Comparing Two Techniques for Intrusion Visualization

Vikash Katta[1,3], Peter Karpati[1], Andreas L. Opdahl[2],
Christian Raspotnig[2,3], and Guttorm Sindre[1]

[1] Norwegian University of Science and Technology, NO-7491 Trondheim, Norway
{kpeter,guttors}@idi.ntnu.no
[2] University of Bergen, NO-5020 Bergen, Norway
andreas.opdahl@uib.no
[3] Institute for Energy Technology, NO-1751 Halden, Norway
{vikashk,christir}@hrp.no

Abstract. Various techniques have been proposed to model attacks on systems. In order to understand such attacks and thereby propose efficient mitigations, the sequence of steps in the attack should be analysed thoroughly. However, there is a lack of techniques to represent intrusion scenarios across a system architecture. This paper proposes a new technique called misuse sequence diagrams (MUSD). MUSD represents the sequence of attacker interactions with system components and how they were misused over time by exploiting their vulnerabilities. The paper investigates MUSD in a controlled experiment with 42 students, comparing it with a similar technique called misuse case maps (MUCM). The results suggest that the two mostly perform equally well and they are complementary regarding architectural issues and temporal sequences of actions though MUSD was perceived more favourably.

Keywords: requirements engineering, security, experiment, threat modeling.

1 Introduction

The increased web presence of modern enterprises due to e-commuting, e-commerce, and distributed, inter-organizational workflows have also created new opportunities for computer crime, thus accentuating the need to focus on enterprise security [1]. Security must be in focus on many levels, from high level strategy and managerial policies and down to the competence and awareness of each single employee and correct implementation and operation of each ICT application. Whether talking about the construction of new information systems or the daily operation of existing ones, a vital precondition for improving their security is the ability to learn from previous failures. One possibility is to look at textual descriptions of successful attacks, like [2], but there may be several advantages in combining this with more generic visual models of attacks, both related to understandability, knowledge reuse, and integration with model-based systems engineering. Many techniques have been proposed to capture threat and attack-oriented information, for example attack trees [3], misuse cases [4] and CORAS [1], and more recently misuse case maps (MUCM) [5], focusing on modelling complex intrusions. This technique highlights the relation between security and system architecture, and thereby provides integrated overviews of user-oriented security threats, mitigations and architecture. Even though MUCM has its strengths to show intrusion across the system architecture, it might be confusing to follow the sequence of steps of the intrusion. Evaluations of MUCM [5,6] have suggested a need for better visualization of the sequence of attack steps. To the authors' knowledge, there is a lack of techniques to visualize such sequences.

This paper proposes a new threat modelling technique called misuse sequence diagrams (MUSD), aiming to give an overview on the sequence of the attacker's actions during an intrusion for different stakeholder groups. The technique is based on UML sequence diagrams [7] and misuse cases [2], and utilizes security concepts like vulnerability exploitation and mitigation. Since MUSD was meant to improve on some shortcomings of MUCM, we found it interesting to evaluate experimentally whether MUSD really provided improvement over MUCM related to these issues. Hence we performed a controlled experiment with 42 students to evaluate the participants' performance with the two techniques, as well as their opinions about the techniques.

The two techniques are relevant to enterprise-modelling practice because service-orientation has made information systems increasingly distributed across internal and external organization boundaries. Hence, software architecture has moved from being a technological concern inside "black-box" information systems, to being a "white-box" concern on the boundary between organization and technology. The techniques we present are, to the best of our knowledge, the first attempt to conceptualize this boundary and support it with useful modelling notations. We hope MUCMs and MUSDs can contribute to understanding distributed service-oriented information systems and their architectures from an enterprise and organizational context.

The rest of the paper is structured as follows. Section 2 discusses background and related work. Then, section 3 presents misuse sequence diagrams. Sections 4 and 5 present the research method and the experiment results, respectively. Section 6 discusses the findings, and section 7 concludes the paper.

2 Background and Related Work

According to the definitions of RFC 2828 [8], a *vulnerability* is a weakness in a system's design, implementation, or operation/management that can be exploited to violate its security policy. A *threat* is a potential for violation of security, depending on circumstance, capability, and an action / event that could cause harm. A *countermeasure/mitigation* is something that reduces a threat or attack by eliminating or preventing it, minimizing the harm caused, or by reporting it to enable corrective action.

Misuse cases (MUC) [4,9] are used for threat modelling and security requirements elicitation. While use cases (UC) present the required behaviour of a system, MUC capture undesired behaviour, extending UC with new elements like misuser, misuse case and mitigation use case, as well as new relations like threaten and mitigate. MUC allow an early focus on security in the development process and facilitate discussion among a wide group of stakeholders.

