best practices developed as new security issues were encountered and overcome. However, improvements to security design strategies did not come from a centralized source such as the NIST security design standard. [NIST, 2001]

This chapter is divided into two sections. Section 2.1 provides an overview of the literature addressing standards for e-commerce security. Section 2.2 describes current industrial practices from the author's experience and the trade literature. Section 2.3 describes other related research. Section 2.4 states our solution requirements and rationale. Finally, we conclude in Section 2.5.

2.1. Current Security Standards

This section gives an overview of current security standards. International standards bodies such as [ISOSC 27] do not provide standards for overall security design and are basically oriented towards specific security topics such as providing non-repudiation, defining encryption algorithms, describing operation and management of intrusion detection systems, etc. More information on the topics that these standards bodies address can be found in [ISOSC 27]. As mentioned earlier, there are currently three security standards for IT systems that address overall system security: the Common Criteria Redbook (CC) [CC, 1999], the NIST "Underlying Technical Models for Information Technology Security" document [NIST, 2001], and the Open Source Security Testing Methodology Manual (OSSTMM) [Herzog, 2001].

In section 2.1.1, we provide an overview of the Common Criteria Redbook. In section 2.1.2, we provide an overview of the NIST "underlying technical models for IT security"

document. Section 2.1.3 describes the NIST security services model which is at the basis of our methodology and approach. In section 2.1.4, we provide an overview of the Open Source Security Testing Methodology Manual. Section 2.1.5, summarizes the current state of security standards.

2.1.1. Common Criteria Redbook [CC, 1999]

Many consumers of IT lack the knowledge, expertise or resources necessary to judge whether their confidence in the security of their IT products or systems is appropriate, and they may not wish to rely solely on the assertions of the developers. Consumers may therefore choose to increase their confidence in the security measures of an IT product or system by ordering a trusted third-party analysis of its security (i.e. a security evaluation).

The Common Criteria (CC) Redbook was one of the first attempts to standardize security assessment / evaluation requirements for Information Technology (IT) systems. It can be used to select the appropriate IT security measures and contains criteria for evaluation of security requirements.

At the time of the writing of this thesis, CC version 2.1 was released and consisted of the following three parts:

1) Part 1, *Introduction and general model*, is the introduction to the CC. It defines general concepts and principles of IT security evaluation and presents a general model of evaluation. Part 1 also presents constructs for expressing IT security objectives, for selecting and defining IT security requirements, and for writing high-level specifications for products and systems. In addition, the usefulness of each part of CC is described in terms of each of the target audiences.
2) Part 2, *Security functional requirements*, establishes a set of security functional components as a standard way of expressing the security functional requirements for Targets of Evaluation (TOEs) or systems.

3) Part 3, *Security assurance requirements*, establishes a set of assurance components as a standard way of expressing the assurance requirements for TOEs. Part 3 also defines evaluation criteria for Protection Profiles (PPs) and Security Targets (STs) and presents evaluation assurance levels that define the predefined CC scale for rating assurancefor TOEs, which is called the Evaluation Assurance Levels (EALs).

However, in the domain of e-commerce systems, the Common Criteria Redbook (CC) is intended to be used to evaluate system security only after the system is implemented. It cannot be used directly during the system design phase. This means that CC does not address the core problem of Section 1.3, namely the lack of a security design-time standard for e-commerce systems.

2.1.2. NIST Underlying Technical Models for IT Security [NIST, 2001]

The purpose of the NIST "Underlying Technical Models for Information Technology Security" document is to provide a description of the technical foundations, termed 'models', which underlie secure information technology (IT). The intent is to provide, in a concise form, the models that should be considered in the design and development of technical security capabilities. These models encompass lessons learned, good practices, and specific technical considerations.

The security goal according to NIST can be met by achieving five security objectives:

1) *Availability* of systems and data for intended use only. This is a requirement to assure that systems work promptly and service is not denied to authorized users and denied for unauthorized / malicious users.
2) *Integrity* of system and data. This objective has two facets:
   a) Data integrity: the property that data has not been altered in an unauthorized manner while in storage, during processing, or while in transit.
   b) System integrity: the quality that a system is performing the intended function in an unimpaired manner, free from unauthorized manipulation.

3) *Confidentiality* of data and system information. This is the requirement that confidential information not be disclosed to unauthorized individuals. Confidentiality protection applies to data in storage, during processing, and while in transit.
4) *Accountability* to the individual level. This is the requirement that actions of an entity can be traced uniquely to that entity.
5) *Assurance* that the other four objectives have been adequately met. This objective is the basis for confidence that the security measures, both technical and operational, work as intended to protect the system and the information it processes. The other four security objectives are adequately met by a specific implementation when:
   a) there is correct implementation of the required functionality
   b) there is sufficient protection against unintentional errors
   c) there is sufficient resistance to intentional penetration or by-pass

   Assurance is a continuum; the amount of assurance needed varies between systems.

In addition to defining the security objectives and existing interdependencies between them, this document also presents the NIST security services model which is at the basis of our work. A detailed description of the model is presented in Section 2.1.3.

This document satisfies the lack for a security design standard in IT systems. Yet, it addresses security in IT systems from a general point of view and does not provide specific security implementation details and guidelines for e-commerce systems. This is natural because the IT domain includes a wide range of inter-related sub-domains and e-commerce is one of them. Therefore, this document cannot be used directly in designing e-commerce systems and, thus, does not solve the core problem motivating this thesis, namely the lack of a security design-time standard approach.

2.1.3. NIST Security Services Model [NIST, 2001]

The NIST security services model is depicted in Figure 2.1 and shows the primary services and supporting elements used in implementing an information technology security capability,

along with their primary relationships. The model also classifies the services according to their primary purpose as follows:

- **Prevent.** These services focus on preventing a security breach from occurring
- **Recover.** The services in this category focus on the detection and recovery from a security breach.
- **Support.** These services are generic and underlie most information technology security capabilities.

2.1.3.1. Prevention Services

Prevention services are intended to prevent a security breach from ever happening. [NIST, 2001]

- *Protected communications:* In a distributed system, the ability to accomplish security objectives is highly dependent on trustworthy communications. The *protected communications service* ensures the *integrity*, *availability*, and *confidentiality* of information while in transit. In most situations all three elements are essential requirements, with *confidentiality* being needed for authentication information.
- *Authentication:* Ensuring that a claimed identity is valid is extremely important. The *authentication service* provides the means to verify the identity of a subject.
- *Authorization:* The *authorization service* enables specification and subsequent management of the allowed user or process actions for a given system.
- *Access control enforcement:* When the subject requesting access has been validated for access to particular processes, enforcing the defined security policy is still necessary. Access control enforcement provides this enforcement. Frequently, the enforcement mechanisms are distributed throughout the system. It is not only the correctness of the access control decision, but also the strength of the access control enforcement mechanism that determines the level of security achieved.
- *Non-repudiation:* System accountability depends upon the ability to ensure that senders cannot deny sending information and that receivers cannot deny receiving it. *Non-repudiation* is a service that provides prevention and detection. This service has been placed into the prevention category because the mechanisms implemented prevent the

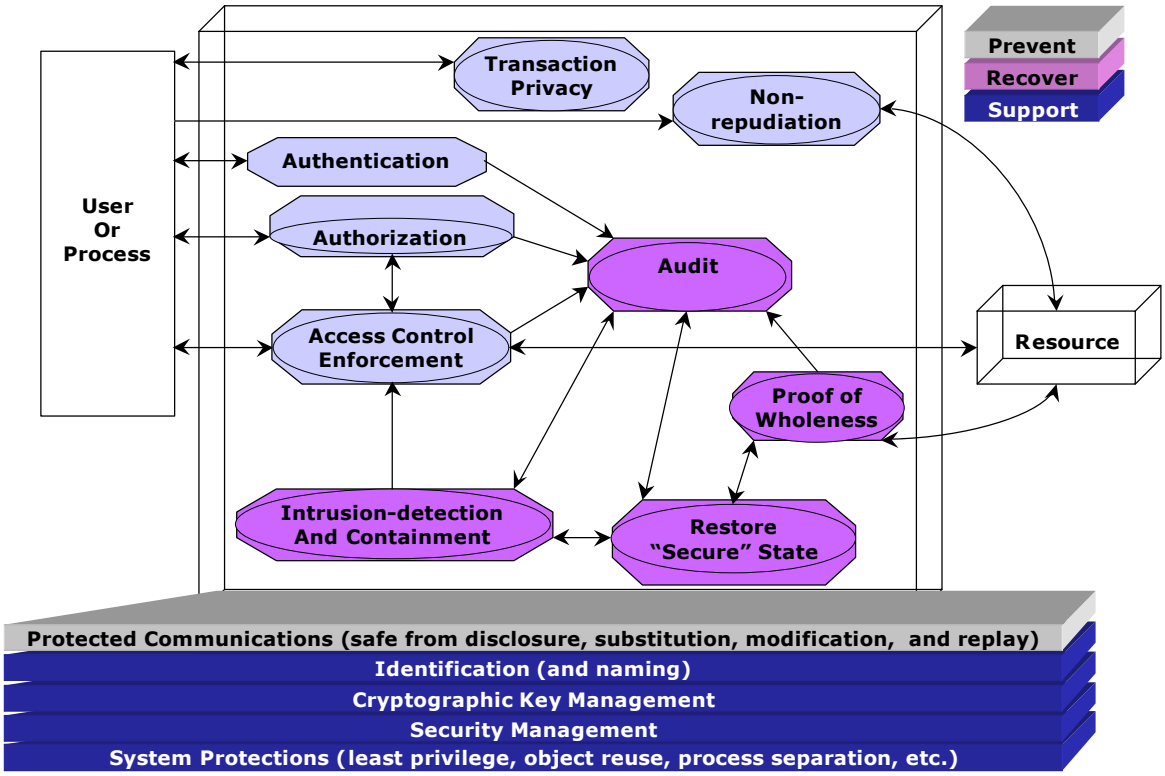

Figure 2.1. NIST underlying technical security services model for IT systems.

ability to successfully repudiate an action. As a result, this service is typically performed at the point of transmission or reception.

- *Transaction privacy:* Both government and private systems are increasingly required to maintain the privacy of individuals using these systems. The *transaction privacy service* protects against loss of privacy of an individual.

2.1.3.2. Detection and Recovery Services

Because no set of prevention measures is perfect, it is necessary to both *detect* security breaches and to take actions to *recover from and reduce* their impact.

- *Audit:* Auditing security-relevant events is a key element for detection of and recovery from security breaches.
- *Intrusion detection and containment:* Detecting insecure situations is essential in order to respond in a timely manner. Also, detecting a security breach is of little use if no

effective response can be initiated. The intrusion detection and containment service provides these two capabilities.

- *Proof of Wholeness:* In order to determine that integrity has been compromised, the ability must exist to detect when information or system state is potentially corrupted. The proof of wholeness service provides this ability.
- *Restorability to a 'secure' state:* When a security breach occurs, the system must be able to return to a state that is known to be secure. That is the purpose of this service.

2.1.3.3. Supporting Services

Supporting services are pervasive and inter-related with many other services. The NIST supporting services are:

- *Identification (and naming):* It is essential that both subjects and objects be uniquely identifiable in order to implement many of the other services. This service provides the capability of uniquely identifying users, processes, and information resources.
- *Cryptographic key management:* When cryptographic functions are implemented in various other services, cryptographic keys must be securely managed.
- *Security administration:* In order to meet the needs of a specific installation and to account for changes in the operational environment, security features of the system need to be administered. As an example, a password database might make use of the operating system platform API in one installation and a SQL database in another installation.
- *System protections:* Underlying the various security functional capabilities is a base of confidence in the technical implementation. This represents the quality of the implementation from both the perspective of the design processes used and the manner in which the implementation was accomplished. Some examples of *system protections* are: residual information protection (also known as object reuse), least privilege, layering, process separation, modularity, and minimization of what needs to be trusted (also known as least-trust). For details, see [NIST, 2001]

2.1.4. Open Source Security Testing Methodology Manual [Herzog, 2001]

The Open Source Security Testing Methodology Manual (OSSTMM) attempts to set a standard for Internet security testing on a running system. Although this type of testing cannot be done during the system design phase, the OSSTMM itself will be used throughout our thesis as a reference for security attacks. An important aspect of the methodology is the way interdependencies are mapped and related to each other. This mapping of interdependencies will help us in Chapters 4 and 5 to map and relate our methodology components with each other. We use this mapping information to "design in" security safeguards in our approach.

A *security test* is performed with two types of *attacks* according to OSSTMM. A *passive attack* is often a form of data collection that does not directly influence the target. An *intrusive attack* however trespasses upon the target and can be monitored, logged, and used to alarm the target. [Herzog, 2001] The process (or methodology) of a security test concentrates on evaluating areas that directly affect security presence. These areas are the following:

- *Visibility:* Visibility is what can be seen, logged, or monitored in the security presence both with and without the aid of electronic devices. This includes, but is not limited to, communication devices (such as e-mail) and network packets (such as TCP/IP).
- *Access:* Access is an entry point into the security presence. This can include, but is not limited to, a web page, a network connection, or anything where a computer interacts with another computer within a network.
- *Trust:* Trust is a specialized attribute with respect to security presence as a whole. Trust includes the kind and amount of authentication, non-repudiation, access control, accountability, confidentiality, and integrity between two or more features within the security presence.
- *Alarm:* Alarm is the timely and appropriate notification of activities that violate or attempt to violate visibility, access, or trust. In many security breaches, alarm is often the single process which initiates further consequences.

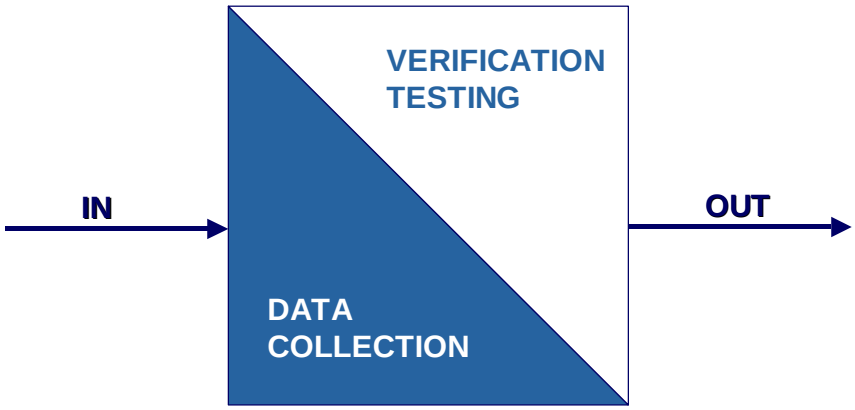

Figure 2.2. A flow representation of modules in OSSTMM.

The methodology starts with a *security map* of the system and is broken down into *sections*, *modules* and *tasks*. The *sections* are specific points in the *security map* which overlap with each other. The *modules* represent the *flow* of the methodology from one *security presence point* to the other. Each module has an *input* and an *output* as seen in Figure 2.2. The *input* is the information used in performing each task. The *output* is the result of completed tasks. [Herzog, 2001]

Output may or may not be analyzed data (also known as intelligence) to serve as an input for another module. It may even be the case that the same output serves as the input for more than one module or section. Some tasks yield no output; this means that modules may exist for which there is no input. Modules which have no input can be ignored during testing. Ignored modules do not necessarily indicate an inferior test; rather they may indicate superior security. [Herzog, 2001] This is because the input of one module depends on the output of another. Having no output from a certain module can either be traced back to an inferior test in the case of weak security testing, or to superior security in the case of having proper security countermeasures.

The methodology flows from the initial module to the completion of the final module. It allows for a separation between data collection and verification testing of and on the collected data. Each module has a relationship to the one before it and the one after it. Each section has inter-relational aspects to other modules and some may inter-relate with all the

other sections. [Herzog, 2001] Overall, security testing begins with an input that is ultimately the addresses of the systems to be tested. Security testing ends with the beginning of the analysis phase and the final report.

As discussed in this section, OSSTMM applies to the ethical hacking process of a running Internet system. Yet, this can only be done after the system is implemented. OSSTMM is different from the Common Criteria document and the NIST "underlying technical models for IT security" document in the fact that it emphasizes blocking malicious user requirements. Still, it cannot be used directly during the system design phase and, thus, does not solve our core problem, namely the need for methods and tools based on a security design standard in e-commerce systems.

2.1.5. Security Standards Overview Summary

The Common Criteria (CC) Redbook [CC, 1999] presents a standard way of modeling and evaluating security through a checklist of evaluation criteria. This allows for security evaluation after designing, implementing, and testing the system. CC cannot be used directly in designing EC systems. The NIST "Underlying Technical Models for Information Technology Security" document [NIST, 2001] in general, and the NIST security services model in particular is not detailed enough to be directly used in e-commerce systems. Moreover, the process of specializing or extending this model for e-commerce systems relies completely on the expertise of security designers. The Open Source Security Testing Methodology Manual (OSSTMM) presents a standard methodology of testing a system through a set of rules and guidelines for ethical hacking, penetration testing, and information analysis. [Herzog, 2001] However, this type of testing cannot be done during the system design phase and will be used throughout our thesis as a reference for security attacks.

2.2. Current Industrial Practices

Until recently, the term "system security" basically referred to a secrecy-based model in

industry [Herzog, 2001]. In practice, ad-hoc security models have been designed and used in many systems and products. Security fixes and patches remain integral components of the system development life cycle. Security designs were based on previous security designs incorporating acquired experience. Thus, the success of the security design depended entirely on the expertise of the designer.

When the NIST security services model was released, security designers in industry considered adopting it for EC systems. Yet, the NIST security services model is a general model for IT systems and needs to be specialized or detailed to be used in EC systems. This is due to the fact that the NIST security services model does not take into account specific details for EC security such as session management, for example.

Figure 2.3 shows our projection of the NIST security services model onto a typical EC system security development life cycle. The NIST security services model is specialized to become the security model for the system under development (SUD). This step relies completely on the expertise that the system security designer has. The SUD is then implemented and the result is a running EC system ready to be tested and ethically hacked by internal security experts simulating possible attacks as the system under test (SUT).

Meanwhile, the security testing team develops a security test plan based on SUT-specific attacks. OSSTMM [Herzog, 2001] can help in building the security test plan since it organizes the ethical hacking process on the Internet part of the system. More investigation is then done on other EC-related security attacks and the test plan is updated to cover other types of security attacks as well.

Any defect discovered during the test execution phase is ideally traced to a development error or sometimes to a design error. Design errors are much more expensive to fix during this phase than if they had been discovered during the design phase. [Treese, 1998] The actual industrial development process is similar to the one discussed above in the author's experience.

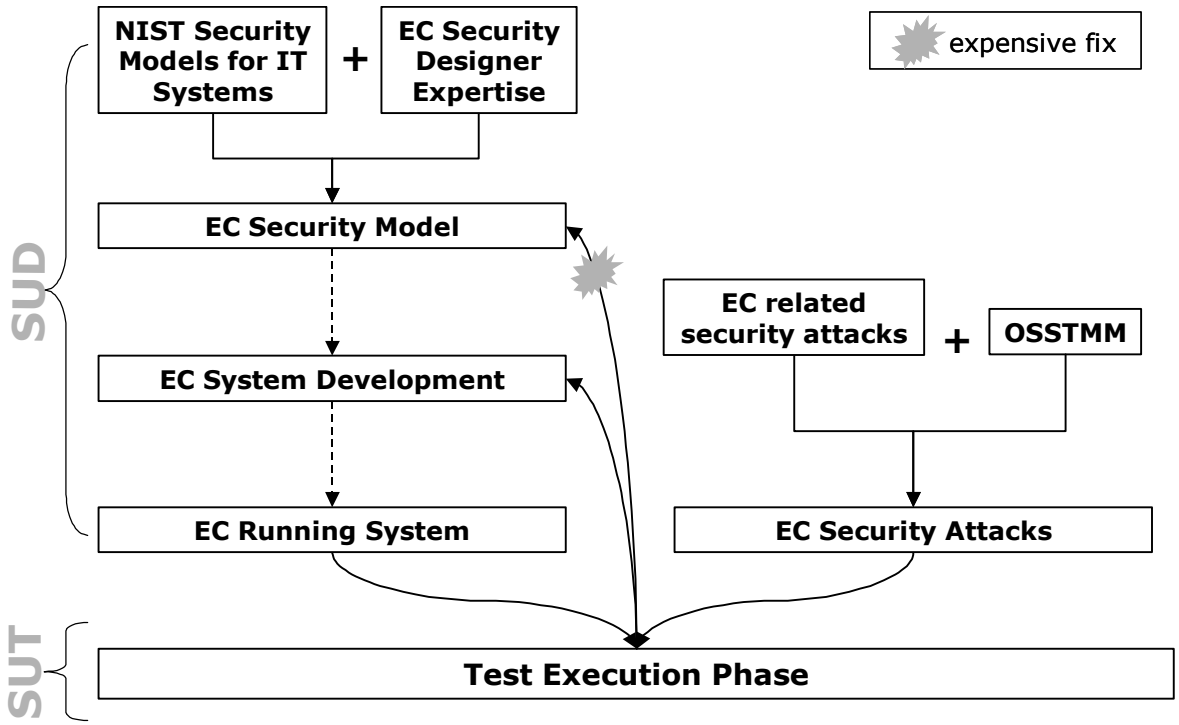

Figure 2.3. Our projection of the NIST security services model onto a typical e-commerce system security development life cycle.

The gap in this approach is that it does not guarantee security against all possible attacks at system design time. Although the required security objectives may be assuredto be correct by being based on the NIST security services model, implementation guidelines and design considerations are not guaranteed to be correct. This is because the details of the security model rely primarily on the security designer expertise. Thus, there is a need for a systematic methodology to designing secure e-commerce systems. Our model-based methodology is one step towards bridging this gap and is described in the next chapter.

2.3. Other Related Research

In this section we will provide a brief overview of other related research in the security domain. In Section 2.3.1 we discuss attack trees and Section 2.3.2 discusses fault tree

analysis. Section 2.3.3 discusses the relationship between attack trees, fault tree analysis, and our research.

2.3.1. Attack Trees and Attack Nets

"An *attack tree* is a formal, methodical way of finding ways to attack the security of a system." [Schneier, 1999] *Attack trees* can therefore be used by security architects to analyze security attacks and prescribe proper security countermeasures.

To create an *attack tree*, a security architect first identifies possible goals of an attack. Each possible goal will be the root node of a tree. The security architect then thinks of all possible attacks, which can be used to reach the goal. These attacks are added to the root nodes to form the first level of leafs on the attack tree. The generated leafs can be seen as sub-goals. For each sub-goal, the security architect thinks of all possible attacks that can be used to reach those goals. By repeating this process, a tree is generated describing, in a detailed way, how the system can be attacked. This tree is called the *attack tree*.

An example of an attack tree for opening a safe is given in Figure 2.4. The goal is to open the safe. Attackers can pick the lock, learn the combination, cut open the safe, or install the safe improperly so that they can easily open it later. To learn the combination, they either have to find the combination written down or get the combination from the safe owner and so on. Each node becomes a sub-goal, and children of that node are ways to achieve that sub-goal. This is just a sample attack tree, and an incomplete one at that.

Note that there are *AND nodes* and *OR nodes*. In the figures, everything that is not an AND node is an OR node. *OR nodes* are alternatives while *AND nodes* represent different steps essential for achieving the goal.

When building a set of attack trees, it is possible, if not probable, that identical nodes in some locations are generated. A node that exists in more than one location will have the same sub-tree everywhere. Thus attack trees support re-use. [Schneier, 1999]

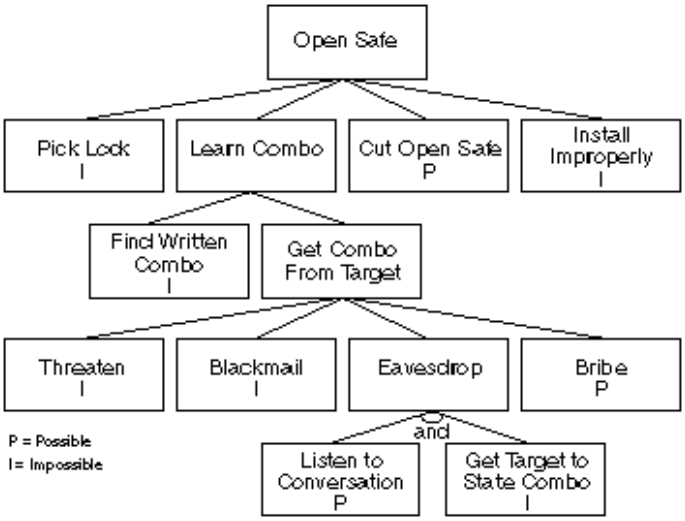

Figure 2.4. An example of an attack tree to open a safe. [Schneier, 1999]

When a sufficient level of detail is reached, a security architect can start assigning values to leafs (or outermost nodes) of the attack tree. An example of assigning values might be the estimated cost of a certain attack. For each leaf, a cost is estimated and assigned. This way, security architects can calculate the minimum cost of reaching the goal, represented by the root. The goal is to make the system more expensive to attack than the actual value of the information to be obtained by the attack. Attack trees are further described and discussed in [Schneier, 1999].

Research has been done on various domains where attack trees can be used and applied. [Phillips, 1998] presents a graph-based approach to network vulnerability analysis.[Steffan, 2002], on the other hand, addresses collaborative attack modeling by combining a graph-based attack modeling technique and a web-based collaboration tool. [Jha, 2002] presents an algorithm for generating attack graphs using model checking.

Another similar related research topic is *attack nets*. [McDermott, 2000] *Attack nets* are very similar to *attack trees* except that *Petri Net trees* are used instead of the simple tree notion introduced in attack trees. For the purpose of our research, attack trees and attack nets are similar since they both allow for designing security on a per-attack basis.

2.3.2. Fault Tree Analysis

A *fault tree* is built up of all sequences of individual component failures, which could make a system stop functioning. The root node of all fault trees is system failure. For each level in the fault tree, events are divided into combinations of sub-events, using logic gates (AND, OR, XOR, etc). The refinement of the fault tree stops when basic events are reached that cannot be further refined. [Fenelon, 1994]

Fault trees are widely used in dependability analysis. With knowledge of the probability of the occurrence of basic events, the probability of system failure can be calculated, using mathematical methods. [Fenelon, 1994]

2.3.3. Relationship to Our Work

Attack trees are used to analyze and model security attacks and allow for cost estimation and countermeasure derivation on a per-attack basis. However, there is no standard for all known security attacks in e-commerce systems. Therefore, using attack trees cannot be applied to design overall security in e-commerce systems. Furthermore, the process of modeling and analyzing security attacks using attack trees relies completely on the security design expertise.

Fault trees are not directly related to our work as they attempt to model system failure. Our work is distinct in that we discuss system security. Fault trees might be helpful in some type of attacks that lead to system failure such as denial of service attacks. However, they cannot be used to design security in general.

