Misuse Cases: earlier and smarter information security

By defining the scenarios in which computer systems could be misused, security professionals can test more thoroughly and close down risk more quickly.

By John **Neil Ruck** and **Geraint Price**

MISUSE CASES PROVIDE information security professionals with a way to identify how their information system might be used inappropriately, either deliberately (an attack) or accidentally (a mistake). This article illustrates three techniques based on misuse cases by applying them to a case study (an IT Contractor Management System).

- **Technique 1:** to identify and prioritise potential misuses (described by misuse cases);
- **Technique 2:** to derive the security requirements needed to counter the misuse cases;
- **Technique 3:** a novel approach to develop test scenarios to verify that security requirements have been met.

1. INTRODUCTION

In the information security profession we are losing the battle. Schneier [Schneier, 2004] argues that today's computers and networks are less secure than they were earlier and they will be even less secure in the future. To stand a chance of reversing our fortunes the information security profession needs to focus on securing information systems faster and smarter. Software engineers recognised over 15 years ago [Davis, 1993] that it is much cheaper to identify and fix requirement-related errors as early on in the software lifecycle as possible. There is much the information security profession can learn from software engineers, and indeed vice versa!

Whilst software engineers use techniques based on "use cases" to describe the functionality an information system must provide; information security professionals can use techniques based on "misuse cases" to describe functionality an information system must not provide.

In this article we apply the three techniques to a hypothetical case study (an IT Contractor Management System, described in Section 4) to perform the following, security-enhancing activities:

- Derive the functionality an information system must not allow; referred to as its misuses and described in the form of misuse cases (Technique 1);

- Derive the security requirements that will counter the misuse cases we are most concerned about (Technique 2);
- Derive test scenarios in order to verify that we have implemented the security requirements correctly and that the misuses are no longer possible (Technique 3).

Techniques 1 and 2 are based on techniques taken from the existing literature, with some modifications and extensions that we propose to make them as applicable to the 'real world' as possible. Technique 3 is a novel approach based on combining two existing techniques that is unique to our Technical Report [Ruck, 2008].

The remained of this article is organised as follows:

- **Section 2** introduces the concept of an information systems lifecycle; to help us understand how to incorporate security at the earliest opportunity;
- **Section 3** provides a brief description of the case study we will use to demonstrate our techniques;
- **Section 4** gives an overview of misuse cases and use cases;
- **Section 5** applies the three techniques based on misuse cases in detail;
- **Section 6** concludes and provides recommendations for future work.

2. THE INFORMATION SYSTEMS LIFECYCLE

Information systems do not just happen; software engineers manage their development using a system development process. There are a range of these available; the one we chose is the Unified

The information security profession needs to focus on securing information systems faster and smarter.

Process (UP) because many other development lifecycles are derived from it and the UP has no intellectual property constraints. The UP is shown in FIGURE 1 on page 4; and described in detail in [Jacobson, Booch, & Rumbaugh, 1999].

The phases of the system lifecycle are shown in the columns of FIGURE 1; the system starts in the 'inception' phase. The rows of FIGURE 1 describe

the workflows that are being followed throughout the lifecycle. The shapes show the amount of effort expended on each workflow in relation to the phase in the system lifecycle. The thicker the shape the more of the workflow is done at that point in the lifecycle.