Misuse case maps (MUCM) [5] is a recently proposed technique combining MUC and use case maps (UCM) [10] for an integrated view of security issues and system architecture. MUCM can be used to visualize the trace of an intrusion on the

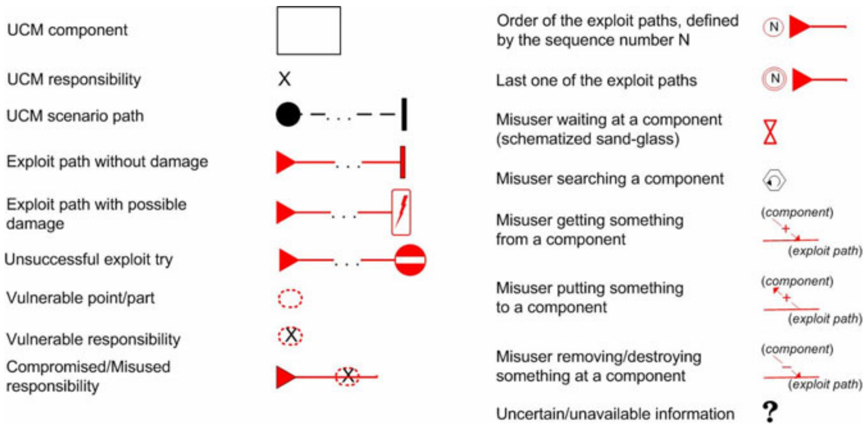
UCM component
UCM responsibility
UCM scenario path
Exploit path without damage
Exploit path with possible damage
Unsuccessful exploit try
Vulnerable point/part
Vulnerable responsibility
Compromised/Misused responsibility
Order of the exploit paths, defined by the sequence number N
Last one of the exploit paths
Misuser waiting at a component (schematized sand-glass)
Misuser searching a component
Misuser getting something from a component
Misuser putting something to a component
Misuser removing/destroying something at a component
Uncertain/unavailable information
(component)
(exploit path)
N

Fig. 1. MUCM notation and its interpretation

architecture of the system, visually based on the UCM notation extended by vulnerabilities, exploit paths and mitigations. An experiment performed to evaluate MUCM showed that it significantly improved the understanding of intrusions and identification of mitigations when compared to MUC combined with a system architecture outline [5].

Fig. 1 shows the MUCM notation. It extends UCM's basic notation with *intrusions*, represented by *exploit paths* that cut through *vulnerable parts* of the system. Each path starts with a triangle and different symbols at the end indicate the outcome. The number on the path indicates its order in a bigger sequence of paths. The system may have vulnerable points or parts which are susceptible to threats, materialized in the attack if the exploit path crosses the vulnerability.

Fig. 2 shows a partial example about a technical entry into the computer system of a company for penetration test purposes [2]. The tester first made 3 unsuccessful attack attempts against the Apache server and firewall. The fourth attempt utilised an undocumented Solaris feature (portmapper - rpcbind - bound to port 32770) to get the dynamic port of the mount daemon (mountd) from the portmapper and direct an NFS request to it, thus succeeding to remotely mount and download the target file system.

As for other related methods, [11] proposes a framework for object-oriented security requirements analysis based on UC, MUC and security use cases for the elicitation and analysis of security requirements in embedded systems. In addition to the framework, misuse sequence diagrams have been introduced to *better* explain a single misuse case scenario. [12] proposes an aspect-oriented methodology for designing secure applications. The methodology uses sequence diagrams for three purposes: describing functionality (primary model), describing attacks on the functionality (misuse model), and describing the incorporation of security mechanisms (security-treated model). The MUSD technique proposed here is similar to those of [11] and [12]. However, they neither visualize complex multi-stage intrusion scenarios nor how vulnerabilities of system components are exploited and mitigated.

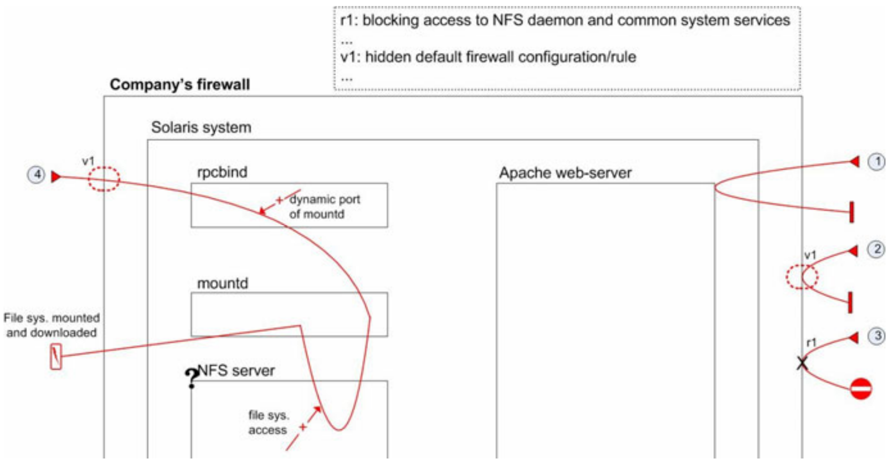
r1: blocking access to NFS daemon and common system services
...
v1: hidden default firewall configuration/rule
...
Company's firewall
Solaris system
rpcbind
dynamic port of mountd
mountd
NFS server
file sys. access
Apache web-server
File sys. mounted and downloaded
v1
r1
1
2
3
4