Our work is distinct from fault trees and attack trees in that we model security features via a systematic and modular methodology. Our approach has the advantage of using security standards, such as the NIST security services model [NIST, 2001], that define security features for IT systems, and hence, for e-commerce systems. By applying our methodology,

the goal of designing overall security can be achieved by specializing the NIST security features for e-commerce systems in a systematic and modular manner, for example.

2.4. Our Solution Requirements and Rationale

Our discussion of the current state of e-commerce security formulates a problem whose solution requires a methodology or approach that satisfies the following **four requirements**.

1. The solution has to be *systematic* in nature. This is required to avoid relying on the expertise of security designers.
2. The methodology or approach must introduce *security countermeasures* during an EC system design phase. This is required to avoid expensive system development life cycles by minimizing the probability of having defects related to the e-commerce system design model discovered in the system-testing phase.
3. The solution must extend / specialize security features, as described in referenced literature standards [NIST, 2001] and [CC, 1999], and must not break interdependent relationships among the security features. This requirement is important to preserve the validity of the overall e-commerce security design model.
4. The solution must be *extensible*. This is a requirement for the ability to support countermeasures against possible unknown security attacks discovered in the future.

2.5. Summary

In this chapter, we provided an overview of e-commerce security. In section 2.1 we discussed current existing security standards including the common criteria redbook, the NIST underlying technical models for IT security, the NIST security services model, and the Open Source Security Testing Methodology Manual (OSSTMM).

In section 2.2, we discussed current industrial security practices along with a projection of the NIST security services model onto a typical EC system security development life cycle.

In section 2.3 we presented other related research, namely attack trees, attack nets and fault tree analysis. We also discussed the relationship to our work and how our work is distinct.

In section 2.4, we presented our solution requirements and rationale. In addition, we provided justifications for each requirement.

Chapter III

Our Methodology for Deriving and Integrating Countermeasures Design Models for Electronic Commerce Systems

In this chapter, we will describe our proposed methodology for designing secure e-commerce systems. This methodology is claimed to satisfy the requirements listed in section 2.3 and, thus, serves as a solution for the central problem stated in Chapter 1, namely relying completely on a security designer expertise to introduce countermeasures against known security attacks during an EC system design phase; and an ad-hoc process for prescribing security countermeasures at system design time that might prove inadequate during the system-testing phase. In section 3.1 we present an overview of the methodology. Section 3.2 describes Phase 1 of the methodology and how the methodology is organized and discusses relationships between its components. Section 3.3 describes Phase 2 of the methodology. Section 3.4 provides a summary of this chapter.

3.1. Methodology Overview

In this section, we provide a detailed description of our proposed methodology for deriving and applying EC security countermeasures design models from the existing IT standards. As mentioned earlier, the NIST security services model is at the basis of our model. Our goal is to describe a model-based approach of how to extend such a model or "specialize" it in order to apply it to e-commerce systems.

3.1.1. Specific Design Objective: Capture and Block Malicious User Requirements

3.1.1.1. A Unique Aspect of our Approach

A unique aspect of our approach is that it focuses on introducing the appropriate security countermeasures early in the design process [Treese, 1998] while taking into consideration the different types of *malicious users* and the attack power each type might have. In general, malicious users can be grouped into three categories: crackers, intruders, and insiders. A *cracker* is someone who breaks systems for nefarious ends. An *intruder* is someone who gains access into systems by force. An *insider* is a person who is in a position of power or has access to system confidential information. The power such users might wield and the threats they may impose vary from one user category to another. [Viega, 2002] Therefore, security designers, and our methodology in this case, have to take into account these different categories and design countermeasures accordingly.

3.1.1.2. Understanding Malicious Users and their Motivations

Taking into consideration the above-mentioned "malicious user" categories supports our methodology for the systematic introduction of the proper security countermeasures early in the system design process as follows. Every EC system is normally built upon a set of

user requirements. In the context of security, user requirements are: availability, integrity, confidentiality, accountability, and assurance. [NIST, 2001]

Malicious user requirements (MUR), on the other hand, are requirements allowing malicious users to attack the EC system. Our methodology captures these requirements through a new notion: "attack enablers". By capturing these requirements, we will be able to understand malicious users and their motivations. This enables an effective security design that provides effective countermeasures against all known malicious behaviors.

3.1.1.3. A New Design Goal: Block Malicious User Requirements

In a typical EC system life cycle, user requirements are captured during the system design phase and the EC system is designed to satisfy those requirements. Our proposed methodology deals with malicious user requirements (MURs) on a similar basis with the exception that the system design must block these requirements rather than satisfy them.

It is worth noting that MUR can exist in different system components: EC system itself, operating system platform, third party software components, etc. In this thesis, we will deal only with MURs for the EC system at the design phase. MURs and security attacks dealing with operating system platforms, networks, third party software components and system coding (implementation) will not be discussed.

3.1.2. Phases of the Methodology

Our methodology, depicted in Figure 3.1, can be divided into two functional phases. In Phase 1, security features are selected and security-oriented design models are derived and verified. In Phase 2, the derived security-oriented design models from Phase 1 are instantiated and integrated into an existing e-commerce system design.

Phase 1: Select features and derive design models

1 - Select security features.

2 - For each security feature, derive a countermeasures design model:

- 2.1 – Identify and abstract all attacks related to the security feature.
- 2.2 – For each security attack:
  - 2.2.1 – Derive all attack enablers.
  - 2.2.2 – For each attack enabler
    - 2.2.2.1 – Prescribe appropriate *security countermeasures*.
    - 2.2.2.2 – For each countermeasure: (repeat as necessary)
      - 2.2.2.2.1 – Analyze countermeasure for residual vulnerabilities.
      - 2.2.2.2.2 – Add corrective measures to overcome vulnerabilities.
- 2.3 – Derive the complete security-oriented countermeasures design model.
  - 2.3.1 – Group all prescribed countermeasures
  - 2.3.2 – Separate countermeasures into action countermeasures and underlying countermeasures.
  - 2.3.3 – Divide countermeasures design model into an action-box and underlying planes.
    - 2.3.3.1 – Put action countermeasures inside box [group into flowcharts]
    - 2.3.3.2 – Add Underlying Countermeasures
- 2.4 – Verify attack coverage via traceability matrix.

Phase 2: Instantiate and Integrate the derived models into an e-commerce system design

3 – Instantiate the Countermeasures Design Models for Each Security Feature

4 – Integrate the Instantiated Models into an Existing Design

Figure 3.1. Our Methodology to Derive EC Security Countermeasures Design Models.

3.2. Phase 1: Select Features and Derive Design Models

In this phase, we apply steps 1 through 2.4 of our methodology. In this thesis, we selected four security features of the NIST security services model; namely authentication, authorization, access control enforcement, and transaction privacy. For each selected security feature, we will derive a security-oriented design model. The derived model will be validated via a validation traceability matrix. These steps are summarized in Figure 3.1 and discussed in detail next.

Phase 1 of our methodology is applied in our case study in Chapter 4. Sections 4.2, 4.3, 4.4, and 4.5 apply derive and validate security-oriented countermeasures design models for authentication, authorization, access control enforcement, and transaction privacy respectively.

3.2.1. Methodology Steps

Figure 3.1 depicts the methodology we will use to derive the specialized EC security countermeasures design models for security features of the NIST security services model (shown in Figure 2.1).

Step 1 - Select security features.
In this thesis, we will select four security features from the NIST security services model; namely *authentication*, *authorization*, *access control enforcement*, and *transaction privacy*.

Step 2 - For each *security feature,* derive a *countermeasures design model*:

> **Step 2.1 – Identify and abstract all attacks related to the security feature.**
> This includes all attacks referenced in OSSTMM as well as other attacks that are applicable to the selected security feature from other referenced literature. In our

thesis, we performed an exhaustive investigation into the set of all known attacks related to authentication, authorization, access control enforcement, and transaction privacy. This list of attacks will be used in this step of the methodology. The goal of having an abstract description of the attacks is to mask platform-specific execution details, such as operating system releases, and concentrate more on the functionality and requirements for understanding how the attack mechanism works. The result is a finite set of abstract attacks (a_1, a_2, .., a_m) applicable to the security feature selected in step 2.

Step 2.2 – For each security attack:

Step 2.2.1 – **Derive all *attack enablers.***

An *attack enabler is* the key component, feature, attribute, or process that an attack relies or depends on in order to be successful. In other words, it is the set of malicious user requirements (MUR) allowing for the success of the attack. The result of this step will be a set of attack enablers that enable the success of security attacks related to the security feature.

Step 2.2.2 – For each attack enabler

Step 2.2.2.1 – **Prescribe appropriate *security countermeasures.***

A *security countermeasure* is a process, technique, or feature that disables an attack enabler. The following points are to be taken into consideration when prescribing countermeasures:

- One or more countermeasures might be applicable for a specific attack enabler.
- Countermeasures might vary from one EC system to another and from one case to another.
- Countermeasures might have a profound negative impact on other non-functional requirements such as performance.

The process of prescribing the proper security countermeasure must take into consideration these factors before deciding on a specific countermeasure. Literature dealing with security risk management ([Anderson, 2001], [NIST, 2001], and [CC, 1999]) is helpful during this step and can help with the countermeasure selection decision. For example, the ultimate goal of prescribing a countermeasure (c_i) for an attack enabler (e_j) can be summarized in a temporal logic formula as:

$$\mathbf{c_i \; ? \quad (\neg e_j)^2}$$

(the presence of c_i implies that attack enabler e_j is always disabled)

Step 2.2.2.2 – For each countermeasure:

Step 2.2.2.2.1 – Analyze countermeasure for *residual vulnerabilities*.

A countermeasure might not block an attack completely and might even enable other attacks while trying to block one. This is known as a *residual vulnerability* for the countermeasure.

Step 2.2.2.2.2 – Add corrective measures to overcome vulnerabilities.

The result will be a set of countermeasures (c_1, c_2, …, c_k) that disable security attacks related to the security feature and any residual vulnerabilities introduced from the prescribed countermeasures.

Step 2.2.2.2 must be repeated as necessary for new countermeasures prescribed in step 2.2.2.2.2 until no new countermeasures are needed.

Step 2.3 – Derive the complete security-oriented countermeasures design model.

The process of deriving the security-oriented design model proceeds as follows.

Step 2.3.1 – Group all prescribed countermeasures

We start by grouping all prescribed countermeasures from steps 3.2 and 3.3. The result is a set of countermeasures that must be applied to the e-commerce system for the selected feature in step 2.

Step 2.3.2 – Separate countermeasures into action countermeasures and underlying countermeasures.

We separate countermeasures into *action countermeasures* and *underlying countermeasures*. An *action countermeasure* is an action that must be done to disable an attack (e.g. lock an account). An *underlying countermeasure* is a countermeasure that does not specify an action but is required to disable an attack (e.g. strong cryptography).

Step 2.3.3 – Divide countermeasures design model into an action-box and underlying planes.

Our countermeasures design model is divided into two sections: an action-flow box and a set of planes below the box containing underlying countermeasures. This architecture is similar to the NIST underlying technical services model. [NIST, 2001]

Step 2.3.3.1 – Put action countermeasures inside box [group into flowcharts]

Countermeasures that represent actions are placed inside the box. We then proceed to group action countermeasures into *process flow charts*. A *process flowchart* specifies the sequential order for applying countermeasures.

Step 2.3.3.2 – Add Underlying Countermeasures

Underlying countermeasures are placed in the planes below the box.

The result model, up to this point, includes the most proper security countermeasures for all known security attacks related to the security feature selected in step 2.

Step 2.4 – Verify attack coverage via traceability matrix.

Attack coverage is defined as the number of security attacks related to the security feature that are blocked by the prescribed countermeasures as well as the number of corrected residual vulnerabilities introduced by the prescribed countermeasures. Verifying attack coverage via a traceability matrix means making sure that all proper countermeasures against known security attacks are introduced at system design time.

After deriving the security-oriented design model and verifying its attack coverage, we proceed to Phase 2 of the methodology where the model is instantiated and integrated into an e-commerce system. The applicability of Phase 1 of this methodology is, of course, limited to all known security attacks. However, it is expected that new attacks will be handled in the same way.

3.2.2. Methodology Organization and Relationships

In this section we will discuss two topics: methodology organization and relationships between methodology components.

3.2.2.1. Methodology Organization

Figure 3.2 is an organizational chart that illustrates how the methodology artifacts derived and generated are related. If we take a snapshot of the system security design process while applying the methodology, the chart will look similar to the one presented above.

As seen in the figure, the methodology starts with a single security feature (SF_i) from the NIST security services model. The selected feature will have a set of possible abstract security attacks ($ASA_{i1}…ASA_{ij}$) as its subordinates. Once the abstract description is done, the process of defining the attack enabler(s) can be started. Every identified attack enabler AE_{ijk} will become a subordinate for its parent abstract security attack ASA_{ij}. Up to this point, we have identified all attack enablers for all known security attacks on the selected NIST security feature. Our next step will be to derive the security countermeasures to block the derived attack enablers. Each countermeasure (CM_{ijkz}) will be a subordinate for its respective attack enabler (AE_{ijk}).

As mentioned earlier, the process of prescribing a countermeasure varies from one system to another and from one situation to another. While it is possible for a selected countermeasure to block more than one attack enabler, we find it better to keep the

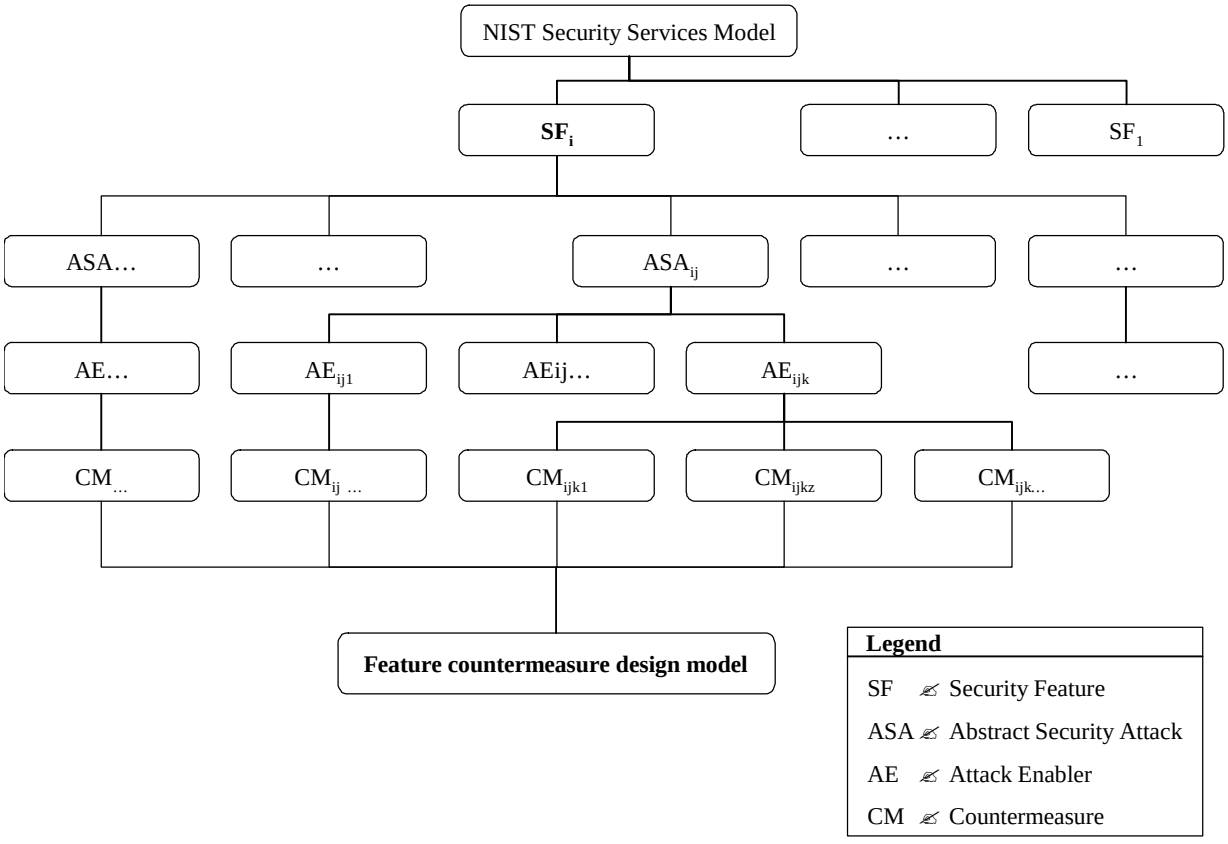

Figure 3.2. An organizational chart of the methodology.

organizational chart as is. This forces the methodology to concentrate more on blocking the attack enabler that the countermeasure is being prescribed for. Gathering and ordering the prescribed countermeasures is kept till the end when the countermeasures design model will be built. This might lead to a relatively slow model derivation process. Yet, the derived model will have all necessary countermeasures with the ability to optimize the overall security model during the final step, i.e. while gathering and ordering the prescribed countermeasures to build the countermeasures design model.

Once all countermeasures are prescribed, we will have a finite set of countermeasures (CM_{ijk1}…CM_{ijkz}) that need to be introduced into the system design. The model assembling these countermeasures in an ordered manner is what we call a countermeasures design model for the selected feature (SFi). The process of ordering and sorting the prescribed countermeasures in the model is a sensitive one and requires care in defining the order that countermeasures are to be introduced. This is due to the fact that some countermeasures are supportive to other countermeasures and must be introduced prior to the countermeasures depending on them. In the case of authentication, for example, checking to see whether the credential information is valid (one countermeasure) must not occur if the user account was locked or disabled (another countermeasure).

3.2.2.2. Relationships Between Methodology Components

Figure 3.3 shows possible relationships among security features, security attacks, attack enablers, countermeasures, and residual vulnerabilities in UML class diagram notation [UML].

- One or more security attacks (whether dependent or independent on other attacks) might be applicable to one security feature. For example, dictionary attacks and brute-force attacks are both applicable to authentication. Thus, the relationship is one feature to one-or-more attacks.
- One security attack can rely on one attack enabler to succeed. Yet, an attack enabler might be a set of properties. For example, sniffing relies on two properties as an

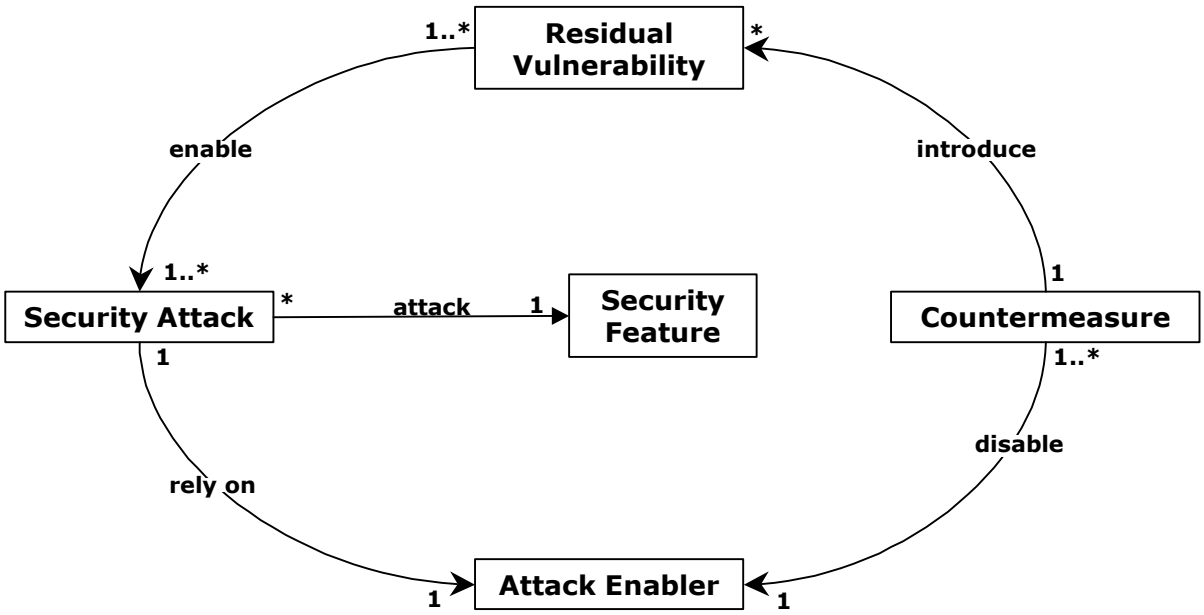

Figure 3.3. A UML class diagram of the relationships among security attacks, attack enablers, countermeasures, and residual vulnerabilities, based on Figure 3.1.

attack enabler: access to communication channel and data in clear text format.

- One-or-more *countermeasures* might be required to *disable* an *attack enabler*. For example, sniffing requires two countermeasures: an encrypted communication channel and a strong cryptographic algorithm. Roughly speaking, every attack enabler property requires one countermeasure.
- Residual vulnerabilities may remain after applying a countermeasure. [CC, 1999] Threat agents might exploit such vulnerabilities by exposing them to subsequent security attacks. One *countermeasure* might or might not *introduce* residual vulnerabilities. Thus, the relationship is "one to zero-or-more".
- On one hand, a residual vulnerability might enable *one or more* security attack. On the other hand, *more-than-one* residual vulnerabilities might be required to enable *one* security attack. Thus, one-or-more *residual vulnerabilities* might enable *one-or-more* security attacks.

3.3. Phase 2: Instantiate and Integrate the Derived Models into an E-Commerce System

In this phase, we apply the two final steps (3 and 4) of the methodology by instantiating and integrating the derived security-oriented design models into an e-commerce system design. Step 3 instantiates the countermeasures design models for each security feature from Phase 1. Step 4 integrates the instantiated models into an existing e-commerce system design. Steps 3 and 4 are described in sections 3.3.1 and 3.3.2 respectively.

3.3.1. Instantiate the Countermeasures Design Models for Each Security Feature

This phase starts by instantiating the derived security-oriented design models for the specific target e-commerce system requirements and constraints. An example of this step is instantiating *challenge information* (see section 4.2) for a SET-integrated e-commerce system.

Challenge information can be a user name and password, a digital certificate, or a smart card. On the other hand, SET (the Secure Electronic Transaction protocol) [SET, 1997-1] requires the use of digital certificates on the merchant side while keeping the use of digital certificates on the client side optional. In this example, "challenge information" will be instantiated to "digital certificates" on the merchant side because the use of SET is a system constraint and SET requires digital certificates. On the other hand, instantiating the client "challenge information" can be any of the three possible choices; namely user name and password, digital certificate, or smart card. In this case, we instantiate "challenge information" to become "user name and password" on the client side because this requires no special setup procedures as opposed to digital certificates and smart cards.

3.3.2. Integrate the Instantiated Models into an Existing Design

After constructing an instantiated model, the model is integrated into the EC system high-level design. Following the same example described above, the high-level design of the SET-integrated e-commerce system will require a digital certificate to be obtained and installed on the merchant side. Another requirement is to create a user name and password for each client user. The high-level design will also provide all security details about how to protect the merchant certificate and validate the client user name and password.

Similar reasoning guides the application of Phase 2 for all security features and their models. The details of the steps of this phase are given and applied to a SET-integrated EC system in our case study in Chapter 5. In particular, section 5.1 provides an overview of the Secure Electronic Transaction protocol (SET). Section 5.2 describes the architecture of our case study SET-integrated EC system. Sections 5.3, 5.4, 5.5, and 5.6 apply Phase 2 of the methodology to instantiate and integrate the security-oriented models for authentication, authorization, access control enforcement, and transaction privacy derived in sections 4.2, 4.3, 4.4, and 4.5 respectively.

3.4. Summary of Chapter 3

In this chapter we presented our systematic and modular methodology for deriving effective countermeasures design models for e-commerce systems along with a detailed explanation of how the methodology is organized. The relationships between different components of the methodology were also presented.

In the next chapter, we will apply Phase 1 of our methodology to four security features of the NIST security services model; namely authentication, authorization, access control enforcement, and transaction privacy. For each security feature model, we will discuss its validity and effectiveness.

Phase 2 of the methodology will also be applied to the derived countermeasures design models from Phase 1, through a case study, to a SET-integrated e-commerce system design in Chapter 5. Finally, we will present an evaluation of our methodology based on the two case studies in Chapter 6.

Chapter IV

Case Study Phase 1: Applying the Design for Security Methodology to the NIST Security Services Model

4.1. Case study overview

In this chapter we will apply our design for security methodology to derive security design models for four preventive security features in the NIST security services model. [NIST, 2001] Section 4.2 applies our methodology to *authentication*, section 4.3 applies the methodology to the *authorization* security feature, section 4.4 applies the methodology to *access control enforcement*, and section 4.5 applies our methodology to the *transaction privacy* security feature. Finally, we conclude the chapter with an evaluation of the methodology based on this case study.

4.1.1. Overview

Every section will begin with a definition of the security feature to which we intend to apply our methodology. As specified in chapter 3, our methodology proceeds as follows:

1. We select security features. For our case study we selected *authentication*, *authorization*, *access control enforcement*, and *transaction privacy*.

2. For each feature,

2.1. We identify and abstract all related security attacks

2.2. For each attack,

3.1. We derive all attack enablers

3.2. We prescribe appropriate security countermeasures

3.3. For each countermeasure,

3.3.1. We analyze the countermeasure for residual vulnerabilities

3.3.2. We add corrective measures to overcome vulnerabilities

2.3. We derive the complete security-oriented countermeasures design model for the feature

2.4. We verify attack coverage via a traceability matrix

Our objective in this chapter is to employ a case study of four features in order to demonstrate that our design-for-security models:

- Meet or exceed security requirements provided in current security standards at system design time for these four features.
- Are able to disable malicious user requirements at system design time for these four features.
- Provide effective security against all known security attacks related to the e-commerce domain with respect to these four features.
- Can be readily integrated into high-level design documents of e-commerce systems.

The reader may also be interested to know that this chapter provides a useful overview of all known security attacks related to e-commerce *authentication*, *authorization*, *access control enforcement*, and *transaction privacy*, respectively.

4.1.2. Case Study Scope

E-commerce systems, in general, include multiple parties: client, web server, business server, database management system, payment gateway, etc. A client usually uses a web

browser to connect to the web server. The web server then contacts the commerce application which might contact any of the other parties such as the database management system for information usage or the payment gateway for payment information.

In our case study, the term e-commerce system denotes the set of applications on the business server side that work together to provide the overall e-commerce solution. This includes web servers, database managements systems, and commerce applications that provide the business logic of an e-commerce system. Client-side applications such as web browsers are not considered here since the responsibility for securing these applications belong to the client.

Our main focus will be on securing the set of commerce applications at the business server side. The reason for this is that e-commerce system providers usually make use of third-party software systems for web servers (such as Apache, IBM HTTP Server, etc.), databases (IBM DB2, Oracle, etc.), and a payment gateway (IBM payment gateway, etc) Designing these third-party components is done by the parties that develop them and, therefore, cannot be controlled by e-commerce system providers.