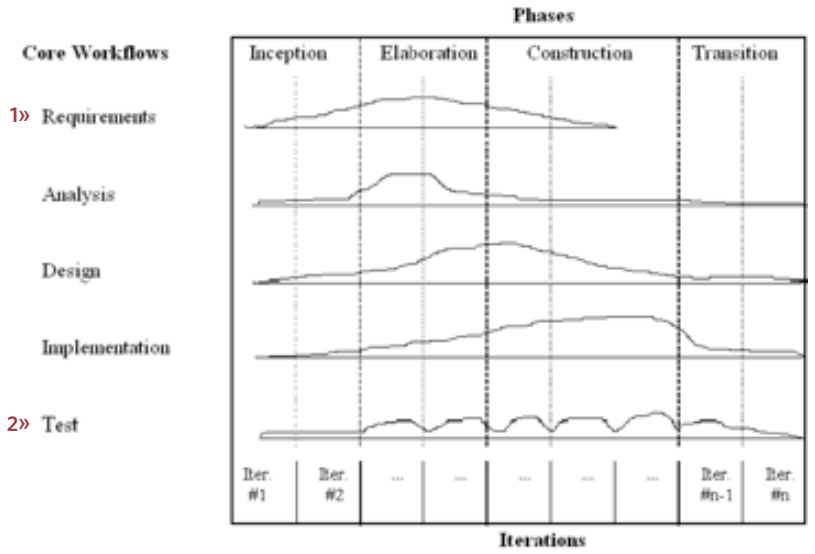
The misuse case techniques we apply relate primarily to the two numbered areas in FIGURE 1:

1) 'Requirements'—we show how misuse cases can be identified and used to capture and prioritise security requirements for information systems (Techniques 1 & 2);

2) 'Testing'—we show how test scenarios can be developed (Technique 3) to be used to verify that security requirements have been satisfied throughout the system lifecycle ('traceability of requirements').

FIGURE 1

THE UNIFIED PROCESS (ELECTRONIC COPY FROM [KIVISTÖ, 2000])



3. CASE STUDY

The case study we use in both this article and the Technical Report [Ruck, 2008] is an IT Contractor Management System. It is a hypothetical information system that is used to demonstrate how the three techniques can be applied. This section summarises the case study for the reader's convenience.

The IT Contractor Management System is an information system that is owned and managed by a hypothetical organisation whose business is providing IT contractors to large multinational companies and managing the relationship. The

organisation deals with administrative tasks such as organising jobs for and making payments to the IT contractors; it employs Human Resources (HR) administrators to do such tasks.

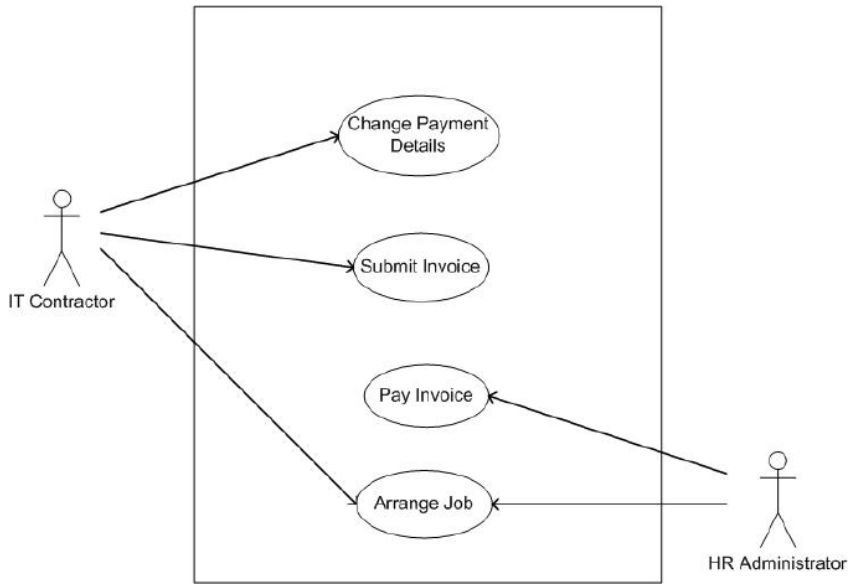
The information system provides the following functions to its users:

- IT contractors can 'Change (their own) Payment Details'
- IT contractors can 'Submit Invoices'
- HR administrators can 'Pay Invoices'
- IT contractors and HR administrators can 'Arrange a Job'

The information system is connected to the Internet and provides a web-based interface to IT contractors.

FIGURE 2

USE CASE DIAGRAM FOR THE IT CONTRACTOR MANAGEMENT SYSTEM



4. MISUSE CASES—BAD USE CASES

Use cases are used to describe the functionality that an information system provides to actors (anyone or anything that interacts with the system). They are commonly used in software engineering during the requirements and analysis workflows of the unified process. Use cases can be either diagrammatic or textual, for more information see Bittner et al [Bittner & Spence, 2003].