Fig. 2. MUCM for the (part of) penetration test example from the experiment

CORAS [1] is a method for security analysis using system descriptions based on UML diagrams as an input to traditional risk analysis techniques, such as HazOp [13] and Fault Tree Analysis (FTA) [14]. While CORAS offers a specialized set of diagrams for security risk analysis, only using UML diagrams as an input, the MUSD is solely based on MUC and SD. MUSD is not a method for security risk analysis, just a technique for visualizing complex intrusion scenarios.

It could be tempting to compare MUSD to STAIRS [15], which is a method with focal point on refinement of interactions in UML. STAIRS allows among others for the specification of behaviour that shall not be allowed in an implementation, but does not refer to the security or risk domain. However, MUSD presents a specialized notion for security and intrusion visualization, while STAIRS' focus is on a general refinement of UML based specifications thus a comparison does not seem relevant.

UMLSec is an UML extension for secure systems development which can be used "to evaluate UML specifications for vulnerabilities using a formal semantics of a simplified fragment of UML" [16]. Its focus is on formal verification of specifications (e.g. for a protocol) against different adversary types. Although sequence diagrams are included in UMLSec, they rely on other diagram types specifying the broader context. Furthermore, UMLSec applies heavyweight methods which need specific trainings. MUSD aims to facilitate discussion of different stakeholder groups allowing competency transfer and trade-off considerations early in the system development.

3 MUSD

Misuse sequence diagrams (MUSD) combine misuse cases (MUC) and sequence diagrams (SD), to depict and analyze complex intrusion scenarios. MUSD show involved objects, their vulnerabilities and how these were misused. The notation extends UML sequence diagrams as shown in Fig. 3. Just like by the MUC notation, regular and misuse symbols can be combined in the same MUSD diagram. Attack-related

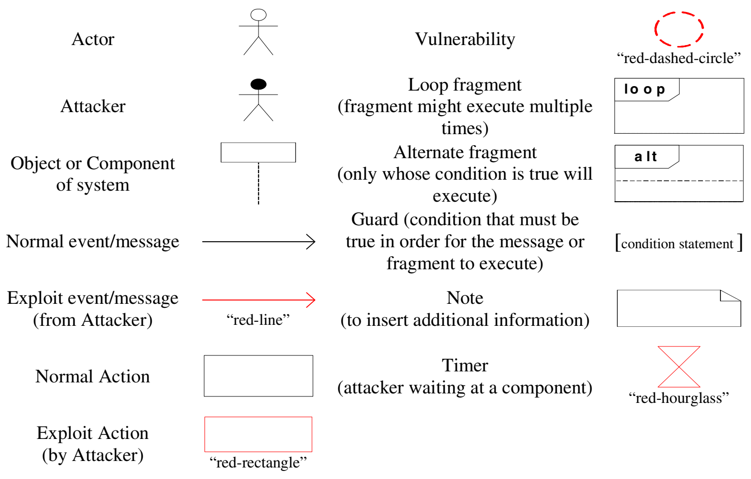
Actor
Vulnerability
"red-dashed-circle"
Attacker
Loop fragment
(fragment might execute multiple times)
loop
Object or Component of system
Alternate fragment
(only whose condition is true will execute)
alt
Normal event/message
Guard (condition that must be true in order for the message or fragment to execute)
[condition statement]
Exploit event/message
(from Attacker)
"red-line"
Note
(to insert additional information)
Normal Action
Timer
(attacker waiting at a component)
"red-hourglass"
Exploit Action
(by Attacker)
"red-rectangle"

Fig. 3. MUSD basic notation and its interpretation

symbols are shown in red color since inversion (a la misuse cases) does not work for arrows. Exploit messages are messages originating from the attacker with the intention of harming the system. Intrusions are represented by one or more exploit messages using vulnerabilities of the objects in the system. Action symbols are used to represent parts of the intrusion scenario which are unclear or not detailed enough. Exploit actions are performed by the attacker with the intention of attacking the system. Objects which are part of an action will have their lifelines covered by the rectangle denoting the action.

The sequence of exploit messages and actions shows the steps taken by the attacker. These steps are mostly causally related; each building on the result of the previous steps. Notes might appear for explanations. Fig. 4 presents a part of the MUSD depicting the same penetration test case as shown with a MUCM in Fig. 2.