All known security attacks that affect an e-commerce system will be taken into consideration. However, countermeasures to defeat these attacks at system design time will be limited to the scope of the commerce application on a business server for the above reasons.

4.2. Security-Oriented Authentication Design Model

Step 1 is to select a NIST model security feature. Here we select *authentication*.

Authentication is the process of verifying the identity of a user, process, or device, often as a prerequisite to allowing access to resources in a system. [NIST, 2001] The identity of a certain user or process is challenged by the system and proper steps must be taken to prove the claimed identity.

Authentication models may depend on specific technologies. An example of such a model is the open *authentication* model supporting electronic commerce in distributed computing [Chang, 2002] that is based on CORBA technology [OMG, 2000] and provides an extension to the Kerberos *authentication* framework [Kohl, 1993] using a public key cryptosystem. Another example of an *authentication* model is a general framework for constructing and analyzing *authentication* protocols in realistic models of communication networks. [Bellare, 1998]

The above referenced literature emphasizes satisfying standard security requirements while providing extensions to different technologies such as CORBA and Kerberos. As discussed earlier, the main emphasis in current standards for security requirements is on satisfying legitimate user requirements from a security point of view; however, in this thesis, we emphasize blocking malicious user requirements at system design time (this is also known as preventive security design [Jonsson, 1998]) The goal of this section is to show how our methodology is useful for deriving a preventive design model for *authentication* that can either be incorporated into any integrated authentication model (such as above) or can be implemented as a standalone authentication module in e-commerce systems.

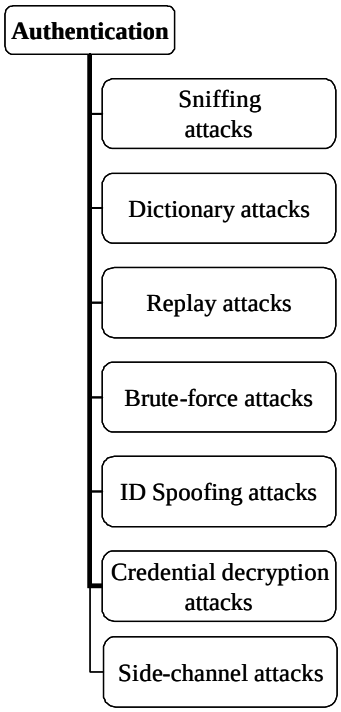

Figure 4.1. Authentication-related security attacks.

4.2.1. Step 2.1: Identify all attacks related to authentication

Security attacks related to authentication can be identified from the literature and from personal experience. In this thesis, we have projected all known security attacks onto various types of e-commerce authentication models described in [Wilson, 1997], [Ford, 1997], [Agnew, 2000], [Bellare, 1998], [Hawkins, 2000] and [Chang, 2002]. This comprises most known attacks applicable to authentication in the domain of EC systems. These seven attacks are summarized in Figure 4.1 listed in decreasing order of commonality and are described next.

Note that the main focus of this thesis will be on attacks directly related to e-commerce systems. Attacks related to network components, to third-party software components, and attacks against the operating system that is supporting the e-commerce application will not be discussed here. In practice, of course, these types of attacks also have to be

handled.

The specific security attacks related to *authentication* in e-commerce systems are ([Herzog, 2001], [Viega, 2002], [Nguyen, 2001], [Anderson, 2001], [SecTeam] and [Schneier, 2000]):

- Sniffing attacks (also known as man-in-the-middle attacks)
- Dictionary attacks
- Replay attacks
- Brute-force attacks
- ID spoofing attacks (also known as spoofing attacks)
- Credential decryption attacks (supplementary to other types of attacks)
- Side-channel attacks

We will now consider each type of attack, derive attack enablers, and prescribe effective countermeasures.

4.2.2. Step 2.2: For each authentication attack, derive its enablers and countermeasures

This section provides a succinct abstract description of all known *authentication*-related security attacks. Attack enablers are then identified, and effective countermeasures are prescribed. The attacks are presented and discussed below in order of dependence, since some of them are related (e.g. dictionary attacks depends on brute-force attacks). Figure 4.2 shows all security attacks related to authentication in e-commerce systems (a) along with the attack enablers (b) and prescribed countermeasures (c). We will refer to this figure as we proceed with our discussion about security attacks below.

4.2.2.1. Sniffing Attacks

Sniffing attacks ([Herzog, 2001], [Viega, 2002], [Nguyen, 2001], and [Schneier, 2000])

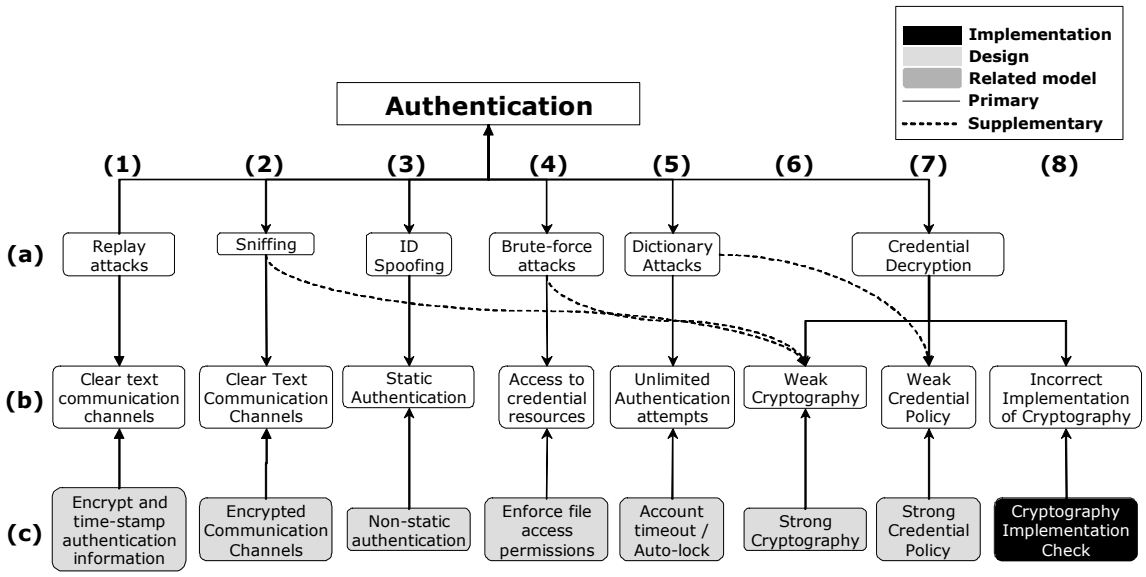

Figure 4.2. Organizational chart for authentication security attacks (a), attack enablers (b), and countermeasures (c). Note that residual vulnerabilities are not shown.

(also known as the *man-in-the-middle attacks*) are the digital analogues to phone tapping or eavesdropping. This attack captures information as it flows between a client and a server. Usually, a malicious user attempts to capture TCP/IP transmissions, because they may contain information such as usernames, passwords, or the actual contents of an e-mail message. A *sniffing attack* (a2) is often classified as a *man-in-the-middle attack* because in order to capture packets from a user, the machine capturing packets must lie in between the two systems that are communicating (a *man-in-the-middle attack* can also be waged on either one of the two systems). The *attack enabler* in this case is the process of sending data across communication channels in *clear text* format (b2). Preventing access to the communication channel is not a valid countermeasure in this case due to the open nature of the Internet. However, by encrypting the communication channel (c2) between the user/process and the system, sniffing attacks are disabled, i.e., sniffing retrieves only useless encrypted information. However, the information can be duplicated and substituted for subsequent transmissions. This type of attack is known as "replay attacks" and will be discussed later on in this section.

4.2.2.2. ID Spoofing Attacks

ID spoofing attacks ([Herzog, 2001], [Viega, 2002], [Nguyen, 2001], [Schneier, 2000], and [Anderson, 2001]) occur when a malicious user or process claims to be a different user or process (a3). This attack allows an intruder on the Internet to effectively impersonate a local system's IP address. If other local systems perform session authentication based on the IP address of a connection (e.g. rlogin with .rhosts or /etc/hosts.equiv files under Unix), they will believe incoming connections from the intruder actually originate from a local "trusted host" and will not require a password. The *attack enabler* for this attack is for authentication to rely on static information (b3) such as IP addresses, host names, etc. This is equivalent to trusting certain hosts or processes according to some pre-defined static information. The system authenticates the user or process only by checking the given static information. In such a case, the attacker will attempt, through complex attack tools, to "spoof" the system by claiming that he/she is the trusted host or process. Since no challenge is attempted in this case, the attack has a great chance of succeeding. The *countermeasure* for such an attack is to use *challenge-based authentication* (c3). Challenge-based authentication includes the use of certificates, user/password combinations, etc. If challenge-based authentication is inapplicable for a certain specific case, then *least privilege static authentication* must be applied. *Least privilege static authentication* means giving the least possible access privilege to the fewest possible number of users, processes or hosts after successful authentication. By doing so, the risk associated with relying on static authentication, when challenge-based authentication cannot be applied, is kept to a minimum.[CERT, 1997]

4.2.2.3. Brute-Force Attacks

A Brute-force attack ([Herzog, 2001], [Viega, 2002], and [Anderson, 2001]) is any form of attack against a password file that attempts to find a valid username and password by successive guessing (a4). This type of attack is *enabled* by gaining access to the credential (user names and passwords) storage medium (b4). The attacker first retrieves a copy of the database system or system file holding credential information. If the

credential information is encrypted, a brute-force attack tool will try all possible combinations of user names and passwords. For each combination, the user name and password are encrypted using the same encryption algorithm that was used to encrypt the original credential information. Then, the encrypted data is compared to the retrieved copy of credential data. Different types of encryption algorithms are used and the attack proceeds until both credentials (user name and password) match. The *countermeasure* for this type of attack is to enforce access permissions through a strong access control policy at the operating system level (c4). By doing so, malicious users will fail to retrieve a copy of credential information and, thus, the brute-force attack is disabled.

4.2.2.4. Dictionary Attacks

A *dictionary attack* ([Viega, 2002], and [Anderson, 2001]) is the "smart" version of brute-force attacks and is directed towards finding passwords in a specific list, such as an English dictionary. Dictionary attacks (a5) are also executed using automated tools. Moreover, these tools are capable of working on web interfaces without access to the encrypted format of credential information. These tools require the prior knowledge of the user name only. Once given a user name, the attack tool will try all possible combinations of that user name with a huge database (such as a dictionary) of possible passwords. This attack has a high probability of succeeding since we, as humans, tend to use passwords that are easy to remember. The *attack enabler* is a "high" number of allowed consecutive unsuccessful authentication attempts (b5). The *countermeasure*, in this case, is to prevent the automation of the attack by setting an upper limit on the allowed number of successive unsuccessful authentication attempts. This can be done through an account auto-lock or a timeout procedure (c5). In other words, when a certain number of consecutive, unsuccessful authentication attempts is reached, the system will automatically lock or disable the account and will alarm the system administrator. This will prevent the dictionary attack from proceeding and, thus, the attack is disabled. Enabling or unlocking the account can be done either by the user or automatically by the system after a certain period of time.

A *residual vulnerability* of the countermeasures in this case, account auto-locks or timeouts, occurs when malicious users target them as means for *denial of service attacks* ([Herzog, 2001], [Viega, 2002], [Nguyen, 2001], and [Treese, 1998]). The *residual vulnerability* occurs when these countermeasures cause the prevention of access by legitimate users using the EC system because their accounts have been disabled. This contradicts an essential security objective: namely availability [NIST, 2001]. The *countermeasure* for this residual vulnerability is to allow legitimate users to unlock their account through an easy-to-perform process that can be done at any time. By doing this, the dictionary attack will still be defeated by the account auto-lock feature and no residual vulnerability will be introduced by the account auto-lock countermeasure. This, therefore, will provide security against dictionary attacks and will still maintain system availability by allowing legitimate users to use the system.

4.2.2.5. Replay Attacks

A *replay attack* (a1) occurs when a malicious user captures an authentication sequence that was transmitted through the network by an authorized user, and then replays the same sequence to the server to get himself/herself authenticated. [Anderson, 2001] The *attack enabler* in this case is, again, access to the communication channel and data sent in clear text format (b1). The proper *countermeasure* is to encrypt and time-stamp all sensitive data sent across the communication channel (c1). By doing this, "replayed" messages can be recognized and discarded and this type of attack is disabled.

4.2.2.6. Credential Decryption Attacks

Credential decryption is a basic supplementary attack for sniffing attacks, brute-force attacks, and dictionary attacks (a7). A tool whose aim is to break the encryption algorithm that was used to encrypt credential information usually performs these attacks. [SecTeam] A*ttack enablers* for this attack might be a weak cryptographic algorithm (b6), a weak credential policy (b7), or an incorrect implementation of the cryptographic algorithm (b8).

Weak cryptography increases the probability of success for a brute-force attack or a sniffing attack by allowing the use of cryptographic systems that are easy to crack. Its countermeasure is to use a strong cryptographic algorithm that is hard to crack (c6). Please note that a weak cryptography is not the same as a weak credential policy.

A *weak credential policy*, on the other hand, increases the probability of a dictionary attack's success by allowing the existence of easy-to-guess passwords. Its *countermeasure* is to have a strong credential policy (c7) that forces legitimate system users to create and maintain a safe password that is easy to remember for a legitimate user and difficult to guess for a malicious user.

The *countermeasure* to an *incorrect implementation of cryptography* is to thoroughly verify the cryptographic algorithm (c8) after system implementation. This cannot be done at design time, and is therefore not discussed further in our thesis.

4.2.2.7. Side-Channel Attacks

Side-Channel Attacks: In cryptographic devices such as smart cards, data useful to an attacker other than input data and output data may 'leak out' during cryptographic procedures. The computation time of cryptographic procedures is one kind of such data, as is power consumption. Because the smart card uses an external power source, power consumption can be monitored.

[Kocher] developed a side channel attack in which an attacker infers stored secret information in a cryptographic device by using such leaked data. This type of attack, which includes a timing attack, a Simple Power Analysis (SPA) attack, and a Differential Power Analysis (DPA) attack, renders smart cards particularly vulnerable.
A *timing attack* is a side channel attack in which an attacker infers the secret information by using computation time as leaked data. Some methods of timing attack use statistical analysis to reveal the secret information, others infer it from a one-time computation.

A *Simple Power Analysis (SPA) attack* is a side channel attack in which an attacker infers the secret information by using power consumption as leaked data. An SPA attack captures the secret information by direct observation of a device's power consumption without the need for statistical analysis.

A *Differential Power Analysis (DPA) attack* is a side channel attack in which an attacker infers the secret information by using statistical analysis of power consumption. This attack is the most powerful side channel attack.

For the purpose of our research, smart cards might be used for authentication purposes only at the client side. In this case, the card and the external power source are both assumed to be secure since they are used by the client and not by the EC system itself. The main emphasis of our research is on securing the EC system. In other words, our goal is to secure data transmitted and received by the EC system. Accordingly, our thesis will not deal with these types of attacks. Yet, it is important that security architects be aware of the existence of these types of attacks.

4.2.2.8. Authentication Security Attacks Summary

Figure 4.2 shows all security attacks related to authentication in e-commerce systems (a) along with the attack enablers (b) and prescribed countermeasures (c). Moreover, this figure is complete with respect to the current state of knowledge.

For example, as shown in Figure 4.2, *access to credential resources* (b4) and *weak cryptography* (b6) are two attack enablers for brute-force attacks (a4). The first provides access to the medium in order to retrieve credential information and the second allows for a low-cost security attack.

Weak credential policy (b7), *weak cryptography* (b6), and *incorrect implementation of cryptography* (b8) are three attack enabler properties for *credential decryption* attacks

((a7). A *weak credential policy* (b7) allows system users to select easy-to-guess passwords.

Weak cryptography (b6), on the other hand, allows for a low-cost brute-force security attack (a4). A brute-force security attack tools can crack a weak cryptographic algorithm in much less time than a strong cryptographic algorithm. The shorter the time period required to crack the cryptographic algorithm, the lower the cost for the brute-force attack.

An *incorrect implementation of the cryptographic algorithm* (b8) can be seen as an implementation defect and can only be checked after the system is implemented. Thus, a c*ryptography implementation check* (c8) is a countermeasure that can not be applied at system design time. Yet, it is still introduced during the system design phase by allowing security architects to provide guidelines for subsequent system implementation and testing. In other words, the countermeasure for this type of attacks is included in the e-commerce design model. However, its effectiveness and adequacy can only be measured after system implementation.

Weak cryptography (b6) is a supplementary attack enabler for *sniffing attacks* (a2) because a weak cryptographic algorithm increases the probability of succeeding in retrieving credential information from an encrypted channel.

S*niffing* (a2) and *replay* (a1) attacks both rely on a *clear text communication channel* (b1 and b2) in order to be able to retrieve data sent across from the client to the server and vice versa.

4.2.3. Step 2.3: Derive the Authentication Security-Oriented Design Model

After identifying the attacks, attack enablers, and countermeasures for *authentication,* the

prescribed countermeasures are grouped and ordered into a countermeasures design model.

The process of deriving the *authentication* design model (shown in Figure 4.3) proceeds as follows:

1- We start by grouping all prescribed countermeasures. The result is a set of countermeasures that must be applied to the e-commerce system for this feature.

2- We separate countermeasures into *action countermeasures* and *underlying countermeasures.* An *action countermeasure* is an action that must be done to disable an attack (e.g. lock an account). An *underlying countermeasure* is a countermeasure that does not specify an action but is required to disable an attack (e.g. strong cryptography).

3- Our countermeasures design model is divided into two sections: an action-flow box

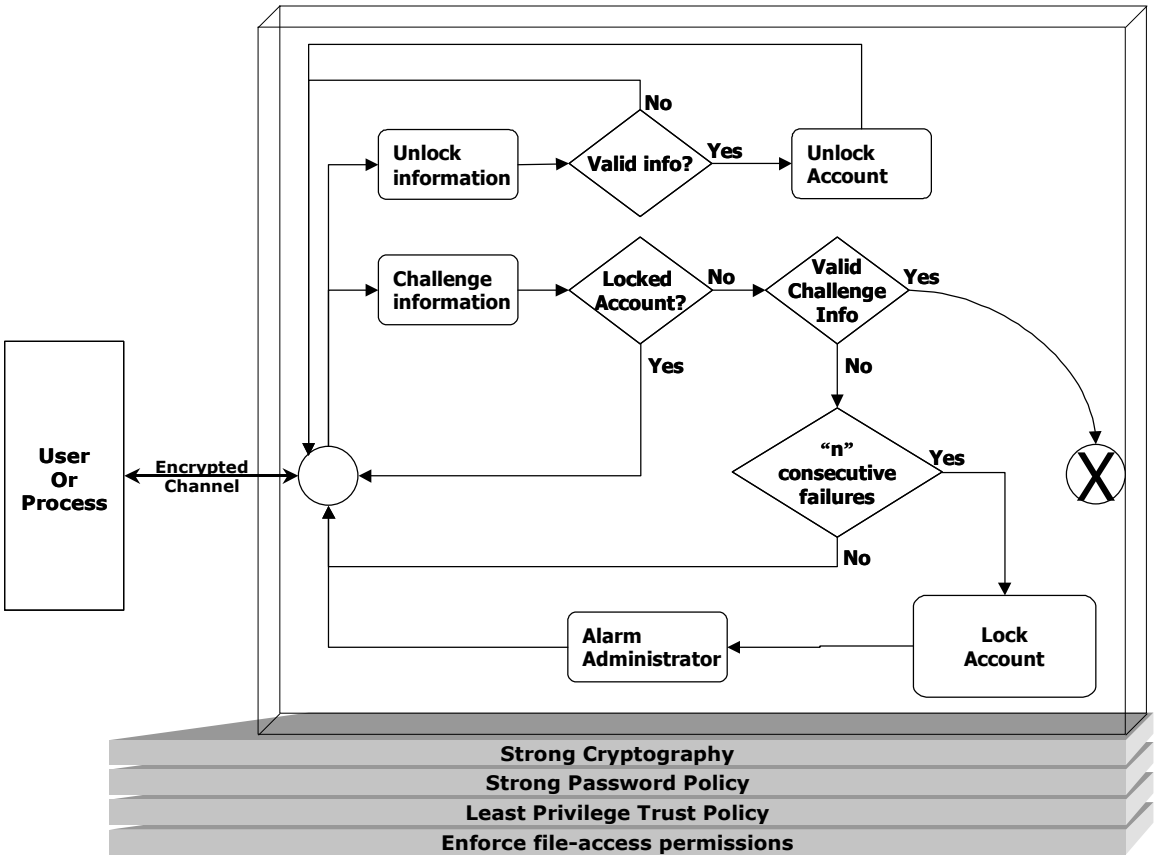

Figure 4.3. The derived e-commerce authentication countermeasures design model

and a set of planes below the box containing underlying countermeasures.

4- Countermeasures that represent actions are placed inside the box. Similarly, underlying countermeasures are placed in the planes below the box.
5- We then proceed to group action countermeasures into *process flow charts*. A *process flowchart* specifies the sequential order for applying countermeasures.

Figure 4.3 shows the detailed authentication countermeasures design model derived by applying our methodology. In the NIST security services model [NIST, 2001], authentication was only a single feature, whereas here it is broken down into a detailed authentication process.

The EC authentication countermeasures design model shown above is detailed enough to be directly applied to the design of EC systems. This model satisfies security requirements and blocks malicious user requirements at system design time. It provides effective security against all known security attacks related to authentication in the e-commerce domain.

4.2.4. Step 2.4: Validate the Security-Oriented Authentication Countermeasures Design Model

We now validate the model by checking the adequacy of its coverage with respect to attacks and residual vulnerabilities. An effective tool for this is the *validation traceability matrix* in Figure 4.4. This matrix shows the security coverage of the *authentication* design model in relation to all known authentication security attacks. A "D" implies a countermeasure was prescribed against an attack during system design time. An "I" implies a countermeasure must be applied after the system is implemented. In both cases, the countermeasure was introduced at system design time to allow proper design for security, and planning for implementation.

COUNTERMEASURES	AUTHENTICATION SECURITY ATTACK TYPE						
		Sniffing attacks	ID Spoofing attacks	Brute-force attacks	Dictionary attacks	Replay attacks	Credential Decryption
	Encrypted comm. channels	D				D	
	Non-static authentication		D				
	Enforce file-access permissions			D			
	Account timeout / Auto-lock				D		
	Strong crypt.						D
	Strong credential policy						D
	Cryptography impl. check						I

D = design time, I = after system implementation

Figure 4.4. Validation traceability matrix for authentication countermeasures versus authentication security attacks.

Countermeasures for sniffing, ID spoofing, brute-force attacks, dictionary attacks, weak cryptography, and weak credential policy were all incorporated at system design time. The only countermeasure to be still applied after system implementation is "cryptography implementation check."

This validation traceability matrix is a very simple, yet very effective tool for assuring that every attack is addressed by at least one countermeasure (no empty columns), and that every countermeasure is needed to guard at least one attack (no empty rows).

Thus, the security of our *authentication* design model is assured at design time. In other words, a faithful implementation of the countermeasures design model for authentication is guaranteed to block all known security attacks by disabling all malicious user requirements that enable security attacks on an EC system authentication process. In this way, we do not have to wait for security weaknesses to be identified only at system test time, or worse (after release to customers). This avoids costly rework.

In the next section, we will apply our methodology to derive an EC design model for the authorization feature in the NIST security services model.

4.3. Security-Oriented Authorization Design Model

Step 1 is to select a NIST model security feature. Here we select *authorization*.

Authorization is the process of giving someone the permission to do or have something. In multi-user computer systems such as EC systems, a system administrator defines which users are allowed access to the system and what privileges of use (such as access to which components, hours of access, and so forth). Assuming that someone has logged in to an EC system, the system may want to identify what resources the user can be given during this session. Thus, *authorization* is sometimes seen as both the preliminary setting up of permissions by a system administrator and the actual checking of the permission values that have been set up when a user requests access.

Authorization models might rely on specific frameworks or models for implementation guidance. An example of such guidance is the *AAA Authorization Framework* which is not intended to be a standard but serves as an asset for modeling authorization into EC systems. ([Vollbrecht, 2000-1], [Vollbrecht, 2000-2] and [Farrell, 2000]) The purpose of this framework is to provide the base requirements for a*uthorization*. It presents an architectural framework for understanding the *authorization* of Internet resources and services and deriving requirements for*authorization* protocols.

The above referenced literature emphasizes satisfying standard security requirements. As discussed earlier, the main emphasis in current standards for security requirements is to satisfy legitimate user requirements from a security point of view. Testing against the existence of malicious user requirements is done after the system is implemented. In this thesis, we emphasize blocking malicious user requirements at system design time (this is also known as preventive security design [Jonsson, 1998]). The goal of this section is to have a preventive design model for *authorization* that can be incorporated into any integrated *authorization* model (such as above) or can be implemented as a standalone authorization module in e-commerce systems.

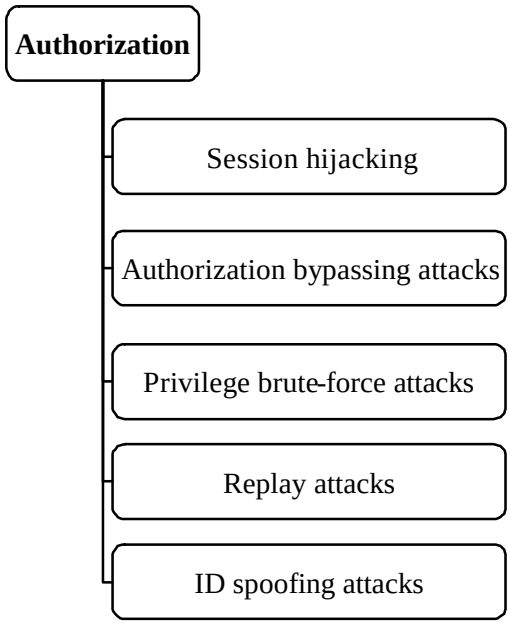

Figure 4.5. Authorization-related security attacks.

4.3.1. Step 2.1: Identify all attacks related to authorization

Security attacks related to authorization can be identified from the literature and from personal experience. In this thesis, we have projected the identified security attacks onto various types of e-commerce *authorization* models described in [Alturi, 2002], and [Wedde, 2001]. This comprises most known attacks applicable to authentication in the domain of EC systems. These five attacks are summarized in Figure 4.5 listed in decreasing order of commonality and are described next.

It is important to remind that the main focus of this thesis will be on attacks directly related to e-commerce systems. Attacks related to network components, to third-party software components, and attacks against the operating system that is supporting the application will not be discussed here. In practice, of course, these types of attacks also have to be handled.

The specific security attacks related to *authorization* in e-commerce systems are ([Herzog, 2001], [Viega, 2002], [Nguyen, 2001], [Schneier, 2000], and [Anderson,

2001]):

- Session hijacking attacks
- Authorization bypassing attacks
- Privilege brute-force attacks
- Replay attacks
- ID Spoofing attacks

We will now consider each type of attack, derive attack enablers, and prescribe effective countermeasures.