The diagrammatic use case in FIGURE 2 shows two actors from the case study (the HR administrator and the IT contractor). Significantly, the HR adminis-

trator can pay invoices and the IT contractor can only submit invoices, not pay them. A possible motivation for an escalation of privileges attack?

Use cases are things that we want to happen; now we introduce the concept of a misuser, an actor who will misuse our system. These misuses are communicated using misuse cases. Misusers could be illegitimate users of the system or legitimate users of the system ('insiders'); shown in FIGURE 3 in black and grey respectively.

FIGURE 3

A SIMPLE MISUSE CASE DIAGRAM

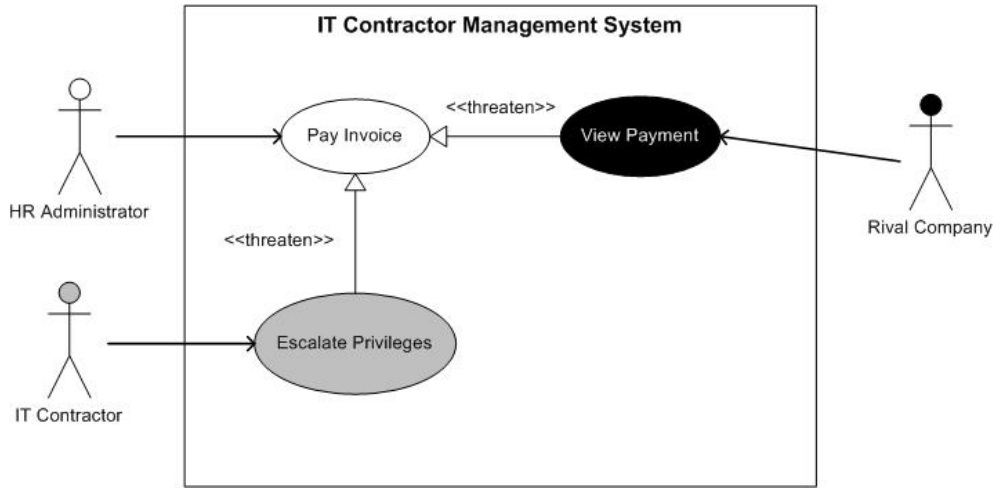


FIGURE 3 shows how we can communicate misuse cases for the IT Contractor Management information system in a visually powerful way (in grey and black). There are two misuse cases shown: 'Escalate Privileges' and 'View Payment'; the former misuse is conducted by an internal misuser (a rogue IT contractor). Essentially the information security professionals can take a use case diagram from the software engineers and inject a healthy dose of paranoia by thinking: "How can the information system be misused by the bad guys?"

Security professionals forget the malicious insider at their peril. In FIGURE 3 we have considered what a malicious IT contractor may do with the IT Contractor Management information system. Escalating their privileges to become a HR administrator would be quite an attractive misuse because they could then pay people of their choosing; in Section 5.2 we consider this misuse in more detail.