4 Research Method

The purpose of the experiment was to evaluate whether MUSD would be better than MUCM for conveying attack sequences. On the other hand, a gain in this respect could result in other weaknesses instead, especially it would be natural to suspect that MUSD would be poorer than MUCM in conveying the relationship between attacks and system architecture. Hence, the experiment compared the understanding of case descriptions resulting from usage of two notations, both for attack sequence and architectural aspects. Understanding may be a goal in its own right in enterprise systems development, but more often it will be the basis for various problem solving activities. Hence the subjects performed one task measuring understanding by a set of True/False questions, and another addressing problem solving in terms of identifying threats and possible mitigations in the given cases. In the following the experiment design, variables and hypotheses are explained in more detail.

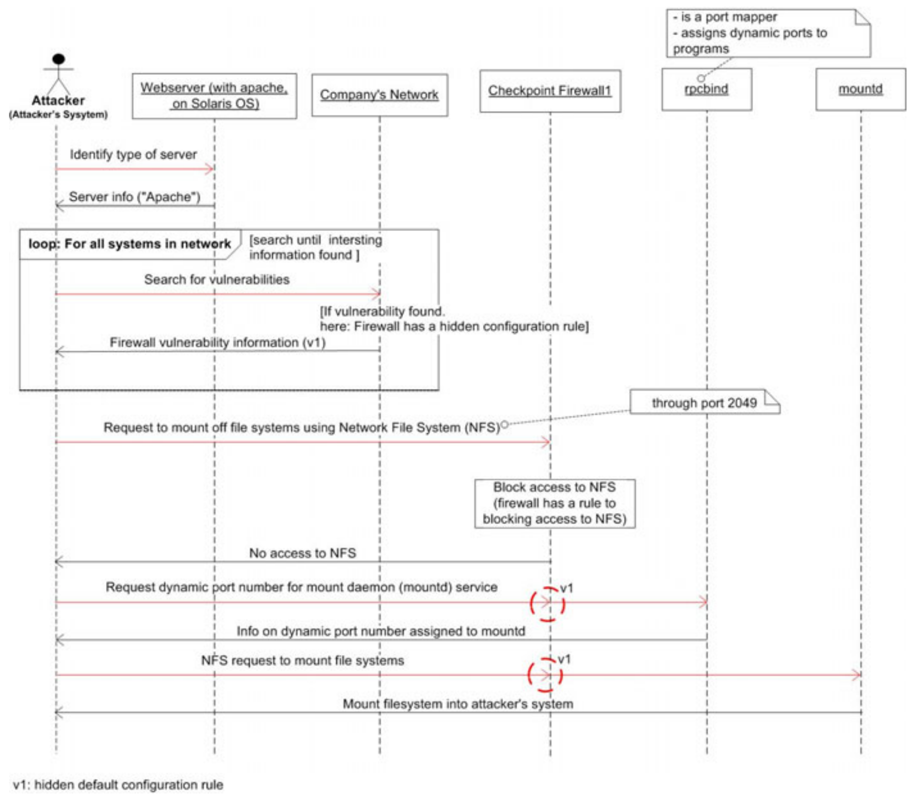
- is a port mapper
- assigns dynamic ports to programs
Attacker (Attacker's Sysytem)
Webserver (with apache, on Solaris OS)
Company's Network
Checkpoint Firewall1
rpcbind
mountd
Identify type of server
Server info ("Apache")
loop: For all systems in network
[search until intersting information found]
Search for vulnerabilities
[If vulnerability found. here: Firewall has a hidden configuration rule]
Firewall vulnerability information (v1)
through port 2049
Request to mount off file systems using Network File System (NFS)
Block access to NFS (firewall has a rule to blocking access to NFS)
No access to NFS
Request dynamic port number for mount daemon (mountd) service
v1
Info on dynamic port number assigned to mountd
NFS request to mount file systems
v1
Mount filesystem into attacker's system
v1: hidden default configuration rule

Fig. 4. MUSD for the (part of) penetration test example from the experiment

4.1 Experimental Design

To compare the two techniques, we conducted a controlled experiment with 42 subjects who used the two techniques individually on two different cases described in the literature, a *bank intrusion* [2] and a *penetration test* [2]. To control for the order in which the techniques were used and the cases were solved, a Latin-Squares experimental design was used as shown in Table 1. The within-experiment data regarding understanding, performance and perception thereby became paired, comprising two dependent samples. Table 1 shows the order in which the techniques were used and cases were solved by the groups. Group 1's experiment sheet is available at [17].

Table 1. Latin-Squares experimental design used in the experiment

Case order: Technique order:	Bank intrusion before penetration test	Penetration test before bank intrusion
MUCM before MUSD	Group 1	Group 2
MUSD before MUCM	Group 3	Group 4

4.2 Variables

After controlling for the participants' *backgrounds*, for each combination of technique and case, three types of tasks were solved: an *understanding* task, a *performance* task and a *perception* task. Table 2 summarizes the main variables used in the analysis.