4.3.2. Step 2.2: For each authorization attack, derive its enablers and countermeasures

This section provides a succinct abstract description of *authorization*-related security attacks. Attack enablers are then identified and effective countermeasures are prescribed. The attacks are presented and discussed below in order of dependence; since some of them are related (e.g. Session hijacking attacks depend on ID spoofing attacks). Figure 4.6 shows all security attacks related to authorization in e-commerce systems (a) along with the attack enablers (b) and prescribed countermeasures (c). We will refer to this figure as we proceed with our discussion about security attacks below.

4.3.2.1. ID Spoofing Attacks

ID spoofing attacks ([Herzog, 2001], [Viega, 2002], [Nguyen, 2001], [Schneier, 2000], and [Anderson, 2001]) occur when a malicious user or process claims to be a different user or process (a1). A detailed description of this type of attacks was provided in section 4.2.2 of this thesis.

The *attack enabler* for this type of attacks on EC systems is the stateless nature of the Internet part of EC systems (b1). Stateless means that there is no record of previous

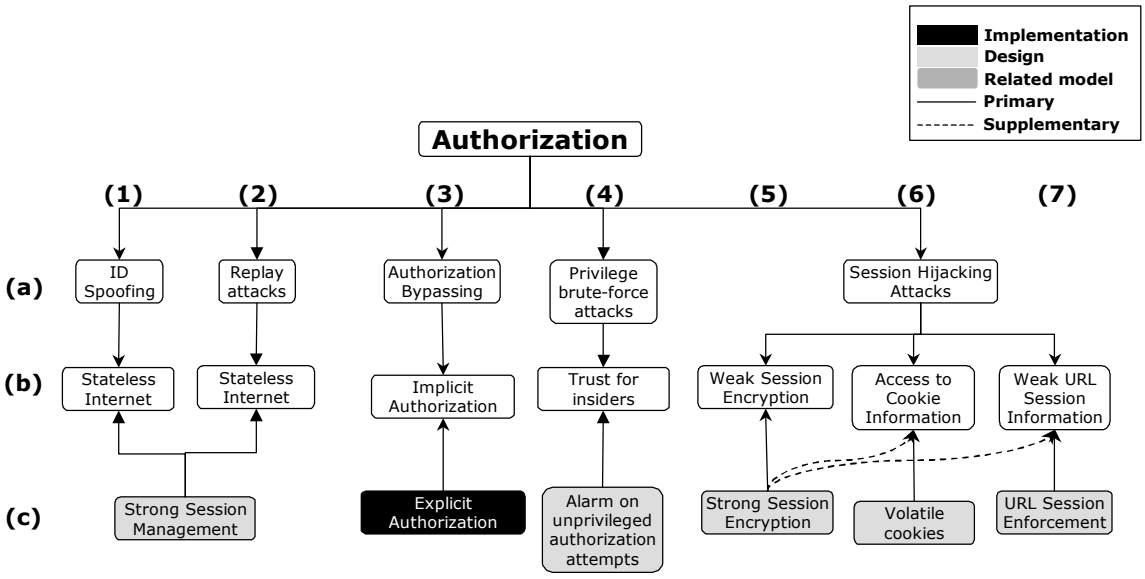

Figure 4.6. Organizational chart for authorization security attacks, attack enablers, and countermeasures.

system interactions and that each interaction request has to be handled based entirely on information provided with it.

The proper *countermeasure* is to provide a stateful EC system. Stateful means the system keeps track of the state of interaction, usually by setting values in a storage field designated for that purpose. In other words, stateful means the ability to identify the user across multiple EC system requests. In this case, state information can be kept through the usage of session management techniques (c1). ([Alturi, 2002], [Vollbrecht, 2000-1], and [Vollbrecht, 2000-2])

Session Management is a technique that enables web applications, and EC systems in our case, to transform the Internet from a stateless medium to a stateful one. From a security point of view, it is also responsible for protecting the EC system from malicious behavior of application clients whether intentional or non-intentional.

Most popular session management implementations make use of cookies and/or URLs in web browsers to save session information. A *residual vulnerability* for introducing

session management is enabling session hijacking attacks (a6). This type of attacks will be detailed later on in this section.

4.3.2.2. Authorization Bypassing Attacks

Authorization bypassing attacks (a3) might succeed with a weak authorization implementation in place. This attack might succeed especially with systems implementing distributed authorization architectures.

The *attack enabler* in this case is that security designers might implicitly rely on the fact that EC system users will not be given access to a certain system component unless properly authorized. This is also known as implicit authorization (b3). As a result, actual authorization might not take place in the system implementation when trying to access the system component and, thus, the authorization process can be bypassed.

The proper *countermeasure* for this attack is to enforce explicit authorization (c3) throughout the whole EC system. By doing so, every system component will be forced to explicitly authorize users before giving access. Thus, authorization bypassing attacks are disabled.

4.3.2.3. Privilege Brute-Force Attacks

Privilege brute-force attacks (a4) are similar to brute-force attacks and were discussed earlier in the case of authentication. What is different in this case is that the malicious user is already authenticated and provides a certain level of trust to the EC system, also known as an insider. The goal of this attack is to brute-force the EC system for higher access privileges.

An example of such an attack is when a malicious user registers as an ordinary user and uses session management information and system commands to access unauthorized components or parts of the EC system. In such a case, malicious users might be able,

through complex tools, to perform reverse-engineering analysis of the EC system. This might lead to more knowledge of the system architecture and, thus, results in an increase in the probability of the attack success.

The *attack enabler* in this case is when the system trusts registered users (b4) without taking into consideration the possibility of having a malicious user, also known as an insider.

The proper *countermeasure* for this attack is to alarm the EC system administrator when unauthorized access attempts seeking higher privileges take place (c4). Repeated attempts from the same system user will help the system administrator capture and disable the attack either by warning the user or by disabling his EC system account.

4.3.2.4. Replay Attacks

Replay attacks (a2) in the case of authorization are similar to a certain extent to the case of authentication. A detailed description of the attack was provided in section 4.2.2.

As discussed earlier, the *attack enabler* for this type of attacks on EC systems is the stateless nature of the Internet part of EC systems (b2).

The proper *countermeasure* in the case of authorization is to provide a stateful EC system through the usage of session management techniques (c2). ([Alturi, 2002], [Vollbrecht, 2000-1], and [Vollbrecht, 2000-2]) This will prevent malicious users from claiming false identities when attempting to seek authorization.

Session management implementations make use of cookies and/or URLs in web browsers for saving session information. A *residual vulnerability* of introducing session management techniques is discussed next.

4.3.2.5. Session Hijacking Attacks

As discussed earlier, a *residual vulnerability* for introducing session management as a countermeasure is that malicious users will be able to execute *session hijacking attacks* (a6). ([Herzog, 2001] and [Anderson, 2001]) This type of attack involves an attacker using captured, brute forced, or reverse-engineered authorization information (such as session information) to seize control of a legitimate user's session while that user is logged into the EC system. This usually results in the legitimate user losing access or functionality to the current EC system session, while the attacker is able to perform all normal application functions with the same privileges of the legitimate user.

This type of attacks usually relies on a combination of other simpler session management attacks (such as brute-force attacks and replay attacks). The act of taking control of the session after successfully obtaining or generating an authentication token is called session hijacking. The user may or may not still have all or partial control of his EC system session, and may be forced out in the process. An attacker might be able to take control of an active session simply by pasting a URL into his web browser or by loading stolen cookie data and accessing a particular web site or URL (similar to a replay attack).

Session management information is usually encrypted and sent to the client side where web browsers save a copy for later usage. There are three options for saving session information on the client side: using cookies, URLs, or hidden HTML forms. Whether saved in the URL or in a hidden HTML form, session information is clearly seen by anyone. Thus, for the purpose of our discussion, using URLs and hidden HTML forms are similar.

The session hijacking *attack enabler* has three properties: weak session encryption (b5), access to cookie information (b6), and weak URL session information (b7).

The first *attack enabler* property is a weak encryption algorithm that allows malicious users to capture session information, decrypt it, and perform session hijacking attacks.

The *countermeasure* for this attack enabler property is to have a strong session encryption algorithm (c5). This will prevent malicious users from retrieving useful session information for the purpose of session hijacking attacks in a timely manner.

The second *attack enabler* property, access to cookie information, occurs when a session lifetime is longer than the web browser session. Web browsers, in this case, are forced to save cookies on local user hard drives. This allows malicious users, through complex tools, to retrieve saved session information and perform session hijacking attacks. The *countermeasure* for this attack enabler property is to use volatile cookies (b6). Volatile cookies are not saved on local user hard drives. They are saved in the system memory, and once the web browser session ends, i.e. the web browser is closed, the cookie is erased from memory and can not be retrieved.

The third *attack enabler* property is using weak URL session information. This usually occurs when cookies are not available and is, practically, similar to saving session information in hidden HTML forms. In this case, session information cannot be volatile since it can be seen by the human eye either in the URL or in the HTML file. Malicious users can, through special tools, retrieve a copy of the session information and start a session hijacking attack. The proper *countermeasure* is to have URL enforcement (c7). This includes but is not limited to re-authenticating the user before critical actions are performed (such as finalizing purchase orders, requesting money transfers, etc.), and mapping session information to web browser instances. By doing so, if a malicious user successfully steals session information and starts a session hijacking attack, our system will either block him by capturing the fact that a different browser instance is used or will prevent him from causing expensive damage by requesting reauthentication before critical actions.

4.3.2.6. Authorization Security Attacks Summary

Figure 4.6 shows the security attacks related to authorization in e-commerce systems (a) along with the attack enablers (b) and prescribed countermeasures (c). Moreover, this

figure is complete with respect to the current state of knowledge.

For example, as shown in Figure 4.6, *strong session management* (c1 and c2) is a countermeasure against ID spoofing attacks (a1) and replay attacks (a2). Its *residual vulnerability* is that it could become subject to session hijacking attacks (a6).
Strong session encryption (c5) is a direct countermeasure to *weak session encryption* (b5). This forces the *authorization* model to include a strong cryptographic system. Moreover, *strong session encryption* (c5) is a supplementary countermeasure to *cookie session enforcement* (c6) and *URL session enforcement* (c7) for blocking *session hijacking attacks* (a6).

Explicit authorization (c3) requires that the EC system implementation be checked. Thus, it cannot be applied at system design time and has to be postponed for the system-testing phase. Yet, it is still introduced during the system design phase by allowing security architects to provide guidelines for subsequent system implementation and testing. The presence of such guidelines does not only help avoid such security attacks, but can also be used to minimize security defects by describing best practices related to implementation and coding.

4.3.3. Step 2.3: Derive the Authorization Security-Oriented Design Model

After identifying the attacks, attack enablers, and countermeasures for *authorization*, the prescribed countermeasures are grouped and ordered in a design model. The process of deriving the design model was described earlier in section 4.2.3 of this thesis.

Figure 4.7 shows the detailed *authorization* design model derived by applying our methodology. As seen in the figure, the main entry point to the *authorization* model is explicit authorization. In our earlier discussion, we explained this as the process of enforcing *authorization* on all EC system components. Once *authorization* is enforced,

we then proceed to actually validate the presented *authorization* information. If successful *authorization* is reached, then the EC system will attempt to enforce *authorization* through session management techniques (i.e. URL sessions and/or cookie sessions) before ending the *authorization* process. If any error occurs while enforcing session management authorization, the process goes back to its starting point and the *authorization* process fails. If, on the other hand, the presented *authorization* information was not valid, the *authorization* process will fail and the EC system administrator gets alarmed. The only path to successful *authorization* is to have explicit *authorization*, valid *authorization* information, and successful session managementenforcement.

Strong session encryption and *strong session management* are two countermeasures at the basis of our authorization model. Without these countermeasures in place, attacking the *authorization* model has a great chance of succeeding.

The EC authorization countermeasures design model shown above is detailed enough to be directly applied to the design of EC systems. This model satisfies security requirements and blocks malicious user requirements at system design time. It provides

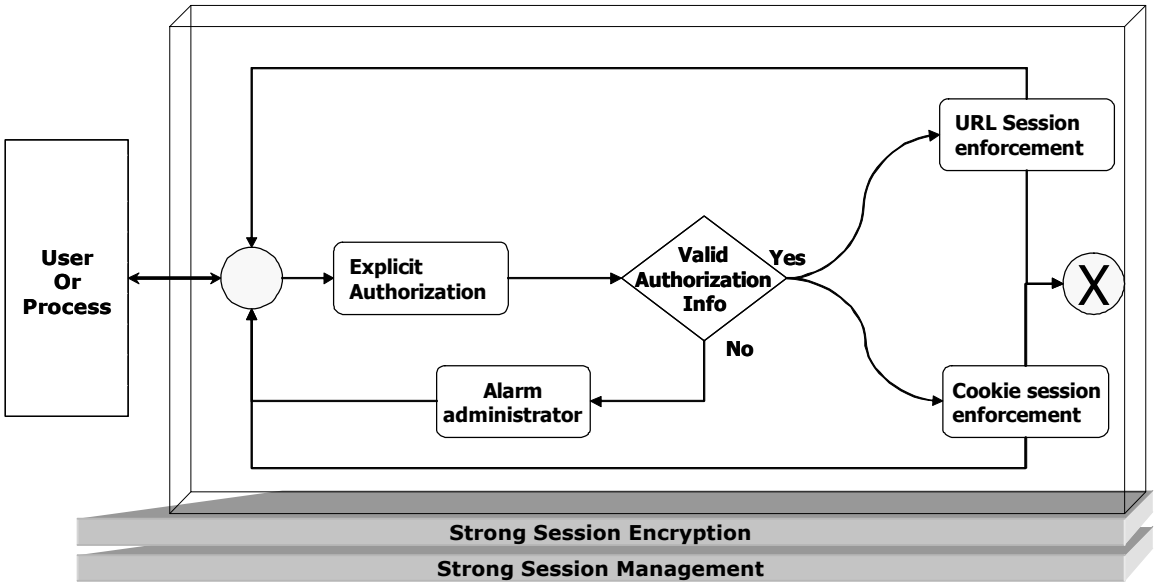

Figure 4.7. The derived e-commerce authorization countermeasures design model.

effective security against all known security attacks related to *authorization* in the e-commerce domain

4.3.4. Authorization Validation Traceability Matrix

We now validate the model by checking the adequacy of its coverage with respect to attacks and residual vulnerabilities, by using the *validation traceability matrix* in Figure 4.8.

All countermeasures against EC *authorization* attacks were introduced at design time except the explicit authorization countermeasure that requires having an implementation of the EC system. An example of how to instantiate this authorization design model for a commercial EC system will be provided later in this thesis when we apply our derived design models to a SET-integrated e-commerce system.

In the next section, we will apply our methodology to derive an EC design model for the access control enforcement feature in the NIST security services model.

COUNTERMEASURES	AUTHORIZATION SECURITY ATTACK TYPE					
		ID Spoofing	Replay attacks	Privilege brute-force attacks	Authorization bypassing	Session Hijacking
	Strong Session Management	D	D			
	Explicit Authorization				I	
	Alarm on unprivileged authorization attempts			D		
	Strong session encryption					D
	Cookie session enforcement					D
	URL session enforcement					D

D = design time, I = after system implementation

Figure 4.8. Validation traceability matrix for authorization countermeasures versus authorization security attacks.

4.4. Security-Oriented Access Control Enforcement Design Model

Step 1 is to select a NIST model security feature. Here we select *access control enforcement.*

When a subject requesting access has been validated for access to particular processes, through authorization, enforcing the defined security policy is still necessary. The *access control enforcement* service provides this enforcement, and frequently the enforcement mechanisms are distributed throughout the system. This does not mean only protecting the EC system from unauthorized access but also having control of authorized access. [NIST, 2001]

It is not only the correctness of the access control decision, but also the strength of the *access control enforcement* that determines the level of security obtained. Checking identity and requested access against access control lists is a common *access control enforcement* mechanism. In general, it is simply a list of the EC system services available, each with a list of the services and/or users permitted to use the service. [Malkin, 1996]

Security architects may depend on research approaches as an asset to derive access control policies. An example of such a research is a paper by Bonatti et al. that describes a modular approach to composing access control policies. [Bonatti, 2000]

The above referenced literature emphasizes satisfying standard security requirements. As discussed earlier, the main emphasis in current standards for security requirements is to satisfy legitimate user requirements from a security point of view. Testing against the existence of malicious user requirements is done after the system is implemented. In this thesis, we emphasize blocking malicious user requirements at system design time (this is also known as preventive security design [Jonsson, 1998]). The goal of this section is to

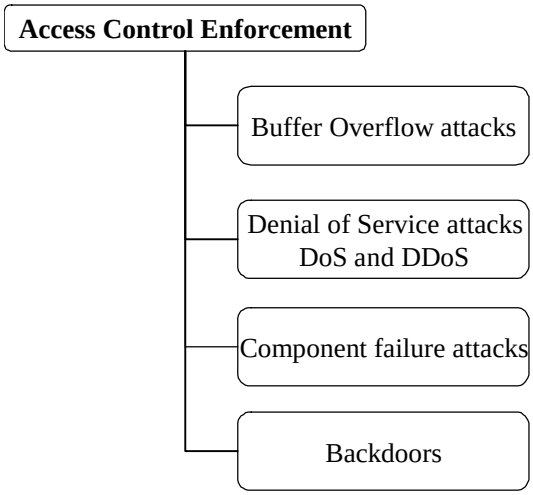

Figure 4.9. Security attacks related to access control enforcement.

have a preventive design model for *access control enforcement* that can be incorporated into any integrated *access control enforcement* model (such as above) or can be implemented as a standalone *access control enforcement* module in e-commerce systems.

4.4.1. Step 2.1: Identify all Security Attacks Related to Access Control Enforcement

Security attacks related to access control enforcement can be identified from the literature and from personal experience. In this thesis, we have projected all known security attacks onto various types of e-commerce access control models and frameworks described in [Botha 2002], [Chuang, 1997], [Bertino, 2000], [Bertino, 2001], and [Ferraiolo, 2001]. This comprises most known attacks applicable to *access control enforcement*. These four attacks are summarized in Figure 4.9 listed in decreasing order of commonality and are described next.

Please note the difference between access control and access control enforcement. Access control is a term that covers the authorization process and the access control enforcement

process. Access control enforcement is not directly related to the process of authorization itself. The NIST model relies on the authorization feature for validating access requests. Thus, the majority of the attacks related to access control, in general, were discussed in section 4.3. In this section, we will only discuss security attacks targeted towards the process of enforcing access control in EC systems.

It is important to remind that the main focus of this thesis will be on attacks directly related to e-commerce systems. Attacks related to network components, to third-party software components, and attacks against the operating system that is supporting the application will not be discussed here. In practice, of course, these types of attacks also have to be handled.

The specific security attacks related to *access control enforcement* in e-commerce systems are ([CERT, 1997] and [Viega, 2002]):

- Buffer overflow attacks.
- Denial of service attacks (DoS), as well as distributed denial of service attacks (DDoS).
- Component failure attacks, as a result of Denial of Service attacks.
- Backdoors, also known as trapdoors.

We will now consider each type of attack, derive attack enablers, and prescribe effective countermeasures.

4.4.2. Step 2.2: For each access control enforcement attack, derive its enablers and countermeasures

This section provides a succinct description of all known *access control enforcement* related security attacks. Attack enablers are then identified, and effective countermeasures are prescribed. Figure 4.10 shows all security attacks related to *access control enforcement* in e-commerce systems (a) along with the attack enablers (b) and

prescribed countermeasures (c). We will refer to this figure as we proceed with our discussion about security attacks below.

4.4.2.1. Buffer Overflow Attacks

A *buffer overflow* (a5) is possibly the most prevalent software vulnerability used to infiltrate a computer or a system. [Viega, 2002] A buffer is a memory location in which a program stores variable data. This data is often supplied directly by the user, as in a name or other text entry, or by a client program such as a web browser or email client. These buffers are usually of a fixed, predetermined size. If the program restricts the amount of data that can be input to the amount of data that the buffer can store, a buffer overflow is not possible. Some programs do not do this. Buffer overflow occurs when more data is entered than there is space for in the buffer. The extra data then gets written to the memory locations after the buffer. Often, the memory after the buffer location originally contains code, and that memory location is referenced for future execution. A buffer overflow attack is possible when an attacker is able to input data that will cause either a program crash (creating a Denial of Service attack) or cause the system to run code granting the attacker further access.

The *attack enabler* for buffer flow attacks is, as discussed above, not restricting the amount of data that can be input to the amount of data that the buffer can store (b5). Restricting the size of data on the client side in EC systems is infeasible because clients are usually web browsers. EC systems cannot depend on web browsers from a security point of view because malicious users have full control of the web browser.

The proper *countermeasure* is to use static or dynamic source code analyzers (c5) to check the written code for buffer overflow problems. Another possible countermeasure is to change the programming language compiler so that it performs bounds checking for protecting certain addresses from overwriting.

4.4.2.2. Denial of Service Attacks

A denial of service attack [CERT, 1997] is characterized by an explicit attempt by attackers to prevent legitimate EC system users from using the system. Examples include attempts to "flood" a network, thereby preventing legitimate network traffic; or attempts to disrupt connections between two machines, thereby preventing access to a certain specific service supporting the EC system.

Not all service outages, even those that result from malicious activity, are necessarily denial-of-service attacks. Other types of attack may include a denial of service as a component, but the denial of service may be part of a larger attack. Denial-of-service attacks can essentially disable the EC system or the network that the system resides on. Some denial-of-service attacks can be executed with limited resources against a large, sophisticated site. This type of attack is sometimes called an "asymmetric attack."

Denial of service attacks are therefore, related to computer networks. Thus, our research will not deal with this type of attacks. Yet, we are interested in one subsidiary of this type of attacks: component failure attacks (a4).

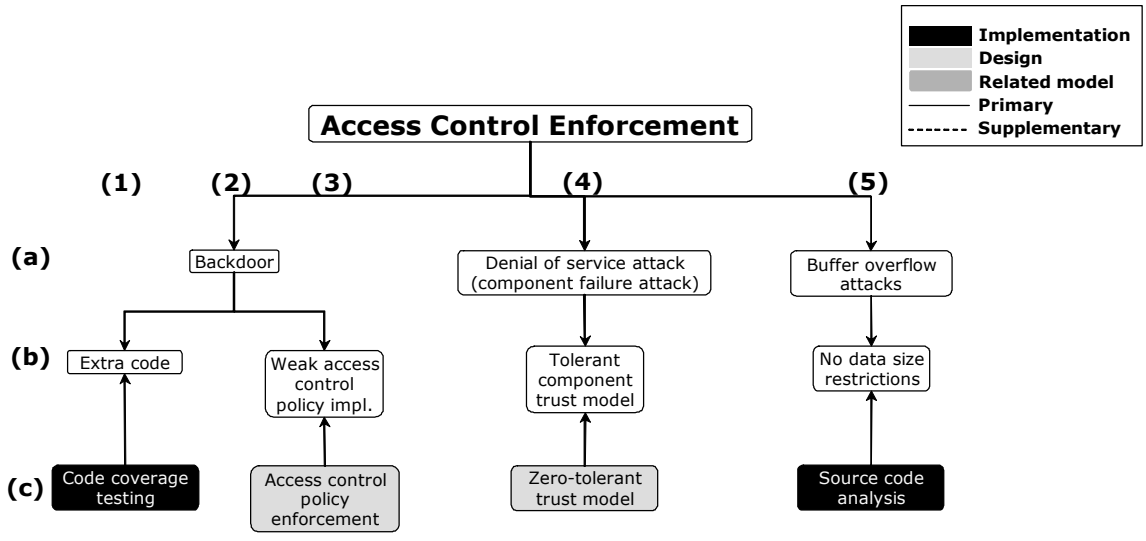

Figure 4.10. Organizational chart for access control enforcement security attacks, attack enablers, and countermeasures.

4.4.2.3. Component Failure Attacks

Component failure attacks (a4) occur when an EC system component fails to respond to other components while providing a service. Malicious users might execute a denial of service attack on a certain component of the EC system with an attempt to disable it. If successful, malicious users will then attempt to attack the EC system through simpler attacks such as spoofing attacks, replay attacks, and session hijacking attacks.

An example of such attacks might occur in the case of centralized authorization. Malicious users might execute a denial of service attack against the EC system component that provides authorization mechanisms. If the attack fails, malicious users will attempt to perform a session hijacking attack on the EC system using erroneous authorization information.

The *attack enabler* is when system components provide tolerance for erratic trusted components (b4). This might lead to unsafe trust relationships, thus, rendering the EC system behavior unpredictable.

The proper *countermeasure* for this type of attack is to have a zero-tolerant trust model (c4). [Kahn, 1999] By doing so, components that fail while providing the service will be considered un-trusted and, thus, the attack fails.

4.4.2.4. Backdoors

A *backdoor*, also called a *trapdoor*, is an undocumented way of gaining access to an EC system. [Viega, 2002] Backdoors (a2), whether intentional or non-intentional, are usually written by the developer who writes the code for the EC system and are often only known by the developer who wrote them. [GMU]

Backdoors might exist in any component of the EC system. Backdoors related to the operating system platform (such as backdoors caused by operating system services like

telnet and FTP) are beyond the scope of this thesis. For the purpose of our discussion, we are concerned with backdoors that are directly related to our EC system. For this purpose, two types of backdoor are identified: coding backdoors and underlying backdoors.

Coding backdoors are usually generated while coding the EC system. Developers might, intentionally or non-intentionally, write code that would provide illegal access to the EC system.

The *attack enabler* in this case is having extra code (b1) that does not support the proper functionality of the EC system. The proper *countermeasure* (c1) is to have good code coverage testing (also known as white box testing). As a result of applying a good code coverage testing process, uncovered code can be identified and observed.

Underlying backdoors are backdoors existing in critical components supporting the EC system such as web servers. An example of such is having weak file access permissions or weak web server aliases.

Weak file access permissions might give unauthorized access to insiders to the EC system. An insider, in this case, will loginto the system as a legitimate user, and then exploit these file access permissions to perform his attack.

Web server aliases, on the other hand, might give unauthorized access to any type of user through the Internet. In this case, a web server alias, created with the intention of simplifying a user's task, might give uncontrolled access to other system components.

The *attack enabler* in both cases is a weak implementation of the access control policy (b3). The proper *countermeasure* is to enforce the access control policy (c3) through an explicit check after system installation. By doing so, any existing backdoor will be uncovered and fixed.

4.4.2.5. Access Control Enforcement Security Attacks Summary

Figure 4.10 shows all security attacks related to *access control enforcement* in e-commerce systems (a) along with the attack enablers (b) and prescribed countermeasures (c). Moreover, this figure is complete with respect to the current state of knowledge.