5. APPLYING THE MISUSE CASE TECHNIQUES

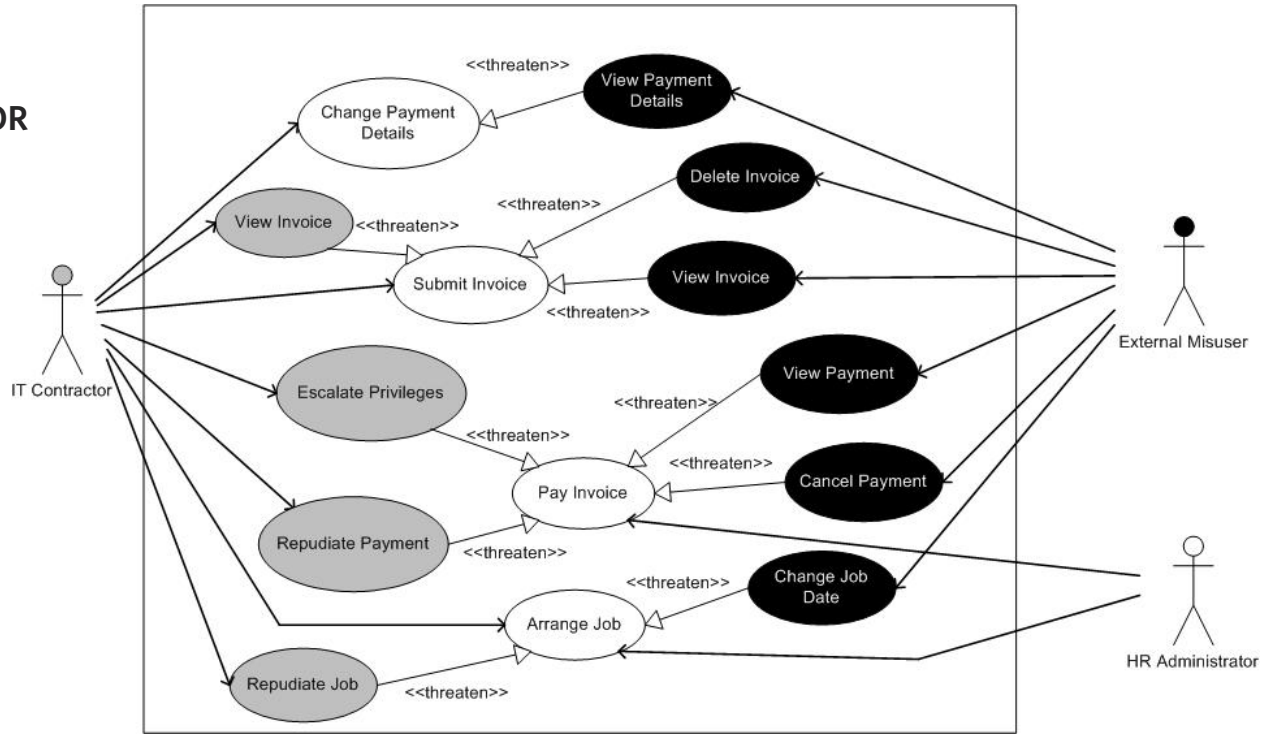
In this section we illustrate three complementary techniques based on misuse cases. They are most effective when applied in numerical order as each technique builds on the information from the previous ones.

5.1 TECHNIQUE 1: MISUSE CASES TO IDENTIFY TOP-LEVEL MISUSES OF A SYSTEM

Expanding on the ideas in FIGURE 3, we can really give our software engineering friends something to think about by applying some 'bad guy thinking' to the use case diagram from FIGURE 2 and super-imposing our misuse cases to create FIGURE 4.

FIGURE 4

A MISUSE CASE DIAGRAM FOR THE IT CONTRACTOR MANAGEMENT INFORMATION SYSTEM



In FIGURE 4 we have identified how our information system could be misused by internal and external actors (we have rightly or wrongly assumed our HR Administrators would not misuse the information system). Notice how the misuse cases relate to the use cases of the system, illustrating the point that every individual use case affords an opportunity for a misuse. For example by enabling jobs to be arranged online we have given malicious external misusers the opportunity to 'Change Job Dates' and IT contractors the ability to 'Repudiate Jobs' (by claiming that "It could have been a hacker!").

FIGURE 4 is not a 'complete' diagram because it does not show all of the possible misuse cases for our information system; this is for two principal reasons:

- Space constraints mean we have not recorded all possible misuses, just the ones we judged to be most important;
- An unstructured application of the technique means that we may not have identified all possible misuses of our system.

Verifiable completeness of misuse analysis is commonly understood to be unachievable because it is not a 'formal analysis method'—refer to the Technical Report [Ruck, 2008] for a more in depth discussion. We propose two modifications to Technique 1 that make the analysis more rigorous.