Background was measured by a pre-task questionnaire addressing the participants' self-assessed knowledge of SD, MUSD, UCM, MUCM and security in general on a 5-point scale. They were also asked to report their numbers of completed *semesters of ICT studies* and months of *ICT-relevant work experience*.

Table 2. Variables used in the experiment

Name	Explanation
TECH=MUCM, TECH=MUSD	The *technique* used in that part of the experiment, either MUCM or MUSD.
CASE=BANK, CASE=PEN	The *case* solved in that part of the experiment, either BANK (the bank intrusion) or PEN (the penetration test).
KNOW_MOD, KNOW_SD, KNOW_MUSD, KNOW_UCM, KNOW_MUCM, KNOW_SEC	The participants' self-assessed knowledge about *systems modelling* (KNOW_MOD), *sequence diagrams* (KNOW_SD), *misuse sequence diagrams* (KNOW_MUSD), *use case diagrams* (KNOW_UCM), *misuse case diagrams* (KNOW_MUCM) and *security analysis* (KNOW_SEC) on a 5-point scale, where 1 is “Never heard about it” and 5 is “Expert”.
STUDY	The participants' self-reported *semesters* of ICT-studies.
JOB	The participants' self-reported *months* of ICT-relevant work experience.
UND_1, UND_2, ...	The 20 statements about the case, scored by the participants as either true or false. A correct assessment is scored 1 and a wrong one -1.
UND_MUCM, UND_MUSD, UND_NEUT	Sum of scores for the statements about of MUCM (architectural issues), MUSD (temporal sequence issues) and neutral aspects of the problem cases.
UND_TOT	Sum of scores for the all twenty statements about the problem cases.
VULN, MITIG	The numbers of unique vulnerabilities and mitigations identified by the participants.
VUMI	The sum of unique vulnerabilities and mitigations identified by the participants.
PER_1, PER_2, ...	Scores on the 5-point Likert scales for the individual statements about perception of the techniques.
PER_PU, PER_PEOU, PER_ITU	Average scores on the 5-point Likert scales for the four statements about perceived usefulness of, perceived ease of use of and intention to use of the techniques.
PER_AVE	Average scores on the 5-point Likert scales for all the twelve statements about the techniques.

Understanding was measured by 20 true/false questions about the case, scoring 1 point for a correct answer, -1 for incorrect, and 0 points for no answer. The statements were designed so that 6 of the statements addressed aspects where MUCM was assumed to be a better technique, 6 addressed aspects where MUSD was assumed to be better and 8 addressed aspects where neither technique was assumed to have an advantage. Specifically, we assumed MUCM would perform best for statements relating to architecture and that MUSD would perform better for statements about the sequence of activities. In each group, half the statements had a positive and half a negative formulation (so that equal numbers of *true* and *false* answers were expected).

Performance was measured by asking the participants to identify and list as many vulnerabilities and mitigations as they could in the planned system. Then the numbers of unique (and relevant) vulnerabilities and mitigations were counted. Three example vulnerabilities were given for both cases. Even though the type and criticality of the vulnerabilities are important, these issues were out of scope for the experiment. Such issues can be considered as a part of the future work.

Perception was measured by a post-task questionnaire adapted from the Technology Acceptance Model (TAM) [18,23]. 4 questions addressed *perceived usefulness* (PU), 4 addressed *perceived ease of use* (PEOU) and 4 investigated the participants' *intention to use* (ITU) the technique in the future. One statement in each group was negative, with a lower score reflecting a positive opinion, inverted before analysis.

Table 3. Hypotheses of the comparison experiment

$H1_1$	Better score for architectural Q's with MUCM than MUSD. That is, more questions about architectural issues were assessed correctly with MUCMs than with MUSDs.	UND_MUCM[MUCM] > UND_MUCM[MUSD]
$H2_1$	Better score for temporal sequence Q's with MUSD than MUCM. That is, more questions about temporal sequence issues were assessed correctly with MUSDs than with MUCMs.	UND_MUSD[MUCM] < UND_MUSD[MUSD]
$H3_1$	Different numbers of statements in the NEUT group were assessed correctly with MUCMs and MUSDs.	UND_NEUT[MUCM] ≠ UND_NEUT[MUSD]
$H4_1$	Different numbers of statements were assessed correctly with MUCMs and with MUSDs.	UND_TOT[MUCM] ≠ UND_TOT[MUSD]
$H5_1$	Different numbers of vulnerabilities were identified with MUCMs and with MUSDs.	VULN[MUCM] ≠ VULN[MUSD]
$H6_1$	Different numbers of mitigations were identified with MUCMs and with MUSDs.	MITI[MUCM] ≠ MITI[MUSD]
$H7_1$	Different numbers of vulnerabilities and mitigations were identified with MUCMs and with MUSDs.	VUMI[MUCM] ≠ VUMI[MUSD]
$H8_1$	The usefulness of MUCMs and MUSDs were perceived differently.	PER_PU[MUCM] ≠ PER_PU[MUSD]
$H9_1$	The ease of use of MUCMs and MUSDs were perceived differently.	PER_PEOU[MUCM] ≠ PER_PEOU[MUSD]
$H10_1$	The intentions to use MUCMs and MUSDs again were different.	PER_ITU[MUCM] ≠ PER_ITU[MUSD]
$H11_1$	MUCMs and MUSDs were perceived differently.	PER_AVE[MUCM] ≠ PER_AVE[MUSD]