For example, as shown in Figure 4.10, *backdoor attacks* (a2) require two countermeasures. The first countermeasure, *enforcing access control policy* (c3) throughout the EC system, is applied during system design time. The second countermeasure, *code coverage testing* (c1), requires the availability the EC system source code. Thus, it cannot be applied during system design phase. Yet, this countermeasure is directly helpful in providing implementation and testing guidelines prevent backdoors from occurring. Moreover, *code coverage testing* can be automated through a variety of tools and, thus, serves as an important advantage for introducing this type of countermeasure in our design model.

As discussed earlier, we are not directly concerned with *denial of service attacks* since they are not directly related to EC systems. Our interest is in *component failure attacks* (a4) that might occur as a result of denial of service attacks. By providing a *zero-tolerant trust model* (c4), our EC system will fulfill its requirement for security against denial of service attacks, i.e. preventing illegitimate access to the system.
Buffer overflow attacks (a5) are directly related to system coding. Some programming languages such as Java™ are not affected with this type of attacks. Yet, it is important to have a proper countermeasure in place if a programming language that can be affected is to be used. *Source code analysis* (c5) can prevent *buffer flow attacks* (a5) from succeeding but cannot be applied at system design time. Yet, this countermeasure is directly helpful in providing implementation and testing guidelines to prevent *buffer overflow attacks* (a5) from succeeding. Another advantage for having this type of countermeasure is the possibility of automating the analysis process through static and dynamic analyzers.

4.4.3. Step 2.3: Derive the Access Control Enforcement Security-Oriented Design Model

After identifying the attacks, attack enablers, and countermeasures for *access control enforcement*, the prescribed countermeasures are grouped and ordered in a countermeasures design model. The process of deriving the design model was described earlier in section 4.2.3 of this thesis.

Figure 4.11 shows the detailed *access control enforcement* countermeasures design model derived by applying our methodology. As seen in the figure, the main entry point to the model is zero-tolerance. In our earlier discussion, we explained this as not trusting malicious behavior from trusted EC system components. With a zero-tolerance trust model in place, we then proceed to enforcing the access control policy. If any error occurs at any point, the process goes back to its starting point and access to the EC system is refused. In other words, access to the EC system is only given if the component requesting it is in a trustworthy state and if the access control policy is respected.

Code coverage testing and source code analysis are two countermeasures at the basis of our access control enforcement model. Without these countermeasures in place, attacking

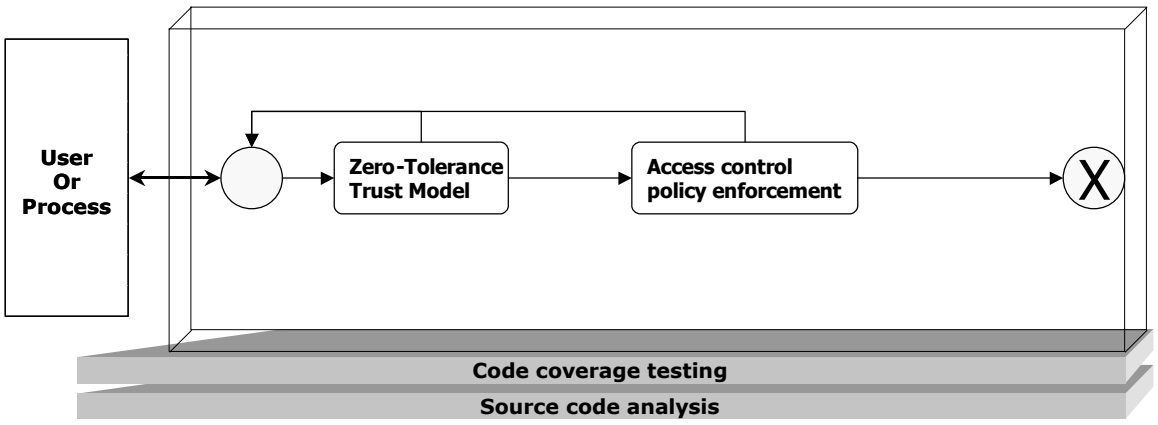

Figure 4.11. The derived e-commerce access control enforcement countermeasures design model.

the access control enforcement countermeasures design model has a great chance of succeeding.

The *access control enforcement* design model shown above is detailed enough to be directly applied to the design of EC systems. This model satisfies security requirements and blocks malicious user requirements at system design time. It provides effective security against all known security attacks related to *access control enforcement* in the e-commerce domain.

4.4.4. Step 2.4: Validate the Security-Oriented Access control Enforcement Design Model

We now validate the model by checking the adequacy of its coverage with respect to attacks and residual vulnerabilities, using the *validation traceability matrix* in Figure 4.12.

Countermeasures against *denial of service attacks* and *weak access control policies* were introduced at system design time. On the other hand, countermeasures against *buffer*

COUNTERMEASURES	ACCESS CONTROL ENFORCEMENT SECURITY ATTACK TYPE				
		Denial of Service	Buffer overflow	Backdoors	
				Extra code	Weak access control policy
	Code coverage testing			I	
	Access control policy enforcement				D
	Zero-tolerant trust model	D			
	Source code analysis		I		

D = design time, I = after system implementation

Figure 4.12. Validation traceability matrix for access control enforcement countermeasures versus related security attacks.

overflow attacks and *backdoor attacks* require system implementation. Thus, they cannot be applied at system design time and must be applied during the EC system-testing phase.

An example of how to instantiate this countermeasures design model in a commercial EC system will be provided later in this thesis when we apply our derived countermeasures design models to a SET-integrated e-commerce system.

In the next section, we will apply our methodology to derive an EC design model for the *transaction privacy* feature in the NIST security services model.

4.5. Security-Oriented Transaction Privacy Design Model

Step 1 is to select a NIST model security feature. Here we select *transaction privacy*.

According to [NIST, 2001], both government and private systems are increasingly required to maintain the privacy of individuals using EC systems. The *transaction privacy* service protects against loss of privacy with respect to transactions being performed by an individual on the EC system.

In [Candolin, 2000], the concept of *transaction privacy* is defined as a service for preventing unauthorized disclosure of transaction contents, parties involved, location of parties involved, and the exact time of occurrence of the transaction. *Transaction privacy*, thus, includes the following:

- Data privacy: the contents of the transaction must be protected from disclosure to unauthorized parties.
- Source and destination privacy: the parties involved in the transaction should not be revealed to unauthorized parties.
- Location privacy: the location of the parties performing the transaction should not be disclosed to unauthorized parties.
- Time privacy: the exact time when a transaction occurs should not be disclosed to unauthorized parties.

Security architects may depend on research approaches as an asset to derive *transaction privacy* models. An example of such a research is the SET specification document [SET, 1997-1] that serves as an open standard for protecting the privacy, and ensuring the authenticity, of electronic transactions.

The above referenced literature emphasizes satisfying standard security requirements. As discussed earlier, the main emphasis in current standards for security requirements is to satisfy legitimate user requirements from a security point of view. Testing against the

existence of malicious user requirements is done after the system is implemented. In this thesis, we emphasize blocking malicious user requirements at system design time (this is also known as preventive security design [Jonsson, 1998]). The goal of this section is to have a preventive design model for *transaction privacy* that can be incorporated into any integrated *transaction* model (such as above) or can be implemented as a standalone *transaction privacy* module in e-commerce systems.

4.5.1. Step 2.1: Identify all Attacks Related to Transaction Privacy

Security attacks related to *transaction privacy* can be identified from the literature and from personal experience. In this thesis, we have projected all known security attacks onto various types of e-commerce access control models and frameworks described in [Bouchard, 2000], [Liddy, 1998], and [SET, 1997-1]. This comprises most known attacks applicable to *transaction privacy*. These three attacks are summarized in Figure 4.13 listed in decreasing order of commonality and are described next.

It is important to remind that the main focus of this thesis will be on attacks directly related to e-commerce systems. Attacks related to network components, to third-party software components, and attacks against the operating system that is supporting the application will not be discussed here. In practice, of course, these types of attacks also have to be handled.

The specific security attacks related to *transaction privacy* in e-commerce systems are ([Herzog, 2001], [Viega, 2002], [Nguyen, 2001], and [Schneier, 2000]):

- DBMS exploits, or attacks targeted towards exploiting security of Data Base Management Systems
- Log data mining attacks, also known as log data analysis attacks
- Sniffing attacks, also known as man-in-the-middle attacks

We will now consider each type of attack, derive attack enablers, and prescribe effective countermeasures.

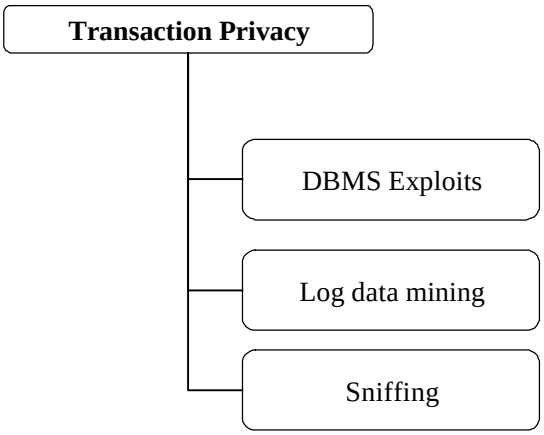

Figure 4.13. Security attacks related to transaction privacy.

4.5.2. Step 2.2: For each Transaction Privacy Attack, Derive its Attack Enablers and Countermeasures

This section provides a succinct description of all known transaction privacy related security. Attack enablers are then identified, and proper countermeasures are prescribed. The attacks are presented and discussed below in order of dependence, since some of them are related (e.g. DBMS exploits attacks depend on log data mining attacks). Figure 4.14 shows all security attacks related to *transaction privacy* in e-commerce systems (a) along with the attack enablers (b) and prescribed countermeasures (c). We will refer to this figure as we proceed with our discussion about security attacks below.

4.5.2.1. Sniffing Attacks

As described earlier in section 4.2.2, *sniffing attacks* ([Herzog, 2001], [Viega, 2002], [Nguyen, 2001], and [Schneier, 2000]) (also known as the man-in-the-middle attacks) are the digital analogues to phone tapping or eavesdropping (a1). This attack captures information as it flows between a client and a server. In the case of transaction privacy, such attacks might be successful at retrieving transaction information while the transaction is being performed.

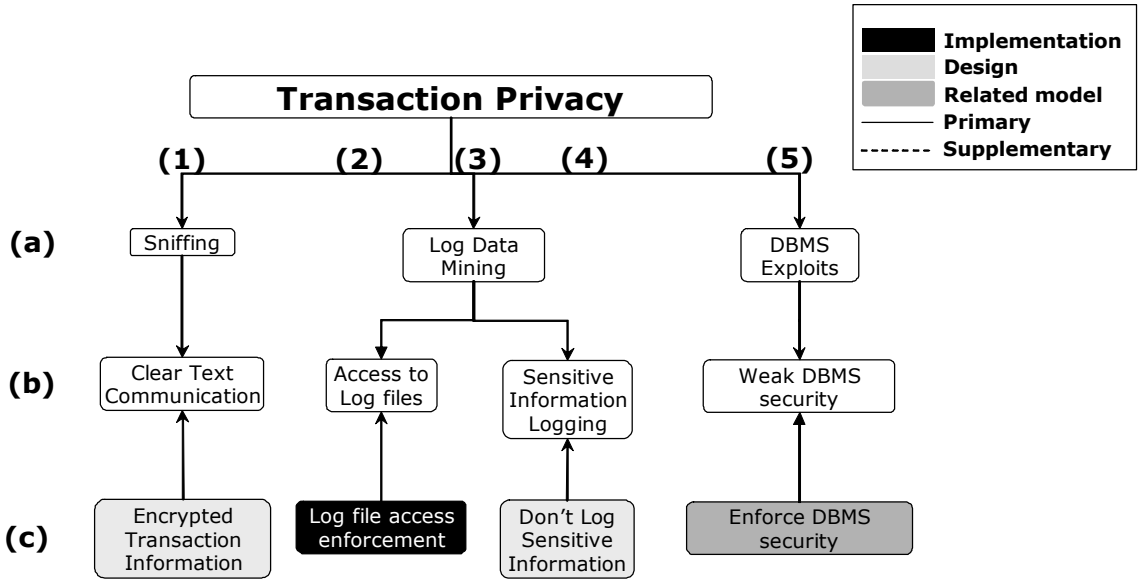

Figure 4.14. Organizational chart for transaction privacy security attacks (a), attack enablers (b), and countermeasures (c).

The *attack enabler* in this case is the process of sending data across communication channels in clear text format (b1). Preventing access to the communication channel is not a valid countermeasure in this case due to the open nature of the Internet.

By encrypting the communication channel between the user/process and the system, sniffing attacks are defeated, i.e., sniffing cannot retrieve any useful information. Yet, encryption might have an impact on the EC system from a performance perspective. Thus, the *proper countermeasure* for this attack is to encrypt the transaction information itself, at least, instead of encrypting all data being communicated between the client and the EC system (c1).

4.5.2.2. Log Data Mining Attacks

Data mining is the search for significant patterns and trends in large databases by using sophisticated statistical techniques and software. This provides information crucial to

helping businesses and industries improve products, marketing, sales, and customer service.

Similarly, *log data mining attacks* (a3) can be targeted towards retrieving private transaction information from EC systems' log data files. The *attack enabler* for this type of attack has two properties: having access to the log data file (b2), and logging sensitive transaction information in the log data file (b4).

The first *attack enabler* property, *physical access to the log data file* (b2), allows malicious users to retrieve a copy of the logged data and perform log data mining attacks. The proper *countermeasure* for this attack enabler property is to enforce access permissions on the log data file at the underlying operating system level (c2). By doing so, malicious users will not be able to have access to the log data file and, thus, the attack is disabled.

The second *attack enabler* property, logging sensitive information (b4), allows *log data mining attacks* to retrieve useful transaction information in case malicious users gain access to the log data file. This is highly probable in the case of malicious users categorized as insiders. The proper *countermeasure* for this attack enabler property is to prevent the EC system from logging sensitive data (c4). This will prevent malicious users from succeeding in retrieving useful transaction information. In case sensitive data logging is required, database management systems (DBMS) must be used to save this type of data.

A *residual vulnerability* for using DBMS to log sensitive data is the fact that the EC system security will be dependent on the security of the DBMS (a5). This residual vulnerability is discussed next.

4.5.2.3. DBMS Exploits

As discussed above, relying on Data Base Management System (DBMS) security is considered a residual vulnerability. This is because malicious users might be able to exploit the EC system by exploiting the DBMS itself (a5).

The *attack enabler* in this case might be any exploit in DBMS security (b5). The proper *countermeasure* is to enforce security at the DBMS level by keeping it up-to-date with security fixes and patches (c5). This will help prevent malicious users from exploiting our EC system by exploiting the DBMS system that we rely on for transaction privacy in general, and sensitive data logging in particular.

4.5.2.4. Transaction Privacy Security Attacks Summary

Figure 4.14 shows all security attacks related to *transaction privacy* in e-commerce systems (a) along with the attack enablers (b) and prescribed countermeasures (c). Moreover, this figure is complete with respect to the current state of knowledge.

For example, as shown in Figure 4.14, all countermeasures related to *transaction privacy* were introduced and applied at system design time except for *log file access enforcement* (c2) that has to be applied after the EC system is implemented. Yet, this countermeasure, in this case, is transformed to a requirement for the EC system and is directly useful in providing implementation and testing guidelines for the ecommerce systems.

Two countermeasures are applied at system design time. *Encrypted transaction information* (c1) blocks *sniffing attacks* (a1). Preventing the EC system from logging *sensitive transaction information* (c4) is a countermeasure against *log data mining attacks* (a3).

Enforcing DBMS security (c5) is a countermeasure related to the DBMS instead of being a part of the EC system. Thus, it cannot be a part of the EC system design or its

requirements. Yet, the purpose of having it in the EC countermeasures design model is to force EC system architects to take DBMS security into consideration when providing EC security documentation. This documentation must include proper steps on how to enforce DBMS security along with a clear explanation why this enforcement is required.

4.5.3. Step 2.3: Derive the Transaction Privacy Security-Oriented Design Model

After identifying the attacks, attack enablers, and countermeasures for *transaction privacy*, the prescribed countermeasures are grouped and ordered in a countermeasures design model. The process of deriving the design model was described earlier in section 4.2.3 of this thesis.

Figure 4.15 shows the detailed transaction privacy countermeasures design model for EC systems derived by applying our methodology. As seen in the figure, the communication channel between the user or process and the EC system has to be encrypted for transaction privacy purposes. If this is not the case, then at least sensitive transaction information should be encrypted. While performing a transaction, the EC system should

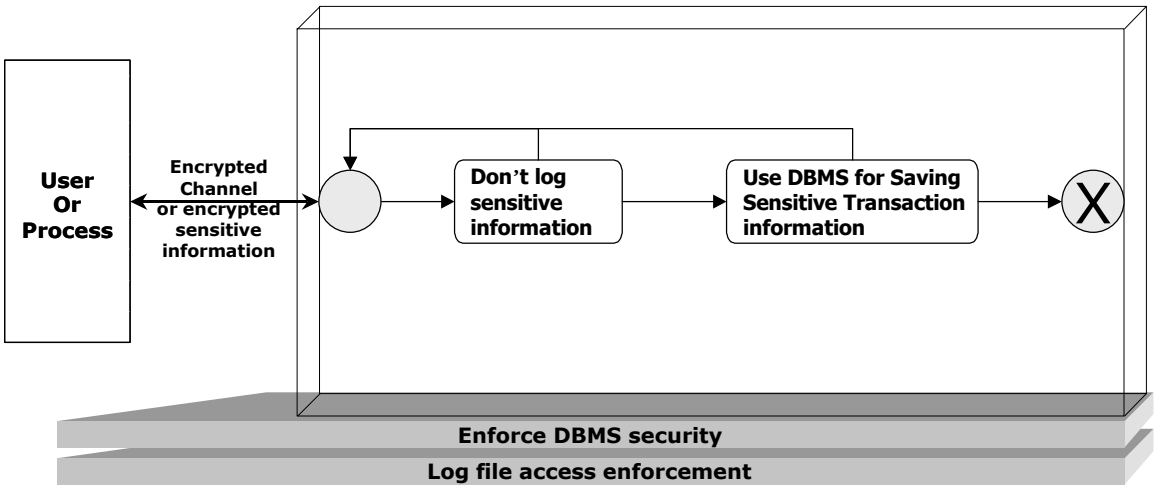

Figure 4.15. The derived EC transaction privacy countermeasures design model.

not log sensitive transaction information. In the case of required logging, such information should be saved in the DBMS. The only path for a successful transaction privacy process is to have all the countermeasures performed. If any countermeasure is skipped, the model goes back to its entry point and the transaction fails from a security point of view. As discussed earlier, two countermeasures support the transaction privacy model: DBMS security enforcement and log file access enforcement.

The EC transaction privacy countermeasures design model shown above is detailed enough to be directly applied to the design of EC systems. This model satisfies security requirements and blocks malicious user requirements at system design time. It provides effective security against all known security attacks related to transaction privacy in the e-commerce domain.

4.5.4. Step 2.4: Validate the Security-Oriented Transaction Privacy Countermeasures Design Model

We now validate the model by checking the adequacy of its coverage with respect to attacks and residual vulnerabilities using the *validation traceability matrix* in Figure 4.16.

Countermeasures for sniffing, DBMS exploits, and access to log files were all incorporated at system design time. The only countermeasure to be still applied after system implementation is *log file access enforcement*.

In the next section, we present an evaluation of our methodology based on the Case Study.

COUNTERMEASURES	TRANSACTION PRIVACY SECURITY ATTACK TYPE				
		Sniffing	DBMS Exploits	Log Data Mining	
				Access to Log files	Sensitive information logging
	Encrypted transaction information / communication channel	D			
	Log file access enforcement			I	
	Don't log sensitive information				D
	Enforce DBMS security		D		

D = design time, I = after system implementation

Figure 4.16. Validation traceability matrix for transaction privacy countermeasures versus related security attacks.

4.6. Evaluation of Methodology

In the previous sections, we have applied our design-for-security methodology to derive countermeasures design models for authentication, authorization, access control enforcement, and transaction privacy. Of course, there are other countermeasures that might be used, but these are the primary countermeasures for the four security features.

In this section, we evaluate our experience of applying the methodology to these four important security features, giving benefits and limitations of our approach, and extrapolating to the remaining ten key security features defined by NIST. [NIST, 2001]

4.6.1. Advantages of our Methodology

In this section, we discuss seven advantages of applying our methodology, compared to present security design and implementation strategies.

The first advantage is our methodology's ability to *satisfy standard security requirements at system design time*. We have shown this by basing our case study on the NIST security services model. [NIST, 2001] This model satisfies standard security requirements provided in the Common Criteria Redbook [CC, 1999]. Our methodology specializes the NIST model by adding effective countermeasures against malicious user requirements. Thus, standard security requirements according to the NIST model will be provided in our derived security-oriented design models for authentication, authorization, access control enforcement, and transaction privacy.

The second advantage of our methodology is the ability to *address and block malicious user requirements at system design time*. This advantage was discussed earlier in sections 4.2.4, 4.3.4, 4.4.4, and 4.5.4 of this chapter using the validation traceability matrix. This matrix showed that all countermeasures were introduced at system design time. Some of these countermeasures required system implementation. However, introducing them at

system design time is helpful in providing implementation guidelines and test planning.

The validation traceability matrix is used to obtain the third benefit of our methodology, i.e. providing *effective security against all known security attacks related to the e-commerce domain* for authentication, authorization, access control enforcement, and transaction privacy. This matrix is a very simple, yet very effective tool for assuring that every attack is addressed by at least one countermeasure (no empty columns), and that every countermeasure is needed to guard at least one attack (no empty rows).

The fourth advantage of our methodology is that it *can be directly integrated into high-level design documents of e-commerce systems*. This arises from the fact that our derived models are represented as flowcharts and underlying planes. Flowcharts represent actions, including countermeasure actions, that the e-commerce system must perform to provide security. Underlying planes represent countermeasures that must be present but are not actions. In both cases, these countermeasures can be specified as design security requirements for the e-commerce system.

The fifth advantage of our methodology is that it *can be used with security-aware technologies* such as CORBA™, and SET, or *can be implemented as a stand-alone module for EC systems*. A case study will be presented in the next chapter where we apply our derived security models to a SET-integrated e-commerce system.

The sixth advantage of our methodology is that it *provides guidelines for implementation and testing*. This advantage was shown earlier while discussing our derived models for authentication, authorization, access control enforcement, and transaction privacy. This specifically applies for countermeasures that are introduced at system design time but can be applied only after the system is implemented. An example of such a countermeasure for authentication is “cryptography implementation check”. This countermeasure can be applied after the system is implemented. However, introducing this countermeasure at system design time allows security architects to emphasize the importance of cryptography implementation and to plan for testing cryptography implementation after

the system is implemented.

The seventh advantage of our methodology is that it is *systematic and modular in nature*. Every step of the methodology can be represented as a module or process that takes a set of input arguments and produces a set of output results. Furthermore, the methodology steps are not inter-related with each other from a functional point of view. This enables distributing tasks while applying the methodology since performing one step does not require any information from other steps. If any relation is to be introduced, it has to be through the set of input and output arguments only.

4.6.2. Limitations

In this section we discuss limitations of our methodology based on our experience applying it to derive the four security-oriented design models for authentication, authorization, access control enforcement, and transaction privacy in this case study.

The first limitation of our methodology is that it does not provide security against future attacks, as it is limited to the set of all known attacks. This limitation is natural and is derived from step 2.1 where we identify and abstract all known security attacks related to the security feature.

The second limitation is that the methodology is not systematic for discovering residual vulnerabilities of prescribed countermeasures. In other words, prescribed countermeasures in step 3.2 must be thoroughly analyzed and discussed in step 3.3.1 to discover any residual vulnerability.

The third limitation of our methodology is that prescribed countermeasures might impact performance if prescribed countermeasures in steps 3.2 and 3.3.2. did not take performance into consideration.

4.6.3. Potential for Extrapolation to Other NIST Security Features

In this section, we describe the potential for extrapolating our methodology for other security features.

We strongly believe that applying the methodology to other security features should be straightforward. However, the limitations discussed earlier must be taken into consideration. In other words, prescribing countermeasures must take performance into consideration. As well, countermeasures must be thoroughly analyzed and discussed to discover any residual vulnerabilities.

In the next chapter, we will apply the derived countermeasures design models for authentication, authorization, access control enforcement, and transaction privacy to a SET-integrated e-commerce system. This corresponds to the last step of our methodology.

Chapter V

Case Study Phase 2: Instantiating and Integrating the Derived Countermeasures Design Models to a SET-Integrated E-Commerce System

In chapter 4, we derived countermeasures design models for four security features of the NIST security services model, namely authentication, authorization, access control enforcement, and transaction privacy. We also verified the effectiveness of these models against the set of all known security attacks. In this chapter, we will apply the derived countermeasures design models to an e-commerce system application. The system we describe is a Secure Electronic Transaction (SET) integrated e-commerce system that allows its users to select products from a catalog, place orders, submit orders, and pay online. This is a typical security design approach for e-commerce systems. [Treese, 1998]

In section 5.1, we provide an overview of the standard for Secure Electronic Transactions (SET). In section 5.2, we integrate SET into an e-commerce system. In sections 5.3, 5.4, 5.5, and 5.6, we apply our derived countermeasures design models for authentication, authorization, access control enforcement, and transaction privacy respectively, in the detailed design of the SET-integrated e-commerce application. In section 5.7, we will compare security presence in the case study e-commerce system before and after applying our countermeasures design models. In section 5.8, we summarize and conclude.

5.1. Secure Electronic Transactions (SET)

In this section we provide an overview of the Secure Electronic Transactions (SET) specification.

5.1.1. Introduction

Secure Electronic Transactions (SET) is an open technical standard for e-commerce developed by Visa and MasterCard, in conjunction with leading computer vendors such as IBM, as a way to facilitate secure payment card transactions over the Internet. SET is an open standard for protecting the privacy of and ensuring the authenticity of electronic transactions. *Digital Certificates* create a trust chain throughout the transaction, verifying cardholder and merchant validity. A *digital certificate* is a message which ensures the identity of an entity (whether a cardholder or a merchant) by identifying a trusted registration authority, naming or identifying the entity, specifying the entity public key, and including a digital signature of the registration authority. This is critical to the success of electronic commerce over the Internet; without privacy, consumer protection cannot be guaranteed, and without authentication, neither the merchant nor the consumer can be sure that valid transactions are being made. [IBM, 1998]

5.1.2. SET Motivation and Objectives

SET specifies its primary motivations as follows: [SET, 1997-1]

- To encourage the payment card community to take a leadership position in establishing a secure payment specification
- To avoid costs associated with future reconciliation of implemented approaches,
- To facilitate rapid development of the marketplace,
- To respond quickly to the needs of the financial services market, and

- To protect the integrity of payment card brands.

SET has three goals: *payment security*, *interoperability*, and *market acceptance*. [SET, 1997-1] Each of these goals is described below in details.