5.1.1 SUGGESTED MODIFICATION 1: TABULATION

To address the issue of space constraints we propose the use of a table; an example of this is shown in the table below.

Prioritising the misuse cases (in a similar man-

EXAMPLE MISUSES TO THE IT CONTRACTOR MANAGEMENT SYSTEM

PRIORITY	DESCRIPTION OF MISUSE	MISUSER	IMPACT	LIKELIHOOD
1	Escalate Privileges	Internal User	VH	M
2	View Invoice (of another user)	Internal User	M	H
3	View Payment	External Misuser	H	M
4	...			

HOME
THE INFORMATION SYSTEMS LIFECYCLE
CASE STUDY
USE AND MISUSE CASES
MISUSE CASE TECHNIQUES
CONCLUSIONS AND FURTHER WORK

ner to risks) enables us to know which misuses we are most concerned about; and therefore are more important to mitigate.

5.1.2 SUGGESTED MODIFICATION 2: APPLICATION OF THE STRIDE CLASSIFICATION

To address the issue of an unstructured application of the misuse case techniques we propose that each individual use case in turn is considered against the STRIDE classification system (Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service and Escalation of Privilege), by asking the question "Which of the STRIDE misuses are we concerned about for this individual use case?".

We maintain that all of the misuse cases identified in FIGURE 4 could be derived from a STRIDE classification; for example: the 'View Invoice' misuse case could be derived from the 'Information Disclosure' STRIDE classification.

For more information on STRIDE refer to Microsoft's Threat Modelling process [Swiderski & Snyder, 2004].

5.2 TECHNIQUE 2: SECURITY USE CASES

Application of Technique 1 and our suggested enhancements to the IT Contractor Management System will result in a prioritised list of top-level misuse cases that is as complete as possible. At this stage however we will not know sufficient detail of how these misuses can be conducted. To get more detail it is necessary to develop textual descriptions; security use cases provide an ideal vehicle for this.

Security use cases were initially proposed by Sindre et al. [Sindre, Firesmith, & Opdhal, 2003]. They are a powerful tool that can be used to communicate both the details of the misuses and what the system needs do to counter the misuses. Firesmith [Firesmith, 2003] elaborated on the original technique and we adopt his approach and make some slight modifications; see the Technical Report [Ruck, 2008] for more detail.

Creating a security use case is time consuming so it makes sense to focus on the misuse cases that have been identified as high priority (by application of Technique 1). To that end we have used a security use case for authorisation to develop the 'Escalation of Privilege' misuse case.

The blue arrow shown in FIGURE 5 on pages 10 and 11 traces the actions performed by the misuser; in order left to right, top to bottom. In doing this we can identify and record what the system can do to counter (prevent, detect or respond to)

(Continued on page 12)

FIGURE 5

SECURITY USE CASE FOR AUTHORISATION (TO MITIGATE THE PRIVILEGE ESCALATION MISUSE)

- **SECURITY USE CASE:** Authorisation
- **SECURITY USE CASE PATH:** Attempted escalation of privileges
- **SECURITY THREAT:** The user becomes a misuser and bypasses authorisation controls to obtain the privileges of the administrator
- **MISUSER PROFILE:** Users are IT contractors therefore it reasonable to assume a high degree of technical competence (capability). Due to recent disputes over payments it is judged to be moderately likely that users would be motivated to conduct the misuse (likelihood).
- **TRIGGER:** Always true. This can happen at any time
- **PRECONDITIONS:** 1) Misuser is authorised by the system as a user; 2) Misuse is able to upload data of their choosing to the system
- **PREVENTION REQUIREMENTS:** 1) The system must only permit on site administration of the system; 2) The system must not allow users to upload information of their choosing to the system
- **POST CONDITIONS:** 1) The system should have prevented the user escalating their privileges; 2) The system must have detected and responded to the user attempting to escalate their privileges
- **MITIGATION GUARANTEE:** Verified by testing at each stage in the system development lifecycle
- **TECHNOLOGY AND DATA VARIATIONS:** If zero-day attacks are used it is highly unlikely it will be possible to prevent the attack (hence the importance of detection and response)