4.3 Hypotheses

The hypotheses for our experiment are listed in Table 3. The corresponding set of null hypotheses, as well as an additional set of hypotheses about correlations between understanding ($H1_1$-$H4_1$), performance ($H5_1$-$H7_1$) and perception ($H8_1$-$H11_1$), are omitted here for space reasons.

4.4 Experimental Procedure

The 42 participants of the experiment were recruited from a class of second year computer science students, receiving financial support for an excursion as "payment" for participating. Each group consisted of 10 or 11 students solving the task under equal conditions (same room, same time limits). The experiment comprised 10 steps:

1. Filling in the pre-experiment questionnaire (2 min)
2. Reading a short introduction to the experiment (1 min)
3. Using the first assigned technique on the first assigned case:
   a. Reading the introduction to the first technique (1.5 pages, 9 min)
   b. Reading the textual description of case 1 while looking at the related diagrams (3-4 pages, 12 min)
   c. Answering 20 true/false questions about the case (8 min)
   d. Finding as many vulnerabilities and mitigations as possible (11 min)
   e. Filling in post-experiment questionnaire (4 min)
4. Easy physical exercises as a break (2 min)
5. Repeat steps 3a-e for the second technique and case (7+ 14 + 5 + 10 + 4 min)

The duration of the steps was decided dynamically. There was always enough time to finish the steps, except for steps 3c and 5c, which we stopped before everyone could finish because we wanted to see how efficient the participants were.

5 Results

5.1 Comparing Backgrounds

We used Kruskal-Wallis H tests of four independent groups for all background variables to control for differences between the participant groups. We found no significant differences, neither with respect to *knowledge*, *study semesters* nor *job months*.

We used 2-tailed Wilcoxon signed-rank tests to compare the participants self-assessed knowledge backgrounds in the following areas. They reported being significantly more knowledgeable about *systems modelling* than *security analysis* (KNOW_MOD = 2.79, KNOW_SEC = 1.81, p = .000), as well as about *sequence diagrams* versus *use case maps* (KNOW_SD = 3.20, KNOW_UCM = 2.74, p = .003). There was no significant difference in knowledge about *misuse sequence diagrams* and *misuse case maps* (KNOW_MUSD = 1.37, KNOW_MUCM 1.36).

The participants reported between 2 and 4 semesters of ICT-studies, with a mean of 3.05 and a small standard deviation. They reported 2.07 months of ICT-relevant work experience. Here the standard deviation was higher due to three outliers with 6, 19 and 54 months. All the others reported 2 months or less.

Table 4. Comparison results for understanding

Statement group	Understanding					
	MUCM Mean	MUCM St.Dev.	MUSD Mean	MUSD St.Dev.	Z	Sign. (exact)
MUCM group (UND_MUCM)	3,24	2,01	2,33	2,21	-1,87	p=0.031
MUSD group (UND_MUSD)	2,36	2,37	3,62	2,44	-2,89	p=0.001
NEUT group (UND_NEUT)	5,62	1,83	5,62	1,95	-0,23	–
All statements (UND_ALL)	11,21	5,20	11,57	4,25	-0,22	–

Table 5. Effect and sample sizes for the significant differences in understanding

Statement group	Understanding				
	Pooled st.dev.	Effect size	Cohen	Hopkins	Sample size required
MUCM (UND_MUCM)	2,087	0,44	medium	small	169
MUSD (UND_MUSD)	2,376	-0,53	medium	small	114

5.2 Understanding

We performed Wilcoxon signed-rank tests of two paired groups for all understanding variables using exact significances to compare how well the participants assessed statements about their cases using MUCM and MUSD (see Table 4). As expected, use of MUCM gave more correct answers on architectural questions than MUSD (p = .031, 1-tailed), while MUSD yielded more correct answers than MUCM for questions about temporal sequence (p = .001, 1-tailed). There were no other significant differences. Hence, H1 and H2 are confirmed while H3 and H4 are rejected.

The effect sizes for the groups with significant differences proved to be medium according Cohen's classification [19] or small according Hopkins's [20]. It was 0.436 for MUCM group (UND_MUCM) case and -0.53 for MUSD groups (UND_MUSD) case (see Table 5). The positive value here means that MUCM had the advantage while the negative favours the MUSD technique.