The payment security goal is to provide the following security services:

- To authenticate cardholders, merchants, and acquirers,
- To provide confidentiality of payment data,
- To preserve the integrity of payment data, and
- To define the algorithms and protocols necessary for these security services.

The interoperability goal is to:

- Provide all required details to ensure that applications developed by various vendors will interoperate,
- Create and support an open payment card standard. This implies better interoperability because different platforms will be able to implement the standard.
- Define exportable technology throughout, to encourage globally interoperable software,
- Ensure compatibility with and acceptance by appropriate standards bodies, and
- Support implementation on any combination of hardware and software platforms, such as PowerPC, Intel, Sparc, UNIX, MS-DOS, OS/2, Windows, and Macintosh OS.

The market acceptance goal is to:

- Achieve global acceptance via ease of implementation with minimal constraints on merchant and cardholder end users,
- Allow for easy implementation of the payment protocol within existing client applications,
- Minimize changes to the relationship, existing applications, and infrastructure between acquirers and merchants, and cardholders and issuers, and
- Provide a protocol that will be efficient for financial institutions.

5.1.3. SET Security

The SET protocol relies on two different encryption mechanisms, as well as an authentication mechanism. SET uses symmetric encryption, in the form of the aging Data Encryption Standard (DES), as well as asymmetric, or public-key cryptography to transmit session keys for DES transactions. Rather than offer the security and protection afforded by public-key cryptography, SET only encrypts session keys (56 bits) which asymmetrically– the remainder of the transaction uses symmetric encryption in the form of DES. This has disturbing connotations for a "secure" electronic transaction protocol– because public key cryptography is only used to encrypt DES keys and for authentication, and not for the main body of the transaction. The computational cost of asymmetric encryption is cited as reason for using weak 56 bit DES; however, other reasons such as export/import restrictions, and the perceived need by law enforcement and government agencies to access the plain-text of encrypted SET messages may also play a role. [IBM, 1998] Thus, SET is not secure as it could be.

5.1.4. SET Architecture

The SET system is composed of a collection of electronic commerce entities. [SET, 1997-1]

The collection consists of:

- *Cardholder*, an authorized holder of a payment card supported by an Issuer, registered to perform electronic commerce;
- *Merchant*, a person or organization providing goods, services, and/or information who accepts payment for them electronically;
- *Issuer*, a financial institution that issues payment card products to individuals;
- *Acquirer*, a financial institution that supports merchants by providing services for processing payment card transactions;

- *Payment Gateway*, a system that provides electronic commerce services to the merchants in support of the *Acquirer*, and interfaces with the *Acquirer* to support the authorization and capture of transactions, eg. IBM Payment Gateway™;
- *Brand*, a franchiser of payment systems / instruments; eg. Visa™, MasterCard™, etc.
- *Certificate Authority* (CA), an agent of one or more payment card brands that provides for the creation and distribution of electronic certificates for cardholders, merchants, and payment gateways, eg. SET Secure Electronic Transaction LLC; and
- Payment card brand's *financial network*, the existing private network operated by a payment card brand that links *Acquirers* and *Issuers*.

SET also specifies interfaces for communication between the various entities involved. [SET, 1997-2] The formal description of these interfaces for implementation purposes can be found in [SET, 1997-3]. For the purpose of this thesis, every entity will have its own SET module. This SET module is assumed to be compliant to the SET specification of that entity.

5.1.5. SET Conflicts

SET has two intrinsic conflicts between goals and requirements. [Treese, 1998] The first conflict occurs between banks and merchants. The second conflict occurs between cardholders and complexity.

1. While SET offers banks the potential for a reduction in merchant fraud losses by not revealing credit card numbers to merchants, merchants want to integrate SET-based systems easily into their existing operations. Existing systems usually require access to the customer's card number, in contradiction to SET. As a result, SET provides the option of revealing the card number to the merchant.
2. SET uses digital signatures and certificates to ensure the authentication of the cardholder account and merchant. The purpose of having a cardholder certificate is to greatly reduce problems related to the fraudulent use of card numbers. This benefit is achieved at the expense of increased complexity for the cardholder in obtaining,

protecting, and managing the certificate. Furthermore, the benefit does not directly go to the cardholder. [Treese, 1998]. Thus, the current SET specification specifies that cardholder certificates are optional according to the payment card brand's discretion. [SET, 1997-1]

On one hand, SET's architecture is designed to protect the transmission of financial information involved with a payment transaction between a cardholder, merchant, and Acquirer. On the other hand, SET does not impose security requirements on the transmission of the transaction's order information. In other words, SET will perform a certain transaction regardless of whether the transaction order information was delivered securely or not.

A caveat in the SET specification is that it does not provide non-repudiation. [see Chapter 2] This is justifiable because it allows individual payment card brands to provide their own requirements and implementation for non-repudiation. [SET, 1997-2]

This brief introduction to SET is intended to describe its motivation, objectives, security, architecture, and conflicts. This provides a basis for understanding how, in the next section, we will integrate the SET protocol into our case study of securing a SET-integrated e-commerce system.

5.2. A SET-Integrated E-Commerce System

In this section, we present our case study of designing for security a SET-integrated e-commerce system. Our case study e-commerce system, shown in Figure 5.1, is a virtual store system that involves four entities: a merchant EC system, a client, an EC system administrator, and a payment gateway. A description of each entity will follow in sections 5.2.1 through 5.2.4. Section 5.2.5 provides two typical system usage scenarios for a client and an administrator respectively. In this case study, wewill apply our derived countermeasures design models to the merchant entity of the e-commerce system

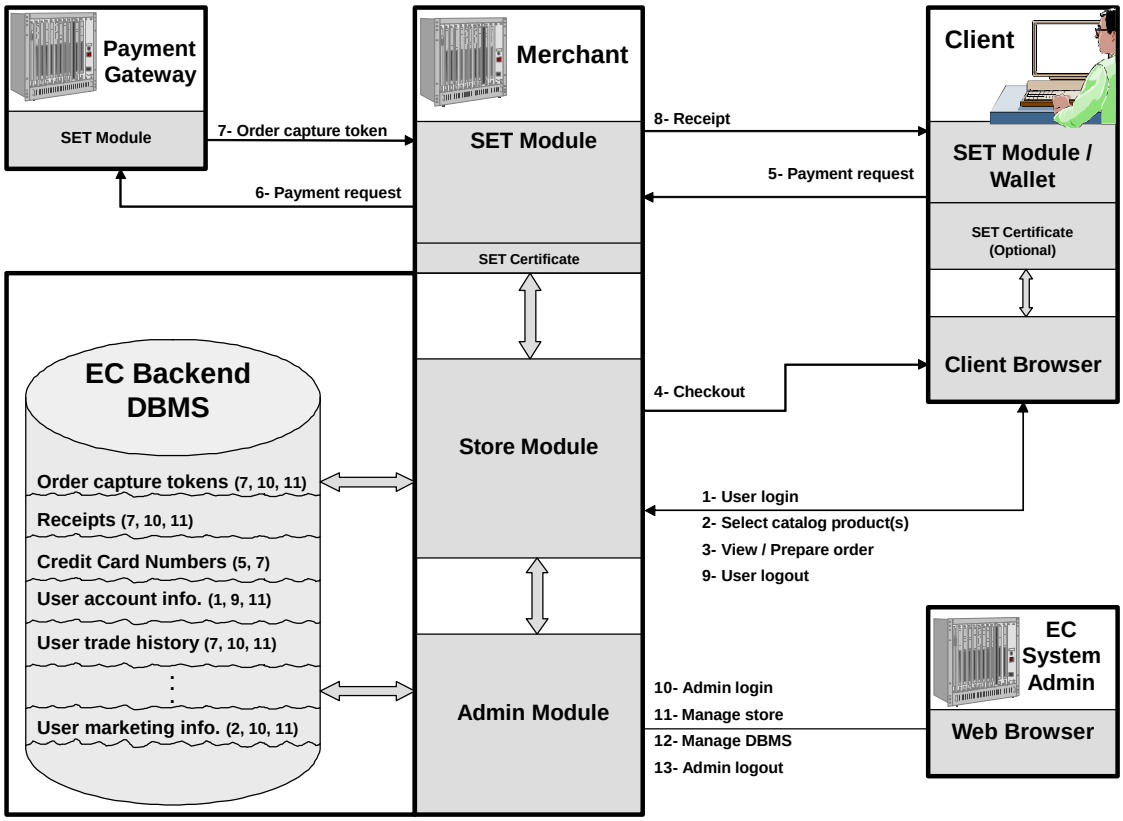

Figure 5.1. A SET-integrated e-commerce system.

only. Designing the security of the other entities (client and payment gateway) will not be considered as the responsibility, in this case, relies on their respective owners; i.e. the client and the payment gateway provider respectively. Other entities described in SET- namely the issuer, acquirer, brand, and certificate authority– are not directly related to our case study e-commerce system. For example, a certificate authority is only required to check the validity of a certificate. This only occurs at the payment gateway side and at the client (cardholder) side.

As stated in section 5.1.4, in this thesis every party requiring a SET implementation will have its own SET module. This SET module is assumed to be compliant to the SET specification of that entity.

5.2.1. Merchant

The merchant EC system is the main part of our case study. As shown in Figure 5.1, this system consists of three main modules: the SET Module, Store Module, and Administrator Module. A description of each of these modules is provided below.

On the other hand, the EC system is both a portal and a virtual store. As a portal, the system requires user account management services. This means that system users have to register to use the system. Furthermore, system users will have the ability to customize certain aspects of the system by providing interests, preferences, etc. As a virtual store, the system allows its users to select catalog products, place orders, pay online, etc. The security of the user is paramount in order to attain a high level of user trust.

5.2.1.1. SET Module

A *SET module* implementing the merchant SET specification is required to provide a SET-compliant transaction processing service for the merchant and clients. As discussed in section 5.1, this type of service is created and runs only during transaction processing.

5.2.1.2. Store Module

A *store module* is required to provide clients with store services such as portal user accounts, online product catalogs, virtual shopping carts, etc. Furthermore, a *store module* can also apply other e-commerce business services such as one-to-one marketing campaigns, advertising campaigns, etc.

5.2.1.3. Administrator Module

An *administrator module* is a typical requirement of e-commerce systems. This module allows EC system administrators to manage, maintain, and update the merchant EC

system. This involves actions like adding product categories, verifying system users, processing and completing orders, etc.

Multiple administrative interfaces usually build up the *administrator module*. This includes interfaces for administering the database, store, orders, marketing, advertising, etc. This is usually done because the system isadministered by more than one person or department. In other words, one person or department might be responsible for maintaining marketing campaigns, while another person or department is responsible for maintaining orders, etc. In some cases, the *administrator module* might be directly integrated into the existing merchant system.

For the purpose of this thesis, we limit our case study to one system administrator responsible for maintaining the EC system through a web interface. This means that the system administrator will connect to the EC system via a web browser. All administrative tasks are assumed to be performed by the *administrator module* of the EC system.

5.2.1.4. Database Management System (DBMS)

Our EC system requires the ability to store, manage, and manipulate data. Thus, a business database management system (DBMS) is also required (The SET module is assume to have its own database management system). Thus, interaction with the EC DBMS is assumed to occur with the store and administrator modules only.

5.2.2. Client

A *client*, or end user, is a registered user of the merchant EC system. The client part consists of a SET module (the cardholder entity module) and a web browser. A *client*, for example, can log onto the system, view catalog products, select catalog products by adding to a virtual shopping cart, preview orders, submit orders, and pay online using the integrated SET module. A typical behavior scenario for registered users is provided in section 5.2.5.

5.2.3. EC System Administrator

An EC *system administrator* is the software entity responsible for maintaining the EC system. In section 5.2.1 we provided an overview of the many different possible architectures and services that such an entity might have. For the purpose of this thesis, we limit our case study to one system administrator responsible for maintaining the EC system through a web interface. This means that the system administrator will connect to the EC system via a web browser. All administrative tasks are assumed to beperformed by the *administrator module* of the EC system. If the EC system administrator is compromised, the entire system is normally compromised.

We also assume that an EC system administrator will not use the *administrator module* to buy products from the EC system. This is a typical assumption, based on the author's experience, that allows for the separation of roles between various system users. Thus, a SET module is not required at the EC system administrator side.

5.2.4. Payment Gateway

A *payment gateway* is required for the merchant and client to perform online payment transactions according to the SET specification. In our case study, we assume that this entity consists of a SET module only. Functional details involved with this entity, such as interactions with banks and credit card brand companies, are also ignored for the purpose of simplification. In other words, our case study assumes that a payment gateway will receive a payment request, process it, and send the result back in a transparent, secure manner.

5.2.5. Typical System Scenarios

In this section, for illustration purposes, we provide two typical functional system scenarios. The first scenario (steps 1-9) mimics a shopping client behavior. The second scenario (steps 10-13) represents a typical EC system administrator behavior. The steps in each scenario are labelled in Figure 5.1 shown earlier.

5.2.5.1. Shopping Client Scenario

A client starts by logging into the EC system (step 1) using his web browser. Once logged in, the user selects catalog products (step 2) and adds them to the shopping cart. The user then previews and checks the order (step 3) and proceeds to checkout (step 4). By asking for checkout (step 4), the client SET module (also known as the SET wallet) is awakened, the merchant is authenticated, and a payment request is prepared, signed, and sent to the merchant SET module (step 5). The merchant SET module signs the payment request and forwards it to the payment gateway (step 6). The payment gateway processes the transaction and sends an "order capture token" to the merchant (step 7). This payment token commits payment from user to merchant and is required later by the merchant to settle the transaction. The merchant sends the client a receipt (step 8). The client is done shopping so he/she logs out of the e-commerce system (step 9).

The scenario depicted above is a straight forward representation of a client performing a normal shopping session.

5.2.5.2. System Administration Scenario

A system administrator starts an administration session by logging onto the EC system (step 10). Once logged in, the administrator module interface can be used to manage the e-commerce system virtual store (step 11) and manage data stored in the database management system (step 12). Once done, the system administrator logs out of the e-commerce system (step 13).

5.2.6. Need for Additional Security

SET secures the transaction while in progress by encryption, but does not take into consideration the security of data on cardholder, merchant, and payment gateway systems including protection from viruses, trojan horse programs, and hackers. [SET, 1997-1]

Furthermore, our discussion in section 5.1.5 requires additional security to address possible SET conflicts with normal principles of online commerce. This includes, but is not limited to the following:

- Protecting card numbers from malicious use when card numbers are revealed to the merchant using the SET reveal option.
- Ensuring proper client authentication since cardholder certificates are optional according to SET
- Securing the transmission of the transaction's order information
- Providing non-repudiation (explained in section 2.1)
- Protecting stored order capture tokens that are required later by the merchant to settle processed transactions.

As a result of the discussion above, SET security is indirectly dependent on the EC system security. For example, a malicious user that successfully penetrates the SET-integrated e-commerce system might be able to capture credit card numbers of legitimate system users or damage order capture tokens.

5.2.7. Summary

In section 5.2, we described the architecture and security aspects of our SET-integrated system along with the different entities involved (merchant, client, administrator, and payment gateway) and the need for additional security.

In sections 5.3 through 5.6, we will apply our derived countermeasures design models for each of the four NIST security features, namely authentication, authorization, access control enforcement, and transaction privacy to the SET-integrated e-commerce system described above. This application facilitates our evaluation of the derived countermeasures design models by comparing system security before and after introducing the derived security models into a simple, but realistic SET-integrated e-commerce system design.

An exciting feature of our approach is that our modeling and analysis activities are carried out at design time. Thus, security is "designed in" and not "tested in" after the initial product has been developed, leading to likely improvements in both quality and time to market.

5.3. Applying and Integrating Authentication Design Model

In section 4.2, we applied our methodology to the authentication security feature of the NIST security services model. The result was a countermeasures design model that is effective against the set of all known security attacks related to authentication. In section 5.2, we described our case study system, namely a SET-integrated e-commerce system. In this section, we will apply the derived countermeasures design model from section 4.2 to the e-commerce system design described in section 5.2, in order to check the effectiveness and applicability of the derived authentication countermeasures design model to a realistic e-commerce system design with SET..

The first step in applying the derived countermeasures design model is to instantiate its features. This is a straightforward process that takes every feature of the model and converts it into an implementable design.

A description of how the authentication model features described in section 4.2 are instantiated is provided below. This process is not dependent on any order of instantiation, and therefore, can be done in any desired order.

5.3.1. Encrypted Channel

The "encrypted channel" feature is instantiated to SSL (Secure Sockets Layer) [Freier, 1996]. SSL is a protocol developed by Netscape Communications Corporation to provide security and privacy over the Internet. This protocol supports server and client authentication, is application independent, and allows HTTP (HyperText Transfer Protocol) to be layered on top of it transparently. Furthermore, SSL is optimized for HTTP and is able to negotiate encryption keys as well as authenticate the server before the web browser exchanges data. The SSL protocol maintains the security and integrity of the transmission channel by using encryption, authentication and message authentication codes. In our case, SSL is selected because it provides an encrypted channel of

communication with no requirements at the client side. The only client requirement is to have an SSL enabled web browser. The majority of existing web browsers, and all major web browsers such as Internet Explorer™, Netscape Communicated™, and Mozilla™, have built-in SSL support.

5.3.2. Challenge Information

The "challenge information" feature is instantiated to become "user name and password". The reason for our selection is to avoid client setup complications such as acquiring a client certificate or a smart card. A user name and a password with a strong password policy will provide the required security presence by this feature.

5.3.3. Consecutive Failures

The number of allowed consecutive failures before locking a client user account is instantiated to 10 (ten) consecutive attempts. The reason for this selection is to have a number that legitimate users will not reach quickly in the case of normal mistakes such as having the CAPS LOCK key on, and that malicious users cannot make use of to perform dictionary attacks.

5.3.4. Unlocking an Account

In order to unlock a locked account, the e-commerce system will send an e-mail message to the system client user once his/her account is locked. This message will contain a URL with a randomly generated hashed string related to the user account. The system user will have to visit the provided URL either by clicking on it or by a copy/paste action. The URL will point to a script at the e-commerce system side. The system will then check if the provided link contains valid informaton for both the user account and the hashed string. If so, the account is unlocked and the user will be allowed to log into the system again. Otherwise, the system will keep the account locked.

The reason for having this type of unlocking process is because the client can do it at any time. Other unlocking processes, such as calling customer service, might contradict with system availability. An example is when a client user does not call within customer service hours. In this case, a legitimate user might be blocked from using the system until a customer service representative is available to unlock his account and, thus, the system availability requirement is violated.

One can argue that sending an e-mail message to unlock the account would rely on the security of the user mailbox. In other words, a malicious user might be able to unlock a locked account by breaking through the e-mail system security. In this case, the worst case is that the malicious user will be able to have another 10 (ten) attemptsbefore the system locks the account again and alarm the system administrator. Even in this worst-case scenario, the risk of having a malicious user breaking into the system is very low. By having a strong password policy, a lock/unlock feature, and an administrator alarm, breaching system authentication in this case is almost impossible.

5.3.5. Strong Cryptography

The "strong cryptography" feature is instantiated to use the RSA cryptosystem with 1024 bits keys. The reason for selecting RSA is because it is a standard for secure cryptography. RSA Laboratories (http://www.rsasecurity.com/) currently recommends key sizes of 1024 bits for corporate use. Several recent standards specify a 1024-bit minimum for corporate use. Less valuable information may well beencrypted using a 768-bit key, as such a key is still beyond the reach of all known key-breaking algorithms.

5.3.6. Strong Password Policy

The "strong password policy" feature is instantiated to obey the SANS standard [SANS] for strong password policies as follows:

- All system-level passwords (e.g., root, enable, NT admin, application administration accounts, etc.) must be changed on at least a quarterly basis.
- All user-level passwords (e.g., email, web, desktop computer, etc.) must be changed at least every six months. The recommended change interval is every four months.
- User accounts that have system-level privileges granted through group memberships or programs such as "sudo" must have a unique password from all other accounts held by that user.
- Passwords must not be inserted into email messages or other forms of electronic communication.
- All user-level and system-level passwords must:
    - Contain both upper and lower case characters (e.g., a-z, A-Z)
    - Have digits and punctuation characters as well as letters e.g., 0-9, !@#$%^&*()_+|~-=\`{}[]:";'<>?,./)
    - Are at least eight alphanumeric characters long.
    - Are not a word in any language, slang, dialect, jargon, etc.
    - Are not based on personal information, names of family, etc.
    - Passwords should never be written down or stored on-line. Try to create passwords that can be easily remembered. One way to do this is create a password based on a song title, affirmation, or other phrase. For example, the phrase might be: "This May Be One Way To Remember" and the password could be: "TmB1w2R!" or "Tmb1W>r~" or some other variation.

5.3.7. Authentication Summary

Based on the instantiation discussion from section 5.3.1 through 5.3.6, the derived authentication countermeasures design model from section 4.2 is instantiated to become as shown in Figure 5.2.

The instantiated model shown in Figure 5.2:

- Can be directly incorporated into the high-level design document of our case study e-commerce system.
- Contains effective countermeasures against the set of all known security attacks related to authentication
- Provides chronological order of the authentication process. Thus, converting the model into a flow chart for implementation purposes is straightforward and can be easily done.
- Provides high-level implementation guidelines to avoid security pitfalls.

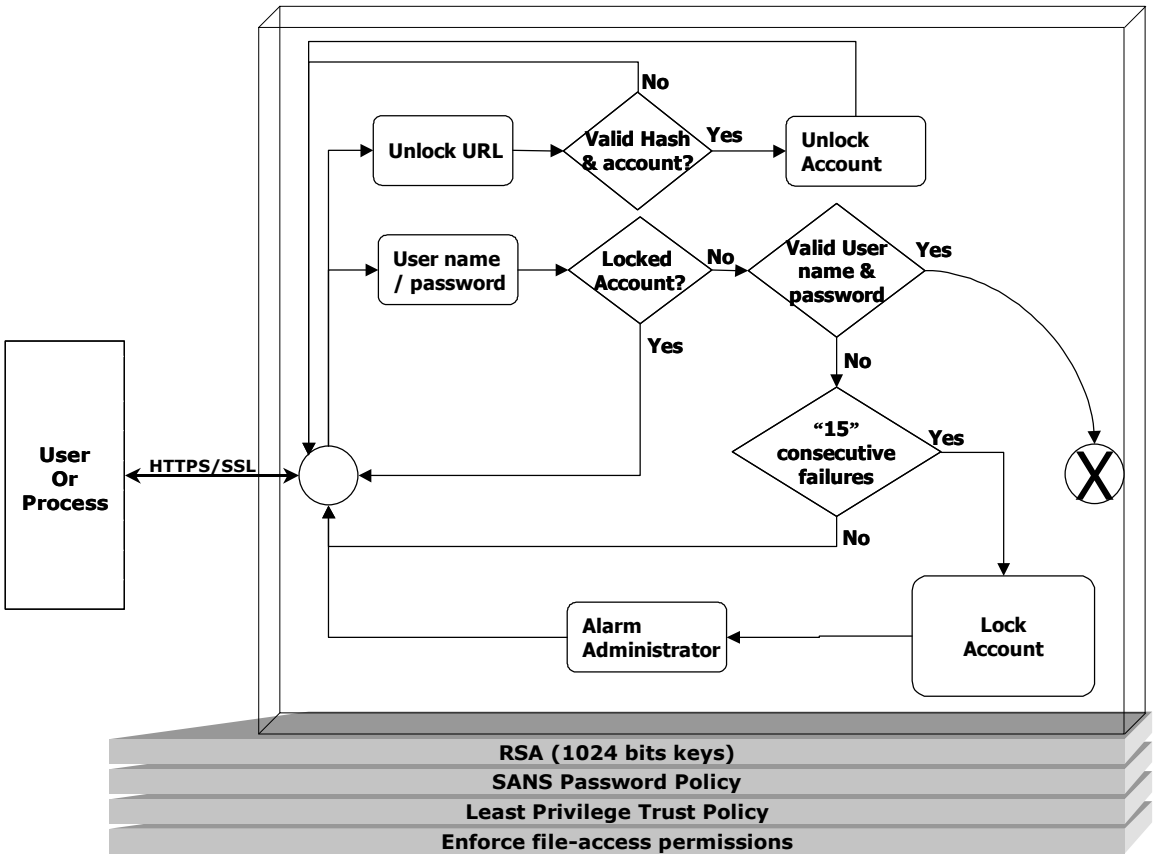

Figure 5.2. The SET-integrated system authentication countermeasures design model.

5.4. Applying and Integrating Authorization Design Model

In section 4.3, we applied our methodology to the authorization security feature of the NIST security services model. The result was a countermeasures design model that is effective against the set of all known security attacks related to authorization. In section 5.2, we described our case study consisting of securing a SET-integrated e-commerce system. In this section, we will apply the derived authorization countermeasures design model derived in section 4.3 to the e-commerce system described in section 5.2. This application is intended to prove soundness of the derived authorization countermeasures design model.

The first step in applying the derived countermeasures design model is to instantiate its features. This is a straightforward process that takes every feature of the model and converts it into an implementable feature.

A description of how the authorization model features described in section 4.3 are instantiated is provided below.

5.4.1. Explicit Authorization

The "explicit authorization" feature is instantiated to become a system command to explicitly validate the presented authorization information. This system command must be executed at the beginning of each and every script that our e-commerce implementation will be using. This is in parallel with our discussion of authorization bypassing attacks in section 4.3.2.

Because the system command is to be used in each and every script of the e-commerce system, this command must be easy to use and, at the same time, easy to update for future system releases. Thus, the preferred implementation guideline for this command is to have a black-box system module responsible for performing the authorization task This

module will have a fixed interface that defines how to start the authorization process and how to interpret the results.

5.4.2. URL Session Enforcement

Our discussion in section 4.3.2 showed that the "URL session enforcement" feature is required to prevent session hijacking attacks from occurring. Thus, this feature is instantiated as follows:

- Our e-commerce system must re-authenticate the user before critical actions are performed (i.e. a purchase, money transfer, etc.)
- Our system must limit the unique session tokens to the user browser instance. An example is to generate the session token by a one-way hash function of the client/administrator computer address and the process id of the browser being used.

The above-mentioned two system requirements will guarantee that a session hijacking attack will fail even when a malicious user succeeds at capturing the URL being used by the client/administrator. This is because it is very hard to extract the browser process id from the one-way hashed URL.

Furthermore, even if a malicious user succeeds in extracting the process id of the web browser, our e-commerce system will re-authenticate the client before critical actions. Thus, our URL session enforcement implementation is concluded to be robust.

5.4.3. Cookie Session Enforcement

The "cookie session enforcement" feature is instantiated in a similar way to URL session enforcement. Furthermore, our discussion in section 4.3.2 required the cookie to be volatile. Volatile cookies are not saved on local user hard drives. They are saved in the system memory, and once the web browser session ends, i.e. the web browser is closed, the cookie is erased from memory and can not be retrieved.

In addition to using volatile cookies, our e-commerce system must re-authenticate the user before critical actions are performed and session tokens must be limited to the user browser instance.