(CONTNUED ON PAGE 11)

FIGURE 5

SECURITY USE CASE FOR AUTHORISATION (CONTINUED FROM PAGE 10)

			SYSTEM REQUIREMENTS	
IT CONTRACTOR INTERACTIONS	**HR ADMINISTRATOR INTERACTIONS**	**MISUSER INTERACTIONS**	**SYSTEM INTERACTIONS**	**SYSTEM ACTIONS**
The user authenticates to the system and is authorised as a user				
		User (now misuser) uploads a malicious program to the system		1) The system must log user transactions 2) The system must scan all uploaded data for malicious content
		Misuser attempts to execute malicious program and gain administrator authorisation	The system *could* alert other administrators that an escalation of privilege has occurred	The system must log that an escalation of privilege has occurred
	Administrators investigate the event, if it is unexpected consider shutting down the system	Misuser attempts to intercept notification	The system must require notification of receipt for any alerts	
		Misuser attempts to cover their tracks by deleting system logs		The system log must be append only (no entries can be deleted)

(Continued from page 9)
what the misuser is doing and record them in the grey shaded area of the table in FIGURE 5. These countermeasures are the system's security requirements.

The power of this technique is that it provides a means of binding the security requirements (in the grey area in FIGURE 5) with a top-level misuse case from Technique 1 (privilege escalation) by encapsulating them in a security use case. From the table on page 8, we know that the 'Privilege Escalation' misuse case is the one we consider highest priority and therefore the system security requirements that counter privilege escalation (contained in the security use case for authorisation) should also be the highest priority. If money and time became an issue later on in the system lifecycle we would look to 'de-scope' the requirements relating to our lower priority misuses cases instead.

5.3 TECHNIQUE 3 : ADAPTING SECURITY USE CASES TO GENERATE TEST SCENARIOS

Application of Technique 2 has enabled us to identify and communicate the security requirements that we really care about. The question now becomes *"how can we verify the security requirements have been implemented successfully throughout the system lifecycle?"*

Technique 3 is a novel approach that combines activity diagrams (for more information see Bittner et al [Bittner & Spence, 2003]) and security use cases (Technique 2) to describe test scenarios for an information system. Individual system security requirements can be verified by conducting the test cases (TCs) that make up the test scenario. FIGURE 6 on page 12 shows the test scenario for the privilege escalation misuse for the IT Contractor Management System; the test cases are shown in the red boxes.

If a test case fails then we will know that the corresponding security requirement has not been satisfied and remedial action will be needed (if it is deemed to be of sufficiently high priority).

By placing the test scenario shown in FIGURE 6 in the hands of the system testers, it becomes incredibly powerful because it gives them a simple way of verifying:

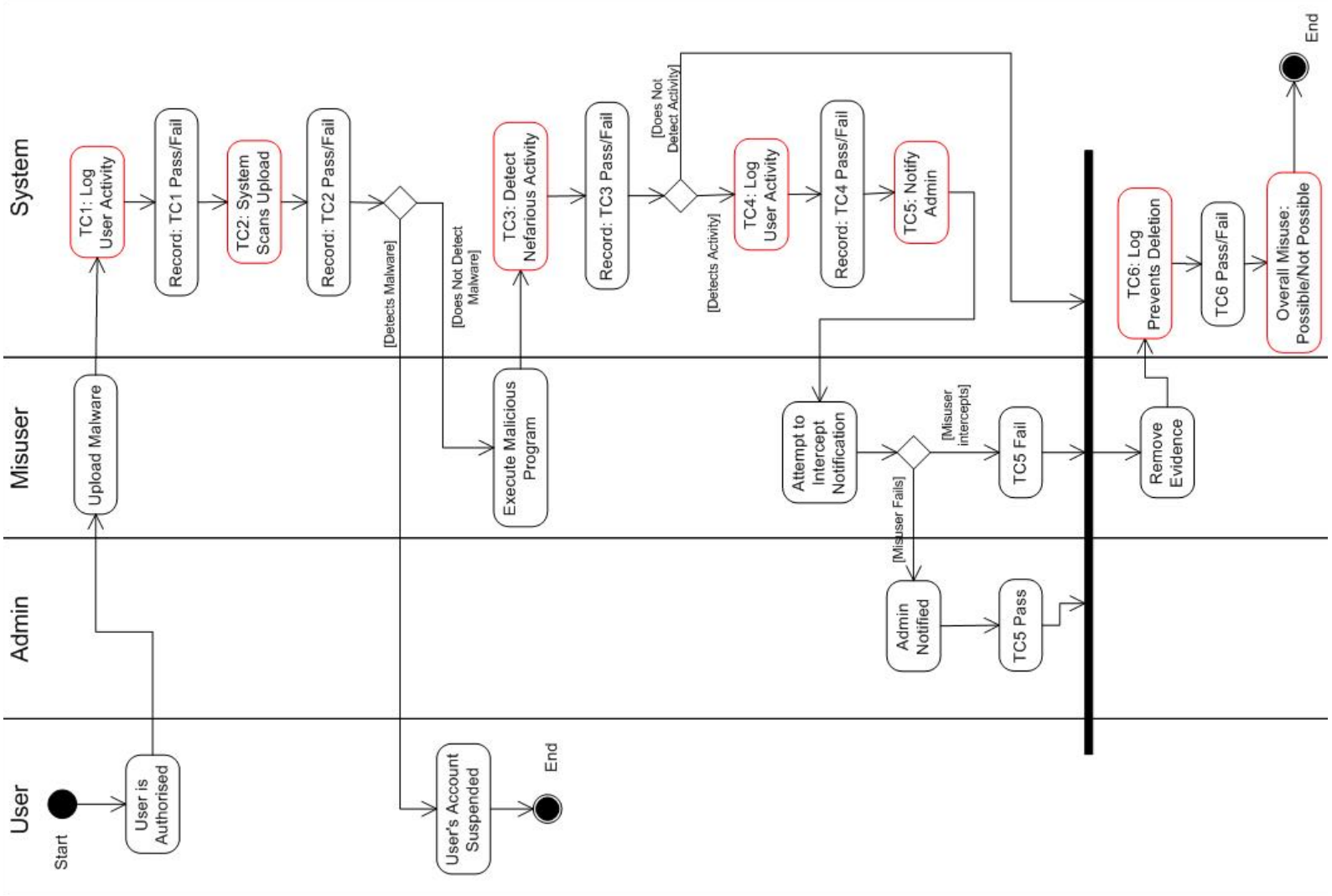
- Individual security requirements have been realised by the implementation;
- Misuses of the system are prevented, detected or responded to by the information system.

Conducting the test scenarios throughout the

(Continued on page 14)

FIGURE 6

ACTIVITY DIAGRAM SHOWING THE TEST SCENARIO AND TEST CASES (TCS) FOR THE 'PRIVILEGE ESCALATION' MISUSE FOR THE IT CONTRACTOR MANAGEMENT SYSTEM



(Continued from page 12)
system development lifecycle, as directed by the UP (shown in FIGURE 1), will achieve 'traceability of the requirements' throughout the phases of the information system.

6. CONCLUSIONS AND FURTHER WORK

6.1 SUMMARY

We have taken the 'best of breed' misuse case techniques and applied them to an IT Contractor Management System. We have proposed our own modifications and extensions to the existing techniques (Techniques 1 & 2) and come up with a novel approach that combines two existing techniques (Technique 3). By applying the techniques to the case study we have identified:

- The potential misuses of our information system and the ones we care about most (Technique 1);
- The system security requirements that are needed to mitigate the misuses we care about most (Technique 2);
- The test scenarios that we can use to verify that the security requirements that we care about have been met throughout the lifecycle of the information system (Technique 3).

As we have shown the three techniques can be extremely powerful especially when used sequentially within the context of a system development process.