We also used Wilcoxon signed-ranks to control for differences in the participants' understanding of the two different cases and their performance depending on technique order, but found no significant differences.

5.3 Performance

We had two completely blank responses for the task of identifying vulnerabilities and mitigations. Both came from groups of 11 participants. After removing the outliers,

Table 6. Comparison results for performance

Identification task	Performance					
	MUCM Mean	MUCM St.Dev.	MUSD Mean	MUSD St.Dev.	Z	Sign. (exact)
Vulnerabilities (VULN)	4,18	1,75	4,08	1,67	-0.75	–
Mitigations (MITI)	3,22	1,72	3,42	1,54	-0.18	–
Both (VUMI)	6,75	3,23	7,32	3	-0.49	–

we performed Wilcoxon signed-rank tests of two paired groups for all performance variables using exact 2-tailed significances to compare how well the participants identified vulnerabilities and mitigations using MUCM and MUSD. We found no significant results, thus rejecting H5, H6, and H7.

We also used Wilcoxon signed-ranks to compare the participants' performance for the two cases for the first versus second technique used. We found that significantly more vulnerabilities and mitigations (2-tailed case for both) were identified for the bank intrusion case.

We did not attempt to rate, categorise or otherwise analyse the vulnerabilities and mitigations in further detail, leaving this for further work. In particular, further work should investigate whether there are systematic differences between the types of vulnerabilities and mitigations identified using the two techniques.

5.4 Perception

Finally, we performed Wilcoxon signed-rank tests of two paired groups for all perception variables using exact 2-tailed significances to compare how the participants perceived the techniques in terms of usefulness and ease of use and whether they intended to use them again in the future. The participants perceived MUSDs significantly more positively than MUCMs, both for perceived usefulness (p = .002), perceived ease of use (p = .000), intention to use (p = .000) and on average for all perception questions (p = .000). Hence, hypotheses 8, 9, 10 and 11 are all confirmed.

All individual questions also gave significant results for the Wilcoxon signed-rank test in favour of MUSD, except statement 5, which did not give a significant result in either direction. This was related to the PU variable and stated that the technique "would be useless in analyzing security vulnerabilities of computer systems."

The effect sizes for the perception measures were higher than for understanding: 0.547, -1.19, -0.957 and -1.027 for PU, PEOU, ITU and their averages (see Table 8).

We also used Wilcoxon signed-ranks to compare the participants' perceptions for the bank intrusion case and the penetration test case and to compare their perceptions of the first technique they used with the second one. The only significant difference was that the participants perceived their first technique used as more useful (p = .023).

Table 7. Comparison results for perception, all p-values 2-tailed

TAM variable	Perception					
	MUCM Mean	MUCM St.Dev.	MUSD Mean	MUSD St.Dev.	Z	Sign. (exact)
Perceived usefulness (PU)	3,68	0,70	4,05	0,67	-3,05	p = .002
Perceived ease of use (PEOU)	3,26	0,76	4,06	0,59	-4,44	p = .000
Intention to use (ITU)	3,07	0,82	3,80	0,72	-4,16	p = .000
Average (AVE)	3,34	0,65	3,97	0,59	-4,25	p = .000

Table 8. Effect and sample sizes for the significant differences in perception

TAM variable	Perception				
	Pooled st. dev.	Effect size	Cohen	Hopkins	Sample size required
Perceived usefulness (PU)	0,677	-0,55	medium	small	108
Perceived ease of use (PEOU)	0,672	-1,19	large	large	23
Intention to use (ITU)	0,762	-0,96	large	moderate	35
Average (AVE)	0,613	-1,03	large	moderate	31

6 Discussion

6.1 Main Findings

The experimental comparison indicates that the two techniques are complementary in terms of understanding. They aid understanding of different aspects about intrusions into systems, i.e., the architecture (MUCM) and the sequence of events, actions (MUSD). They are equal in terms of performance, i.e., they encourage users to identify similar numbers of vulnerabilities and mitigations in the planned system, although further analysis is needed to investigate if they encourage identifying the same types of vulnerabilities and mitigations. However, MUSD is perceived more positively by users, i.e., they rate the technique more highly in terms of perceived usefulness, perceived ease of use and intention to use. The difference is more marked for perceived ease of use. A summary on the decisions about the hypotheses can be seen in Table 9.

It must be admitted, of course, that the results of this experiments are not particularly surprising. MUSD was better for attack sequence knowledge, and MUCM for architectural knowledge, just as suspected. As for the participants' preference towards MUSD, this may plausibly be explained by the fact that the students were previously familiar with sequence diagrams, but not with use case maps, hence MUSD may have required a smaller learning effort than MUCM. Although the result was in many ways as expected, it still feels useful to have such suspicions confirmed by controlled experiments rather than letting them remain on the purely speculative level.