This case is very similar to the case of URL session enforcement discussed above. Thus, our cookie session enforcement instantiation is concluded to be robust.

5.4.4. Strong Session Management

The "strong session management" is instantiated to become a system module responsible for retrieving, validating, and updating session information being used. This command will be used by the authorization module described in section 5.4.1. The objective of having this module is to provide the following tasks:

- Retrieve session information from URLs and cookies depending on the session type being used by the system
- Validate session information such as browserinstance, MAC address, client account, etc.
- Implement a session time-out checking algorithm responsible for rejecting sessions that have been inactive for a certain period of time (defined by the system administrator)
- Implement session enforcement (for both URLs and cookies)

5.4.5. Strong Session Encryption

The "strong session encryption" feature is instantiated to use triple-DES. The reason for selecting triple-DES is because it is much faster than the RSA algorithm that we use for authentication. (http://rsasecurity.com/) Triple-DES is, as well, the FIPS approved symmetric encryption algorithm of choice. [FIPS 463]

Encryption, in session management, is required to protect the session information in the URL or cookie being used. Thus, cryptography is used only by the e-commerce system in this case. Exchanging the key, used to encrypt/decrypt the session information, with the client is not required. Furthermore, the session information becomes obsolete once the session ends when the user logs out of the system or when the session times out.

Thus, triple-DES is better in this case since it provides better performance with a sufficient amount of security for the encrypted session information.

5.4.6. Authorization Summary

Based on the instantiation discussion from section 5.4.1 through section 5.4.5, the derived authorization countermeasures design model from section 4.3 is instantiated to become as shown in Figure 5.3.

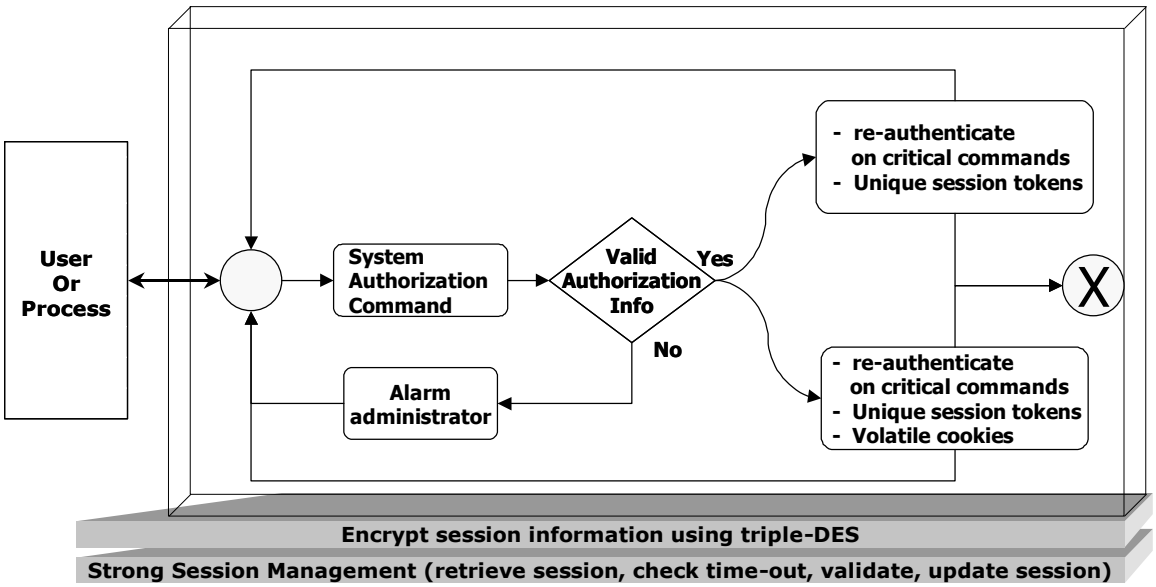

Figure 5.3. The SET-integrated system authorization countermeasures design model.

The instantiated model shown in Figure 5.3:

- Can be directly incorporated into the high-level design document of our case study e-commerce system.
- Contains effective countermeasures against the set of all known security attacks related to authorization
- Provides chronological order of the authorization process. Thus, converting the model into a flow chart for implementation purposes is straightforward and can be easily done.
- Provides implementation guidelines to avoid security pitfalls for the different types of security attacks related to authorization and session management.

5.5. Applying and Integrating Access Control Enforcement Design Model

In section 4.4, we applied our methodology to the access control enforcement security feature of the NIST security services model. The result was a countermeasures design model that is effective against the set of all known security attacks related to access control enforcement. In section 5.2, we described our case study consisting of securing a SET-integrated e-commerce system. In this section, we will apply the derived access control enforcement countermeasures design model derived in section 4.4 to the e-commerce system described in section 5.2. This application is intended to prove the applicability of the derived access control enforcement countermeasures design model.

The first step in applying the derived countermeasures design model is to instantiate its features. This is a straightforward process that takes every feature of the model and converts it into an implementable feature.

A description of how the access control enforcement model features, described in section 4.4, are instantiated is provided below.

5.5.1. Zero-Tolerance Trust Model

Our "zero-tolerance trust model" feature is instantiated to become a role-based access control model. ([Ferraiolo, 2001] and [Bonatti, 2000]) In our discussion of access control enforcement in section 4.4, we showed that one of the most challenging problems in managing large networked systems is the complexity of security administration. Today, security administration is costly and error-prone because system administrators usually specify access control lists for each user on the system individually. Role based access control (RBAC) is a technology that is attracting increasing attention because of its potential for reducing the complexity and cost of security administration. Our case study,

the SET-integrated system, can make use of this technology for enforcing its access control policy.

Our main interest in RBAC is to be able to apply a “don't trust, verify” access control policy. [Kahn, 1999] In our case, this means that some application agent A might make an assertion to agent B. Agent B must verify the assertion before acting on it. If A lies, the false assertion cannot cause much damage. At worst, it causes B to waste resources checking on a false assertion.

Furthermore, NIST provides an implementation of an RBAC web server [NIST, 2002] that is helpful for our case study during the implementation phase.

5.5.2. Access Control Policy Enforcement

The “access control policy enforcement” feature is instantiated to using an automation tool called COPS for checking access permissions to our system implementation. [COPS, 1993] COPS is a collection of programs that each attempt to tackle a different problem area security. For example, some of the checks that COPS performs are:

- File, directory, and device permissions/modes
- Poor passwords.
- Content, format, and security of password and group files
- Scheduled system program files
- A CRC check against important binaries or key files to report any changes therein
- Anonymous ftp setup
- Unrestricted tftp, decode alias in sendmail, SUID uudecode problems, hidden shells inside inetd.conf, rexd running in inetd.conf.
- Dates of CERT advisories vs. key files. This checks the dates that various bugs and security holes were reported by CERT against the actual date on the file in question.

5.5.3. Code-Coverage Testing

Code coverage testing is a countermeasure that can be performed once the system implementation is done. Yet, planning for this type of testing must be done while the system is still being implemented. Furthermore, test cases required to perform this type of testing are directly related to the system design.

Each project must choose a minimum percent coverage for release criteria based on available testing resources and the importance of preventing post-release failures. Clearly, safety-critical software, such as our case study, should have a high goal. We might set a higher coverage goal for unit testing than for system testing since a failure in lower-level code may affect multiple high-level callers. [CORNETT, 2002]

In our case study, the "code coverage testing" feature is instantiated to attain 80%-90% code coverage with technical reviews discussing uncovered code before system release. One might argue that setting any goal less than 100% coverage does not assure quality. Yet, our main interest is to avoid backdoors in our system. Code coverage results of 80%-90%, accompanied with formal technical reviews discussing uncovered code, are enough, in our opinion, to ensure that no backdoors exist in our system.

5.5.4. Source-Code Analysis

The "source code analysis" feature can be instantiated depending on the programming language used to develop our case study. The purpose of this countermeasure is to avoid buffer overflows in our system implementation.

If the Java™ programming language is to be used to develop the system, buffer overflow errors are impossible. The Java™ language simply does not provide any way to store data into memory that has not been properly allocated.

If the C/C++ language is to be used to develop the system, then our "source code analysis" feature can be instantiated to use BOON (Buffer Overrun detection). [Wagner,

2000] BOON is a tool for automatically finding buffer overrun vulnerabilities in C source code and can be downloaded from http://www.cs.berkeley.edu/~daw/boon/releases.html.

For the purpose of our case study, Java™ was used to develop the e-commerce system. Thus, no source-code analysis tool is required. Yet, BOON might be required to test any third-party source code involved with SET-integrated modules.

5.5.5. Access Control Enforcement Summary

Based on the instantiation discussion from section 5.5.1 through section 5.5.4, the derived access control enforcement countermeasures design model from section 4.4 is instantiated to become as shown in Figure 5.4.

The instantiated model shown in Figure 5.4:

- Can be directly incorporated into the high-level design document of our case study e-commerce system.
- Contains effective countermeasures against the set of all known security attacks related to access control enforcement
- Provides implementation guidelines to avoid security pitfalls for the different types of security attacks related to access control enforcement.

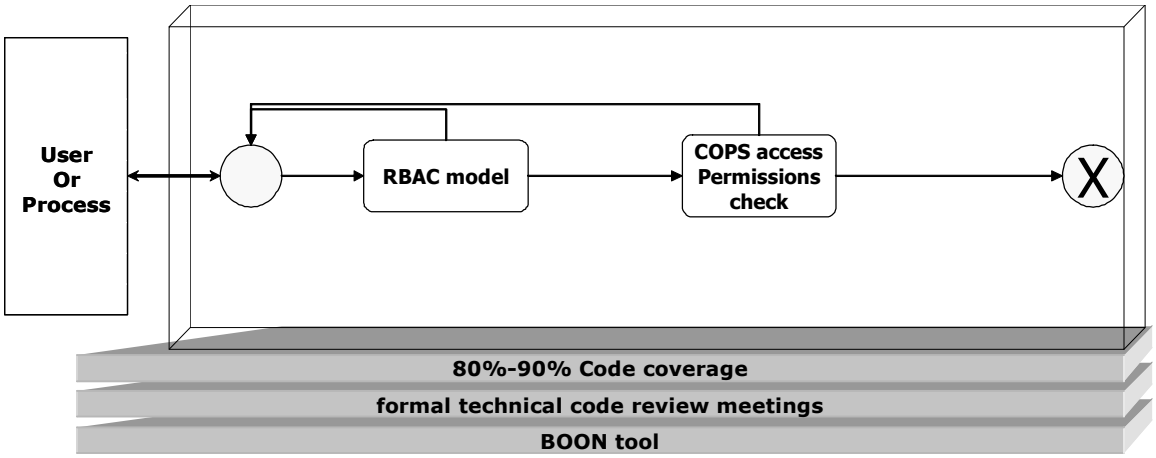

Figure 5.4. The SET-integrated system access control enforcement countermeasures design model.

5.6. Applying and Integrating Transaction Privacy Design Model

In section 4.5, we applied our methodology to the transaction privacy security feature of the NIST security services model. The result was a countermeasures design model that is effective against the set of all known security attacks related to transaction privacy. In section 5.2, we described our case study consisting of securing a SET-integrated e-commerce system. In this section, we will apply the derived transaction privacy countermeasures design model derived in section 4.5 to the e-commerce system described in section 5.2. This application is intended to prove the applicability of the derived transaction privacy countermeasures design model.

The first step in applying the derived countermeasures design model is to instantiate its features. This is a straightforward process that takes every feature of the model and converts it into an implementable feature. A description of how the transaction privacy model features, described in section 4.5, are instantiated is provided below.

5.6.1. Encrypted Channel / Transaction Information

The “encrypted channel / transaction information” feature is instantiated to use SET encryption. [SET, 1997-1] In our overview and discussion of SET in section 5.1, we described the role of SET in securing the transaction information while the transaction is in progress. SET uses symmetric encryption, Data Encryption Standard (DES), as well as asymmetric, or public-key, encryption to transmit session keys for DES transactions. Although this has disturbing connotations for a "secure" electronic transaction protocol– because public key cryptography is only used to encrypt DES keys and for authentication, and not for the main body of the transaction, the computational cost of asymmetric encryption is cited as reason for using weak 56 bit DES. Other reasons such as export/import restrictions, and the perceived need by law enforcement and government

agencies to access the plain-text of encrypted SET messages may also play a role. [IBM, 1998]

For the purpose of instantiating "encryption", our case study SET-integrated system makes use of SET modules. These modules are responsible for encrypting transaction information while the transaction is in progress. Furthermore, any modification to these modules at the encryption level might lead, in most cases, to incompatibility problems with other SET-enabled parties. Thus, the only instantiation possible in this case is to use SET encryption as specified in [SET, 1997-1]

5.6.2. Saving Sensitive Data in DBMS

While discussing SET in section 5.1, we described specific cases where SET reveals credit card information to merchants. Our typical system scenarios in section 5.2.5 also showed that order capture tokens and receipts are involved in a SET transaction. This type of information is considered sensitive as it contradicts with two security objectives: privacy and confidentiality.

On the other hand, a merchant system requires saving this type of sensitive information for future use. Thus, protecting this information is of high importance to ensure system security.

After identifying the sensitive information involved in a transaction, our security feature of using the DBMS for saving sensitive transaction information is instantiated to become "using the DBMS for saving credit card information, order capture tokens, and receipts".

5.6.3. Not Logging Sensitive Information

The "don't log sensitive information" feature is kept as is. This feature will serve as a guideline for the system implementation and will be performed during the system testing phase. The fact that this feature is built into the system security design model better is by

itself an enhancement to system security. This will help avoid discovering defects during the system testing phase that are related to the system design model.
The only part to be instantiated is the term "sensitive information." Based on the discussion in section 5.6.2, this feature is instantiated to become "don't log credit card information, order capture tokens, and receipts."

5.6.4. Enforce DBMS Security

The "enforce DBMS security" feature is instantiated to become the process of keeping the DBMS up-to-date with security fixes and patches. This, of course, requires the prior selection of a stable DBMS having a clean history with security issues. While it is not our direct concern to secure the selected DBMS, we must make sure a secure one is selected in the system specification. This was the main purpose of having this security feature in the countermeasures design model in the first place.

5.6.5. Log File Access Enforcement

The "log file access enforcement" feature is instantiated to become using COPS to ensure proper log file access permissions.[COPS, 1993] This step must be done after the system is implemented and installed. Yet, the reason for having it as a feature in the transaction privacy countermeasures design model is to avoid having defects related to the system specification during the system testing phase. By including this feature in the model, the system specification will be aware of including it as a guideline for system implementation and testing.

The reason for selecting COPS as the tool to automate this step is because we already use COPS to check file access permissions in the instantiated access control enforcement model.

5.6.6. Transaction Privacy Summary

Based on the instantiation discussion from section 5.6.1 through section 5.6.5, the derived transaction privacy countermeasures design model from section 4.5 is instantiated to become as shown in Figure 5.5.

The instantiated model shown in Figure 5.5:

- Can be directly incorporated into the high-level design document of our case study SET-integrated e-commerce system.
- Contains effective countermeasures against the set of all known security attacks related to transaction privacy
- Provides implementation guidelines to avoid security pitfalls for the different types of security attacks related to transaction privacy.

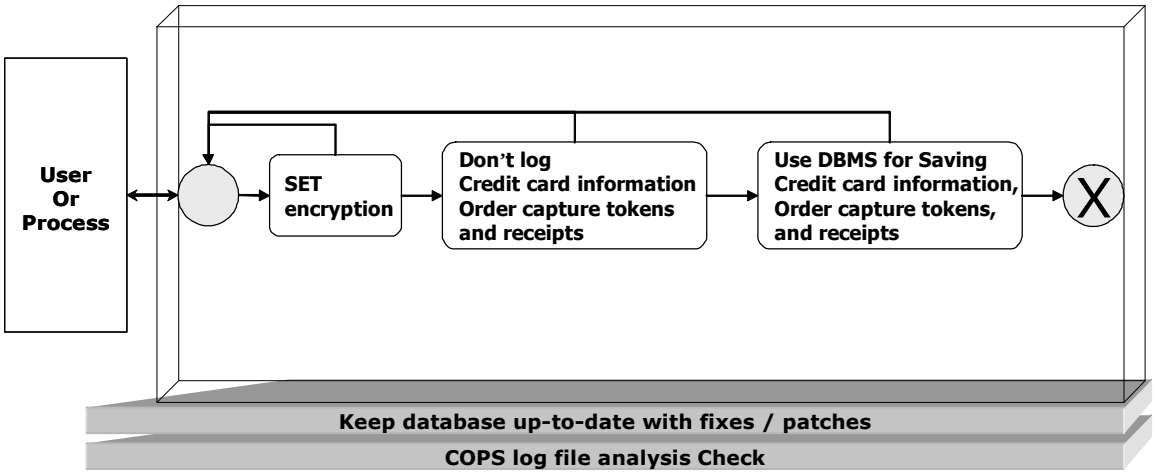

Figure 5.5. The SET-integrated system transaction privacy countermeasures design model.

5.7. Case Study System Security Comparison

In this section, we will compare the state of security in our SET-integrated e-commerce system before applying our countermeasures design models, i.e. while relying on SET security alone, and after applying the models.

Figure 5.6 shows a comparison of our SET-integrated e-commerce system security before and after applying the countermeasures design models. This comparison is based on the effectiveness of blocking security attacks related to the system.

Security feature	Security attack	Before	After
Authentication	Sniffing attacks	Partial	Complete
	ID spoofing attacks	Partial	Complete
	Brute-force attacks	No	Complete
	Dictionary attacks	No	Complete
	Replay attacks	Partial	Complete
	Credential decryption attacks	Partial	Complete
	Side-channel attacks	N/A	N/A
Authorization	ID spoofing attacks	Partial	Complete
	Authorization bypassing attacks	No	Complete
	Privilege brute-force attacks	Partial	Complete
	Replay attacks	Partial	Complete
	Session hijacking attacks	Partial	Complete
Access control enforcement	Backdoors	No	Complete
	Denial of service attacks	N/A	N/A
	Component failure attacks	No	Complete
	Buffer overflow attacks	No	Complete
Transaction privacy	Sniffing attacks	Partial	Complete
	Log data mining attacks	No	Complete
	DBMS exploits	No	Complete

Figure 5.6. A comparison of our SET-integrated e-commerce system security before and after applying the countermeasures design models.

Partial coverage means that an attack is partially blocked. An example of this is an attack that is blocked while a transaction is taking place but now before starting the transaction and after finishing it.

SET security provides partial protection against sniffing attacks, ID spoofing attacks, replay attacks, credential decryption attacks, privilege brute force attacks, and session hijacking attacks while the transaction is taking place. No security is provided for transaction information while in storage.

Furthermore, SET provides no security against brute-force attacks, dictionary attacks, authorization bypassing attacks, backdoor, component failure attacks, buffer overflow attacks, log data mining attacks, and DBMS exploits. These attacks appear due to the SET conflicts that we discussed earlier in this chapter.

Whether partial security or no security is provided, SET provides no security at all for the e-commerce system itself and is only concerned with transactions.

On the other hand, our countermeasures design models for authentication, authorization, access control enforcement, and transaction privacy; provide complete security against the set of all known security attacks related to e-commerce systems.

Security, in this case, is provided for the e-commerce system overall and is not limited to the transaction process itself. Our models ensure a safe e-commerce system before a transaction starts, while the transaction takes place, and after a transaction is done. This is very important in maintaining a secure e-commerce system and is directly related to the fact that the safety of a secure system is that of its weakest component. We clearly show here (Figure 5.6), that our methodology (Phase 1 plus Phase 2) has improved the security of a SET-integrated e-commerce design.

5.8. Case Study Evaluation

In this chapter, we have applied our four countermeasures design models for authentication, authorization, access control enforcement, and transaction privacy; to a SET-integrated e-commerce system. We also provided a comparison of the security of our case study system before and after applying our security-oriented design models for authentication, authorization, access control enforcement, and transaction privacy. In this section, we evaluate our experience of applying our security-oriented design models giving benefits and limitations.

5.8.1. Benefits

In this section, we discuss four benefits of applying our security-oriented design models.

The first benefit is that our security-oriented models *can be directly instantiated and applied for e-commerce systems*. This was shown throughout this case study as a straightforward process of instantiating each model feature into an existing technology.

The second benefit is that our models are *helpful for providing implementation guidelines and planning security testing*. An example of providing implementation guidelines is instantiating the *strong password policy* countermeasure to the SANS standard for strong passwords. This provided guidelines forimplementing strong passwords into the e-commerce system by specifying how to identify a strong password.

The third benefit is that our security-oriented models *minimize relying on the expertise of security architects for designing secure e-commerce systems*. This benefit is inherited from our systematic methodology and is also achieved in this case study. The process of instantiating our security-oriented models only requires research for existing technologies and does not require strong security expertise.

The fourth benefit is that our models *provide complete protection against all known security attacks in the e-commerce domain*. This benefit is also inherited from our systematic methodology and applies in this case study as well. Section 5.7 provided a comparison between the state of security in our SET-integrated e-commerce system before and applying our design models for authentication, authorization, access control enforcement, and transaction privacy.

5.8.2. Limitations

In this section we discuss one limitation of our security-oriented design models based on our experience while applying them to our SET-integrated e-commerce system.

As seen in section 5.7, the security range of our design models is limited to the e-commerce system itself. Third party components (such as network components, operating system platforms, etc.) are not taken into consideration. In practice, of course, these components must be taken into consideration.

In our thesis, this limitation was introduced because, in most cases, EC system designers limit their security range to that of the e-commerce system that they provide. This is logical because the security of any component relies on the security of its designer. Furthermore, it is almost impossible to guarantee security of third-party components that are used in the e-commerce system since these components provide the required functionality in a "black-box" manner.

Chapter VI

Methodology Evaluation

In Chapter 4, we provided a case study for Phase 1 of the methodology to derive four countermeasures design models for authentication, authorization, access control enforcement, and transaction privacy. This case study enabled an evaluation of the applicability and systematicity of our methodology.

In Chapter 5, we instantiated and integrated the derived countermeasures design models from Chapter 4 into a SET-integrated e-commerce system. This case study (Phase 2) enabled an evaluation of the effectiveness of the methodology and the derived countermeasures design models.

In this chapter, we will provide an evaluation of the methodology based on the experience gained while applying the two case studies in Chapters 4 and 5. We will discuss benefits and limitations in sections 6.1 and 6.2 respectively.

6.1. Benefits

In this section, we will discuss the benefits of our methodology. In section 6.1.1, we discuss the benefits of Phase 1 of our methodology for deriving security countermeasures design models. In section 6.1.2, we discuss the benefits of Phase 2 of our methodology

for instantiating and integrating our methodology into e-commerce systems design.

6.1.1. Benefits of Phase 1 of the Methodology

The benefits of Phase 1 of our methodology for deriving security-oriented design models are discussed below with respect to the set of Solution Requirements specified in Section 2.4. Other benefits are also listed.

Our methodology is *systematic and modular in nature* (Requirement 1) since every step can be represented as a module or process that requires a set of input arguments, performs a certain task, and produces a set of output arguments. The only relation between the different methodology modules or processes is the methodology itself. The modularity and systematicity of our methodology enable the distribution of tasks while applying the methodology steps. This also minimizes relying on the expertise of security architects to design security into the target system.

Our methodology *satisfies standard user requirements at system design time* (Requirement 2). This is inherited from the fact that all the derived countermeasures design models are based on a standard security model: the NIST security services model. In other words, our methodology builds on a security standard, the NIST security services model, which relates and satisfies legitimate user requirements. Our methodology then specializes these requirements by introducing effective countermeasures to block malicious user requirements. This leads to another advantage of Phase 1 of our methodology is discussed next.

Another advantage for our methodology is that it *specializes the security features* of the NIST security services model to be used in e-commerce systems *without breaking any interdependent relationships among these features* (Requirement 3). This advantage is achieved because our methodology extends / specializes the NIST security features only by introducing countermeasures. We do not introduce new relationships among security features. For example, the NIST security services model specifies that authorization is

related to access control enforcement (see Figure 2.1). Our methodology specialized these two security features for EC systems by introducing effective countermeasures against all known security attacks in the e-commerce domain and the relationship between authorization and access control enforcement was preserved.

Our methodology *effectively addresses and blocks all known malicious user requirements* (consequently all known security attacks) at system design time. This was shown throughout Chapter 4 via the validation traceability matrix for the four security-oriented design models that were derived; namely authentication, authorization, access control enforcement, and transaction privacy. The validation traceability matrix verified that all malicious user requirements are addressed via proper countermeasures (no empty columns) and that all countermeasures are required (no empty rows).

Another advantage for our methodology is that *if a security attack cannot be blocked at system design time* for any reason such as requiring system implementation; our methodology will *provide implementation and testing guidelines.* This increases the awareness of developers and testers about the security attack and minimizes the probability of the attack being successful during the system-testing phase. The worst case is when an attack succeeds during the testing phase. However, this will be considered an implementation-related defect that is relatively cheaper to fix than a design-related defect.

Our methodology has *better chances of achieving overall security in e-commerce systems* compared to attack trees and fault tree analysis techniques (see Chapter 2). As discussed earlier, fault tree analysis techniques are not directly related to designing security. Attack trees, on the other hand, design security on a per-attack basis. This might not lead to an overall design for security since there is no standard relating all known security attacks to e-commerce system features. In other words, attack trees explicitly rely on security architects to identify and related security attacks to security features. Our methodology designs security on a per-security-feature basis. This approach has better chances of achieving overall system security since it can be based on security standards such as the NIST security services model.

6.1.2. Benefits of Phase 2 of the Methodology

The benefits of Phase 2 of our methodology for instantiating and integrating our methodology into e-commerce systems design are discussed below.

Our derived *countermeasures design models are extensible* (Requirement 4) because these design models are composed of an action box and underlying planes. If a future attack is discovered, then updating the security-oriented design model would only require updating the action box and underlying planes to include any required countermeasures.

A countermeasures design model that is derived via our methodology can be easily instantiated and integrated into high-level design documents of e-commerce systems. This advantage was verified in Chapter 5. The four countermeasures design models derived in Chapter 4 are divided into a box and underlying planes. The box contains a flowchart of countermeasures while the underlying planes contain countermeasures that require no action to be taken. The ease of instantiating and integrating these countermeasures (flowcharts and planes) into high-level design documents was verified in Chapter 5.

An advantage of our methodology is that it *can be used with security-aware technologies* such as SET™ and CORBA™. Our case study in Chapter 5 verified this advantage by applying the four countermeasures design models for authentication, authorization, access control enforcement, and transaction privacy to a SET-integrated e-commerce system. This case study also verified that the relation between designing for security and implementing security is minimal. This is because Phase 1 provides high-level security requirements for designing secure e-commerce systems. These requirements are instantiated in Phase 2. This allows system designers to instantiate the security models while taking into consideration any implementation constraints the e-commerce system might impose. This leads to another advantage of Phase 2 of our methodology and is discussed next.