6.2 USING MISUSE CASE TECHNIQUES IN THE REAL WORLD

In Section 5.1.2 we proposed using the STRIDE classification to make the set of misuse cases we identify for our information system as complete as possible, but how can we be sure that we have identified everything?

If a complete set of pre-defined misuse cases existed, then information security professionals could do a 'sanity check' by comparing the complete list with their set of misuse cases for the information system. In 2003 Sindre et al [Sindre, Firesmith, & Opdhal, 2003] claimed that work was underway to develop a library of re-usable misuse cases. Unfortunately if such libraries already exist they are not readily available in the public domain.

Having a complete set of misuse cases as a starting point would not guarantee that we would derive a complete set of security requirements (using Technique 2) because we could still overlook important countermeasures. To allay this

we suggest that a formal check is done by comparing:

- The security requirements identified when applying Technique 2 and;
- The security functional requirements (SFRs) in the Common Criteria [Common Criteria, 2005, Part 2].

6.3 FUTURE WORK

Developing a library of re-usable misuse cases would be a significant contribution to the area. Schumacher et al [Schumacher, Fernandez-Buglioni, Hybertson, Buschmann, & Sommerlad, 2006] propose the use of patterns (solutions to common problems in a given context) as a solution.

There is no documented study of a 'real world' application of misuse case techniques. Trialling our techniques and, in particular, the modifications we propose would provide invaluable real world feedback. If you are working with misuse cases or interested in trialling the techniques in an operational environment please contact the authors of this article. ■

ABOUT THE AUTHORS

John Neil Ruck is a senior consultant working for the UK government and has 7 years of experience in the information security profession. His specialities include the evaluation of the security of information systems and Identification and Authentication (ID&A). He has worked on a number of high-profile government projects including leading the technical delivery of the ePassports Country Signing Certificate Authority (used in the digital signing of all UK ePassports).

He has an MSc with distinction in Information Security from Royal Holloway and is a guest lecturer at Coventry University on the subject of Information Security.

Dr. Geraint Price is a lecturer in information security at Royal Holloway, University of London, with research interests in public key infrastructures and denial of security attacks. Dr. Price teaches the course in Security Technologies in the Secure Digital Business pathway of the M.Sc. in Information Security at Royal Holloway.

REFERENCES:

Bittner, K., & Spence, I. (2003). Use Case Modelling. Boston: Pearson Eduction.

Common Criteria. (2005). ISO 15408:2005 Common Criteria for Information Technology Security Evaluation version 3.1. International Organisation for Standardisation.

Davis, A. M. (1993). Software Requirements: Objects, Functions and States. Prentice-Hall.

Firesmith, D. (2003). Security Use Cases. Journal of Object Technology , 2 (3), 53-64.

Jacobson, I., Booch, G., & Rumbaugh, J. (1999). The Unified Software Development Process. Reading: Addison Wesley.

Kivistö, K. (2000, December). A Third Generation Object-Oriented Process Model: Roles and Architectures in Focus. Retrieved August 28, 2008, from University of Oulu, Finland: http://herkules.oulu.fi/isbn9514258371/html/c199.html

Ruck, J. (2008, September). Applying Misuse Cases to improve the Security of Information Systems, MSc Dissertation for Royal Holloway, University of London: http://www.ma.rhul.ac.uk/static/techrep/2009/RHUL-MA-2009-05.pdf

Schneier, B. (2004). Secrets & Lies- Digital Security in a Networked World (with new information post-9/11 security). Indianapolis: Wiley Inc.

Schumacher, M., Fernandez-Buglioni, E., Hybertson, D., Buschmann, F., & Sommerlad, P. (2006). Security Patterns- Integrating Security and Systems Engineering. Chichster: John Wiley and Sons.

Sindre, G., Firesmith, D. G., & Opdhal, A. L. (2003). A Reuse-based Approach to Determining Security Requirements. REFSQ'03 Pre-proceedings (pp. 106-114). Klagenfurt/Velden: REFSQ.

Swiderski, F., & Snyder, W. (2004). Threat Modelling. Redmond, Washington: Microsoft Press.