Another advantage of our methodology is that once a countermeasures design model is derived for a particular security feature (Phase 1), *the same model from Phase 1 can be instantiated and integrated into a variety of e-commerce system designs* (Phase 2) *without needing to repeat Phase 1*. For example, if two e-commerce systems are designing authentication security with one system using requiring SET in order to process system transactions [SET, 1997-1] and the other system requiring the Banking Internet Payment System (BIPS) in order to process payments [BIPS, 2001], then the same authentication countermeasures design model, for example, derived in Phase 1 of this thesis can be instantiated and integrated into the two e-commerce systems without the need to repeat Phase1 for each system. In other words, shopping online will utilize the same countermeasures design model regardless of the payment system chosen for implementation.

6.2. Limitations

The limitations of our methodology are discussed below. Section 6.2.1 discusses the limitation of Phase 1 of our methodology. Section 6.2.2 discusses the limitations of Phase 2 of our methodology.

6.2.1. Limitations of Phase 1 of our Methodology

The limitations of Phase 1 of our methodology for deriving security-oriented countermeasures design models are discussed below.

The first limitation is that our methodology does not provide countermeasures against future attacks. This is a normal constraint because we cannot provide countermeasures against attacks that cannot be analyzed.

Another limitation is that our methodology is not systematic when analyzing residual vulnerabilities of prescribed countermeasures. A prescribed countermeasure must be

analyzed and discussed in order to discover any potential remaining residual vulnerability.

Prescribed countermeasures might impact performance if these countermeasures do not take performance into consideration. In other words, our methodology warns for possible performance impact but does not go into depth to analyze and model how such impact might occur and how to overcome it.

6.2.2. Limitations of Phase 2 of our Methodology

The limitations of Phase 2 of our methodology for instantiating and integrating our methodology into e-commerce systems design are discussed below.
The first limitation of Phase 2 is that the instantiation process is not systematic in nature. This step still requires some security expertise in order to be done. This is a normal limitation since the process of instantiating a certain countermeasure has to take into consideration all limitations that the target e-commerce system might impose. Different e-commerce systems might impose different constraints and, thus, it is impossible to model all possible system constraints that e-commerce systems might impose. However, Phase 1 explicitly states the attack enablers that the countermeasures are required to block. Thus, the process of instantiating the countermeasures can be described as a straight forward process that requires minimal expertise in the security domain.

Another limitation of Phase 2 of our methodology is that it does not explicitly state and model the relationship between the various countermeasures introduced into the e-commerce systems. Of course, if two proper countermeasures from two countermeasures design models were instantiated improperly, the result could be a dangerous residual vulnerability.

6.3. Summary

Most of the questions that imposed limitations on our methodology were raised during our research. We were not able to solve these questions because of constraints on the scope of the work. However, we believe we have established a foundation for future research. For more information on limitations to the design for security, please see[NRC, 1991].

Chapter VII

Thesis Summary

7.1. Conclusion

This thesis described a new methodology for deriving countermeasures design models for e-commerce systems. The methodology is based on the NIST security services model. Our approach focuses on satisfying legitimate user requirements while blocking malicious user requirements at system design time.

We assessed and showed that our methodology is systematic through a case study that derived four countermeasures design models for authentication, authorization,access control enforcement, and transaction privacy.

The derived countermeasures design models were assessed through a realistic case study on a SET-integrated in e-commerce systems. These models were also proven to be effective against all security attacks related to the e-commerce domain.

We see the primary benefits of our research as follows:

- A comprehensive matrix listing and mapping all known security attacks to four security features in e-commerce systems; namely authentication, authorization, access control enforcement, and transaction privacy.

- Four new security models that extend the NIST security services model for e-commerce systems. These models are proven to be effective against all known security attacks related to e-commerce systems.
- A faithful implementation of a countermeasures design model was proven to be guaranteed to block all known security attacks related to that feature.
- Security architects can avoid expensive system development life cycles fixes. This is achieved by having an effective countermeasures design model that is directly applicable to EC systems and that specifies detailed requirements for the security feature.
- A cost-effective, systematic methodology for deriving countermeasures design models for the other security features of e-commerce systems.
- An overview of all known security attacks related to the four security features discussed in this thesis; namely authentication, authorization, access control enforcement, and transaction privacy.

7.2. Future Directions

Little research has been done on methodologies and approaches for designing secure e-commerce systems. Therefore, many research opportunities are still available.

Further research is needed to optimize our methodology and countermeasures deisgn models. In particular, further studies should:

- Apply our methodology and approach to the remaining features of the NIST security services model.

- Further enhance the methodology to map other security-related features, such as impact on performance, into the design process.

- Formally describe the methodology and provide automation.

- Apply the methodology to other standard security models once available.

- Update the derived countermeasures design models with new security attacks once available.

Most of these questions were raised during our research. We were not able to answer these questions because of necessary constraints on the scope of the work. However, we believe we have established a foundation for such future research.

APPENDIX A: Glossary of Terms

Term	Definition
Access Control	Enable authorized use of a resource while preventing unauthorized use or use in an unauthorized manner. [NIST, 2001]
Accountability	The requirement that actions of an entity may be traced uniquely to that entity. [NIST, 2001]
Assets	Information or resources to be protected by the countermeasures of a system. [CC, 1999]
Assurance	The basis for confidence that the security measures, both technical and operational, work as intended to protect the system and the information it processes. [NIST, 2001]
Attack Enabler	A feature, whether property or service, that enables a security attack to succeed.
Authentication	Verifying the identity of a user, process, or device, often as a prerequisite to allowing access to resources in a system. [NIST, 2001]
Authentication data	Information used to verify the claimed identity of a user. [CC, 1999]
Authorization	The granting or denying of access rights to a user, program, or process. [NIST, 2001]
Authorization bypassing attack	An attack that attempts to bypass authorization by exploiting a weak authorization implementation process.
Authorized user	A user who may, in accordance with the security policy, perform an operation. [CC, 1999]
Availability	The security objective that generates the requirement for protection against intentional or accidental attempts to (1) perform unauthorized deletion of data or (2) otherwise cause a denial of service or data. [NIST, 2001]
Backdoor	An undocumented way of gaining access to an EC system.

Buffer overflow	An attack that allows a malicious user to input data that will cause either a program crash (creating a Denial of Service attack) or cause the system to run code granting the attacker further access.
Component failure attack	An explicit attempt on an EC system component to force failure while providing a service.
Confidentiality	The security objective that generates the requirement for protection from intentional or accidental attempts to perform unauthorized data reads. Confidentiality covers data in storage, during processing, and while in transit.. [NIST, 2001]
CORBA	CORBA is a vendor-independent architecture and infrastructure that computer applications use to work together over networks. Using the standard protocol IIOP, a CORBA-based program from any vendor, on almost any computer, operating system, programming language, and network, can interoperate with a CORBA-based program from the same or another vendor, on almost any other computer, operating system, programming language, and network. [OMG, 2000]
Countermeasure	Any action, device, procedure, technique, or other measure that reduces the vulnerability of or threat to a system. [GMU]
Credential decryption	The process of converting encrypted credential data back into its original form.
Credential information	Any form of identity information including combinations of user names and passwords, certificates, etc.
Credential policy	A policy for creation of strong credentials (such as passwords), the protection of those credentials, and the frequency of change. [SANS]
Data Integrity	The property that data has not been altered in an unauthorized manner while in storage, during processing, or while in transit [NIST, 2001]
DBMS exploits attack	An attempt to exploit DBMS security for the purpose of exploiting the security of an e-commerce system.

Denial of Service attack	An explicit attempt by attackers to prevent legitimate EC system users from using the system.
Dictionary attack	An attack directed towards finding passwords in a specific list, such as an English dictionary.
Digital Certificate	A message which ensures the identity of an entity by identifying a trusted registration authority, naming or identifying the entity, specifying the entity public key, and including a digital signature of the registration authority.
Differential Power Analysis attack	A side channel attack in which an attacker infers the secret information by using statistical analysis of power consumption. [Kocher]
Entity	Either a subject (an active element that operates on information or the system state) or an object (a passive element that contains or receives information). [NIST, 2001]
ID Spoofing attack	This attack occurs when a malicious user or process claims to be a different user or process.
Identity	Information that is unique within a security domain and which is recognized as denoting a particular entity within that domain. [NIST, 2001]
Log data mining attack	A search for significant patterns and trends in large log data files by using sophisticated statistical techniques and software in an attempt to exploit system privacy.
Man-in-the-middle attack	See sniffing attack.
Model	A simplification or abstraction of some area of concern, organized in a way to make it easy to understand and study. [CC, 1999]
Protection Profile	An implementation-independent set of security requirements for a category of TOEs that meet specific consumer needs. [CC, 1999]
Replay attack	An attack by recording and replaying previously sent valid messages (or parts of messages) [Herzog, 2001]

Residual vulnerability	A non-exploitable vulnerability that could be exploited by an attacker with greater attack potential than is anticipated in the intended environment for the TOE [CC, 1999]
Risk Analysis	The process of identifying the risks to system security and determining the probability of occurrence, the resulting impact, and the additional safeguards that mitigate this impact. Part of risk management and synonymous with risk assessment. [NIST, 2001]
Risk Management	The total process of identifying, controlling, and mitigating information technology related risks. It includes risk analysis; cost-benefit analysis; and the selection, implementation, test, and security evaluation of safeguards. This overall system security review considers both effectiveness and efficiency, including impact on the mission/business and constraints due to policy, regulations, and laws. [NIST, 2001]
Security	The quality or state of being protected from uncontrolled losses or effects. Security is a system property. Security is much more than a set of functions and mechanisms. Information technology security is a system characteristic as well as a set of mechanisms which span the system both logically and physically. [NIST, 2001]
Security attack	Actions performed by a threat agent to give rise to threats. [CC, 1999]
Security feature	A service or a supporting component of security. [NIST, 2001]
Security objective	Statement of intent to counter identified threats and/or satisfy identified organization security policies and assumptions. [CC, 1999]
Security policy	The statement of required protection of the system. [NIST, 2001]
Security presence	The defined environment for security [Herzog, 2001]
Security target	A set of security requirements and specifications to be used as the basis for evaluation of an identified TOE.
Session	A session begins when a user/process is authenticated successfully and is usually used, subsequently, to identify the user without the

	need for re-authentication. Sessions are useful for transforming the Internet (stateless by default) into a stateful medium.
Session management	The process of creating, maintaining, and ending a session in a secure manner.
Session hijacking attack	This type of attack involves an attacker using captured, brute forced, or reverse-engineered authorization information (such as session information) to seize control of a legitimate user's session while that user is logged into the EC system. [Herzog, 2001]
SET	An open technical standard for the commerce industry developed by Visa and MasterCard as a way to facilitate secure payment card transactions over the Internet. [SET, 1997-1]
Side channel attack	An attack in which an attacker infers stored secret information in a cryptographic device by using data other than input or output data. An example of such data is computation time or power consumption.
Simple Power Analysis attack	A side channel attack in which an attacker infers the secret information by using power consumption as leaked data. [Kocher]
Sniffing attack	Also known as the man-in-the-middle attacks, a sniffing attack is the digital analogue to phone tapping or eavesdropping. This attack captures information as it flows between a client and a server.
Spoofing attack	See ID Spoofing attack
System Integrity	The quality that a system has when performing the intended function in an unimpaired manner, free from unauthorized manipulation [NIST, 2001]
Target of Evaluation	An IT product or system and its associated administrator and user guidance documentation that is the subject of an evaluation. [CC, 1999]
Threat	The potential for a “threat source” to exploit (intentional) or trigger (accidental) a specific vulnerability. [NIST, 2001]

Threat-agent	An agent wishing to abuse and/or damage assets by giving rise to threats [CC, 1999]
Threat-source	Either (1) intent and method targeted at the intentional exploitation of a vulnerability or (2) the situation and method that may accidentally trigger a vulnerability. [NIST, 2001]
Timing attack	A side channel attack in which an attacker infers the secret information by using computation time as leaked data. [Kocher]
Trapdoor	See *backdoor*
Vulnerability	A weakness in system security procedures, design, implementation, internal controls, etc., that could be accidentally triggered or intentionally exploited and result in a violation of the system's security policy. [NIST, 2001]

APPENDIX B: Acronyms

Abbreviation	Description
AE	Attack Enabler
API	Application Program Interface
ASA	Abstract Security Attack
CC	Common Criteria Redbook
CGI	Common Gateway Interface
CM	Countermeasure
DBMS	Data Base Management System
DDoS	Distributed Denial of Service
DoS	Denial of Service
DPA	Differential Power Analysis
EAL	Evaluation Assurance Level
EAL	Evaluation Assurance Level
EC	Electronic Commerce
FIPS	Federal Information Processing Standard
FTP	File Transfer Protocol
HTML	HyperText Markup Language
HTTP	Hyper-Text Transfer Protocol
IP	Internet Protocol
IT	Information Technology
MIME	Multi-purpose Internet Mail Extension
MUR	Malicious User Requirements
NIST	National Institute for Standards and Technology
OSSTMM	Open-Source Security Testing Methodology Manual
PP	Protection Profile
SA	Security Attack
SET	Secure Electronic Transactions

SF	Security Feature
SMTP	Simple Mail Transfer Protocol
SPA	Simple Power Analysis
SQL	Structured Query Language
ST	Security Target
SUD	System Under Development
SUT	System Under Test
TCP	Transmission Control Protocol
TOE	Target of Evaluation
URL	Uniform Resource Locator

References

1. [Agnew, 2000] G. Agnew, "Cryptography, Data Security, and Applications to E-commerce", Electronic Commerce Technology Trends Challenges and Opportunities, Feb. 2000, IBM Press, pp. 69-85. ISBN: 1-58347-009-3
2. [Alturi, 2002] V. Atluri and A. Gal, "An Authorization Model for Temporal and Derived Data: Securing Information Portals", ACM Transactions on Information and Systems Security, Vol. 5, No. 1, Feb. 2002, pp. 62-94
3. [Anderson, 2001] R. Anderson, "Security Engineering: A Guide to Building Dependable Distributed Systems", John Wiley & Sons, 2001. ISBN: 0-471-38922-6
4. [Bellare, 1998] M. Bellare, R. Canetti, and H. Krawczyk, "A Modular Approach to the Design and Analysis of Authentication and Key Exchange Protocols", 1998, ACM 0-89791-962-9 98
5. [Bertino, 2000] E. Bertino, S. Castano, E. Ferrari, and M. Mesiti, "Specifying and Enforcing Access Control Policies for XML Document Sources", World Wide Web 3, Baltzer Science Publishers BV, 2000, pp. 139-151
6. [Bertino, 2001] E. Bertino, P. Bonatti, and E. Ferrari, "TRBAC: A Temporal Role-Based Access Control Model", ACM Transactions on Information and System Security, Vol. 4, No. 3, August 2001, pp. 191-223.
7. [BIPS, 2001] Financial Services Technology Consortium, "Bank Internet Payment System: Specification and Protocol", version 1.1, November 2001. website: http://www.fstc.org/projects/bips/spec/BIPS_spec_v1p1.pdf
8. [Bonatti, 2000] P. Bonatti, S. Vimercati, and P. Samarati, "A Modular Approach to Composing Access Control Policies", 2000, ACM 1-58113-203-4/00/0011
9. [Botha 2002] R. Botha and J. Eloff, "A framework for access control in workflow systems", Information Management & Computer Security, Vol. 9, No. 3, 2002, pp. 126-133, MCB University Press. ISSN: 0968-5227
10. [Bouchard, 2000] L. Bouchard, "Securing Transactions and Ensuring Non-Repudiation: The Experience of the Department of Justice in Quebec", Electronic Commerce Technology Trends Challenges and Opportunities, Feb. 2000, IBM Press, pp. 339-354. ISBN: 1-58347-009-3
11. [Brinkley, 1995-1] D. Brinkley and R. Schell, "Concepts and Terminology for Computer Security", Information Security: An Integrated Collection of Essays, pp. 40-97, IEEE Computer

Society Press, 1995. ISBN: 0-8186-3662-9

12. [Brinkley, 1995-2] D. Brinkley and R. Schell, "What Is There to Worry About? An Introduction to the Computer Security Problem", Information Security: An Integrated Collection of Essays, pp. 11-39, IEEE Computer Society Press, 1995. ISBN: 0-8186-3662-9
13. [Candolin, 2000] C. Candolin and H. Kari. "Time Privacy of Electronic Transactions". Proceedings of the Helsinki University of Technology Seminar on Network Security, 2000, website: http://www.tcm.hut.fi/Opinnot/Tik-110.501/2000/papers/candolin.pdf
14. [CC, 1999] Common Criteria (CC), "Common Criteria for Information Technology Security Evaluation", August 1999, website: http://www.commoncriteria.org/
15. [CERT, 1997] CERT Coordination Center, website: http://www.cert.org/
16. [Chang, 2002] K. Chang, B. Lee, and T. Kim, "Open Authentication Model Supporting Electronic Commerce in Distributed Computing", Electronic Commerce Research, Vol 2, pp. 135-149, 2002, Kluwer Academic Publishers
17. [Chuang, 1997] S. Chuang and P Wernick, "A Credibility-based Model of Computer System Security", ACM New Security Paradigm Workshop Lake Arrowhead CA, 1997, ACM 0-89791-878-9 96 09
18. [COPS, 1993] Farmer, D., "Computer Oracle and Password System", May 1993. Website: http://www.fish.com/cops/
19. [CORNETT, 2002] Cornette, S., "Code Coverage Analysis", Oct. 2002. Website: http://www.bullseye.com/webCoverage.html
20. [Dierks, 1999] T. Dierks and C. Allen, "The TLS Protocol: version 1.0", Network Working Group, Request for Comments: 2246, website: http://ietf.org/rfc/rfc2246.txt
21. [EUC] European Commission, "Information Society Statistics", website: http://europa.eu.int/information_society/topics/ebusiness/ecommerce/3information/statistics/index_en.htm
22. [Farrell, 2000] S. Farrell, et al., "AAA Authorization Requirements", Network Working Group, RFC 2906, August 2000, website: http://www.faqs.org/rfcs/rfc2906.html
23. [Fenelon, 1994] P. Fenelon , J. A. McDermid, M. Nicolson, and D. J. Pumfrey, "Towards Integrated Safety Analysis and Design", ACM Press, New York, NY, USA, pp. 21-32, 1994. website: http://doi.acm.org/10.1145/381766.381770
24. [Ferraiolo, 2001] D. Ferraiolo, R. Sandhu, S. Gavrila, D. Kuhn, and R. Chandramouli, "Proposed

NIST Strandard for Role-Based Access Control", ACM Transactions on Information and Systems Security, Vol. 4, No. 3, August 2001, pp. 224-274

25. [FIPS 463] "Data Encryption Standard (DES)", U.S. Department of Commerce/National Institute of Standards and Technology, October 1999. Website: http://csrc.nist.gov/publications/fips/fips46-3/fips46-3.pdf
26. [Ford, 1997] W. Ford and M. Baum, "Secure Electronic Commerce", Prentice Hall, 1997. ISBN 0-13-476342-4
27. [Freier, 1996] A. Freier, Karlton, P., and Kocher, P., "The SSL Protocol Version 3.0", Nov. 1996. website: http://wp.netscape.com/eng/ssl3/draft302.txt
28. [GMU] George Mason University Center for Secure Information Systems, "Security Glossary", website: http://www.ise.gmu.edu/~csis/glossary/
29. [Hawkins, 2000] S. Hawkins, D. Yen, and D. Chou, "Awareness and Challenges of Internet Security", Information Management & Computer Security, Vol. 8, No. 3, 2000, pp. 131-143, MCB University Press. ISSN: 0968-5227
30. [Herzog, 2001] P. Herzog, "The Open Source Security Testing Methodology Manual", version 1.5, May 2001, website: http://ideahamster.org/
31. [IBM, 1998] IBM Corporation, "An overview of the IBM SET and the IBM CommercePoint Products", http://www.software.ibm.com/commerce/set/overview.html, June 1998
32. [ISOSC 27] International Organization for Standardization, ISO Technical Programme, IT Security Techniques Sub-Committee 27. Website: http://www.iso.ch/iso/en/stdsdevelopment/techprog/workprog/TechnicalProgrammeSCDetailPage.TechnicalProgrammeSCDetail?COMMID=143
33. [Jha, 2002] S. Jha, O. Sheyner, and J. M. Wing, "Minimization and Reliability Analyses of Attack Graphs", February 2002. website: http://reports-archive.adm.cs.cmu.edu/anon/2002/CMU-CS-02-109.pdf
34. [Jonsson, 1998] E. Jonsson, "An Integrated Framework for Security and Dependability", Proceedings of the 1998 workshop on New security paradigms, pp. 22-29, ACM Press, 1998, ISBN: 1-58113-168-2
35. [Kahn, 1999] C. Kahn, "Tolerating Penetrations and Insider Attacks by Requiring Independent Corroboration", ACM Press, 1998, ISBN:1-58113-168-2
36. [Kocher] C. Kocher, "Cryptanalysis of Di? e-Hellman, RSA, DSS, and Other Systems Using Timing Attacks" Website: http://www.cryptography.com/

37. [Kohl, 1993] J. Kohl and C. Neuman, "The Kerberos Network Authentication Service (V5)", Network Working Group, RFC 1510, September 1993, ftp://ftp.isi.edu/in-notes/rfc1510.txt
38. [Liddy, 1998] C. Liddy and A. Sturgeon, "Seamless Secured Transactions", Information Management & Computer Security, Vol. 6, No. 1, 1998, pp. 21-27, MCB University Press. ISSN: 0968-5227
39. [Malkin, 1996] G. Malkin, "Internet Users' Glossary", Network Working Group, RFC 1983, August 1996, website: http://www.faqs.org/rfcs/rfc1983.html
40. [McDermott, 2000] J. McDermott, "Attack Net Penetration Testing. In the 2000 New Security Paradigms Workshop (Ballycotton, County Cork, Ireland), 2000, ACM SIGSAC, ACM Press, pp. 15-22.
41. [Nguyen, 2001] H. Nguyen, "Testing Applications on the Web", pp. 285-310, John Wiley & Sons, 2001. ISBN: 0-471-39470-X
42. [NIST, 2001] National Institute of Standards and Technology (NIST), Special Publication 800-33, "Underlying Technical Models for Information Technology Security", 2001. website: http://csrc.nist.gov/publications/nistpubs/800-33/sp800-33.pdf
43. [NIST, 2002] "Role Based Access Control", National Institute of Standards and Technology, August 2002. Website: http://csrc.nist.gov/rbac/
44. [NRC, 1991] National Research Council (NRC), "Computers at Risk: Safe Computing in the Information Age", pp. 7-73, National Academy Press, 1991. ISBN: 0-309-04388-3
45. [Okeya, 2002] K. Okeya and K. Sakurai, "On Insecurity of the Side Channel Attack Countermeasure Using Addition-Subtraction Chains under Distinguishability between Addition and Doubling ", June 2002, website: http://link.springer.de/link/service/series/0558/papers/2384/23840420.pdf
46. [OMG, 2000] OMG, "CORBA Services: Common Object Security Specification." Available at ftp://ftp.omg.org/pub/docs/security/99-12-02.pdf
47. [Philips, 1998] C. Phillips and L. P. Swiler, "A Graph-Based System for Network-Vulnerability Analysis", ACM Press, New York, NY, USA, pp. 71-79, 1998, ISBN: 1-58113-168-2. website: http://doi.acm.org/10.1145/310889.310919
48. [PMEL] Precision Measurement Equipment Laboratories (PMEL), "Electronic Commerce Glossary", website: http://www.pmel.org/EC-Glossary.htm
49. [RSA] R. Rivest, A. Shamir, and L. Adleman, "A Method for Obtaining Digital Signatures and

Public-Key Cryptosystems," Communications of the ACM, 21(2):120-126, February 1978

50. [SANS] "Password Policy", website: http://www.sans.org/newlook/resources/policies/Password_Policy.pdf
51. [Schneier, 1999] B. Schneier, "Attack Trees", Doctor Dobb's Journal, December 1999, pp 21-29. website: http://www.counterpane.com/attacktrees-ddj-ft.html
52. [Schneier, 2000] B. Schneier, "Secrets and Lies: Digital Security in a Networked World", John Wiley & Sons, 2000. ISBN: 0-471-25311-1
53. [SecTeam] SecuriTeam Security Tools Archive. Website: http://www.securiteam.com/tools/archive.html
54. [SET, 1997-1] SET Secure Electronic Transaction Specification, Book 1: Business Description, Version 1.0, May 31, 1997. Website: http://www.setco.org/download/set_bk1.pdf
55. [SET, 1997-2] SET Secure Electronic Transaction Specification, Book 1: Programmer's Guide, Version 1.0, May 31, 1997. Website: http://www.setco.org/download/set_bk2.pdf
56. [SET, 1997-3] SET Secure Electronic Transaction Specification, Book 3: Formal Protocol Definition, Version 1.0, May 31, 1997. Website: http://www.setco.org/download/set_bk3.pdf
57. [Steffan, 2002] J. Steffan and M. Schumacher, "Collaborative Attack Modeling", ACM Press, pp. 253-259, 2002, ISBN: 1-58113-445-2. website: http://doi.acm.org/10.1145/508791.508843
58. [Treese, 1998] G. Treese and L. Stewart, "Designing Systems for Internet Commerce", pp. 209-294, Addison-Wesly, 1998. ISBN: 0-201-57167-6
59. [UML] Unified Modeling Language (UML), version 1.4. website: http://www.omg.org/technology/documents/formal/uml.htm
60. [Viega, 2002] J. Viega and G. McGraw, "Building Secure Software", Addison-Wesley, 2002. ISBN: 0-201-72152-X
61. [Vollbrecht, 2000-1] J. Vollbrecht, P. Calhoun, S. Farrell, L. Gommans, G. Gross, B. de Bruijn, C. de Laat, M. Holdrege, and D. Spence "AAA Authorization Framework", Network Working Group, RFC2904, August 2000, website: http://www.faqs.org/rfcs/rfc2904.html
62. [Vollbrecht, 2000-2] J. Vollbrecht, Calhoun, P., Farrell, S., Gommans, L., Gross, G., de Bruijn, B., de Laat, C., Holdrege, M. and D. Spence, "AAA Authorization Application Examples", Network Working Group, RFC 2905, August 2000, website: http://www.faqs.org/rfcs/rfc2905.html
63. [Wagner, 2000] Wagner, D., Foster, J., Brewer, E., and Aiken A., "A First Step Towards Automated Detection of Buffer Overrun Vulnerabilities", Network and Distributed System

Security Symposium 2000. Website: http://www.cs.berkeley.edu/~daw/papers/overruns-ndss00.ps

64. [Wedde, 2001] H. Wedde and M. Lischka, "Modular Authorization", 2001, ACM 1-58113-350-2/01/0005
65. [Wilson, 1997] S. Wilson, "Certificates and trust in electronic commerce", Information Management & Computer Security, Vol. 5, No. 5, 1997, pp. 175-181, MCB University Press. ISSN: 0968-5227