Table 9. Results of hypothesis testing (A = accept, R = reject)

$H1_1$	$H2_1$	$H3_1$	$H4_1$	$H5_1$	$H6_1$	$H7_1$	$H8_1$	$H9_1$	$H10_1$	$H11_1$
A	A	R	R	R	R	R	A	A	A	A

6.2 Implication for the MUSD Technique

Our intention is to combine MUSD with other related techniques to provide complete intrusion models. As indicated, MUCM could reflect the architecture related aspects and MUSD the order of the steps. In the future, it would be interesting to see how MUSD could be used along with other attack modelling techniques.

Since UML sequence diagrams (SD) are well known, it would be easy to adapt and use MUSD in system development projects. This is supported by the results of the experiment, indicating a positive perception towards MUSD. Further investigation is needed to explore how MUSD can utilize various SD notations like Decomposition and InteractionUse to improve the modelling.

6.3 Threats to Validity

In [21] conclusion, internal, construct and external validity are suggested as relevant categories for the validity of experiments. *Conclusion validity* concerns the relationship between the technique used and the outcome in terms of scores for tests and post-task questionnaire. One important question is whether the sample size is big enough to justify the conclusions drawn. The main effect claimed about understanding was a significant advantage for MUCM regarding architecture related statements and for MUSD regarding event sequence related statements. MUSD was also significantly more favoured than MUCM regarding PU, PEOU, ITU and their averages.

Denoting the Type I error probability by α and the Type II error probability by β, the following relationship holds:

$$N = \frac{4(u_{\alpha/2} + u_{\beta})^2}{ES^2} \ .$$

If we use $\alpha = 0.05$ and $\beta = 0.20$, we get $N = 32/(ES)^2$ [21] as a required sample size or $ES = \sqrt{32/N}$ as required effect size. The results are presented in Table 5 and 8.

Hence, we have sufficient observations only for PEOU, ITU and the averages (AVE). With the actual sample size of 42 we would have needed an ES of at least 0.873 for the three other cases. We note that these effect sizes must be used with caution, because our data are not in general normally distributed.

Internal validity assesses whether the observed outcomes were due to the treatment or to other factors. To avoid the threat of selection bias, Latin-Squares experimental design was used where all participants tried both techniques. Moreover, the participants were randomly assigned to the four groups and a pre-experiment questionnaire was used to control for confounding factors. The Latin-Squares design eliminated potential problems with learning effects, boredom or fatigue as the participants tried the techniques in different orders. Furthermore, all subjects were in the same auditorium with equal working conditions, and sitting far apart with no interaction between them.

Construct validity concerns whether it is legitimate to infer from the measures made in the experiment to the theoretical constructs that one was trying to observe. With respect to the performance data, we were trying to observe the understanding and problem solving effectiveness achieved using the respective techniques, and this was measured by scores on 20 true/false statements and the number of vulnerabilities and mitigations identified. Of course, there are other ways to explore understanding or effectiveness, but the ability to answer questions about a case should be a reasonable approximation of understanding, and the identification of vulnerabilities and mitigations would be relevant problem solving tasks in secure software engineering.

External validity is concerned with the question of whether it is possible to generalize from the experimental setting to other situations, most importantly to industrial systems development. The use of students instead of practitioners is a notable threat but as observed in [22], the level of competence is more relevant for performance in an experiment than whether the person is student or practitioner. Our participants were soon to finish their second year of computing studies. They had learnt about system modelling techniques (e.g. sequence diagrams). Moreover, the task in this experiment was to learn two new methods and apply them on a task explained during the experiment – the difference between students and practitioners would probably have been bigger if the task was about using a method well-known to practitioners.

7 Conclusions and Further Work

The paper has presented misuse sequence diagrams (MUSD) for visualizing system intrusions and compared it with a similar technique, MUCM, through a controlled experiment. The results indicate that MUSD and MUCM are complementary techniques having their strengths on visualizing temporal sequence of actions and architectural issues respectively. Experimental comparison between MUSD and other approaches not invented by the authors are planned for the future.

Further work is needed to improve MUSD for better intrusion modelling. For example, MUSD could be extended to incorporate UML sequence diagram features like InteractionUse and Decomposition. It is also needed to look into how MUSD could be combined with other techniques to provide a complete picture of intrusions. In particular, this experiment indicates that it would be interesting to run future experiments where MUSD and MUCM are used together instead of as competitors, to see if this gives better results than using either technique alone. It would be also interesting to see whether MUSD can be extended to model dependability issues other than security and to evaluate it also through larger industrial case studies, since controlled experiments can only investigate tasks of a fairly limited size.

References

1. Aagedal, J.Ø., et al.: Model-based Risk Assessment to Improve Enterprise Security. In: Proceedings of the Sixth International Enterprise Distributed Object Computing Conference (EDOC 2002). IEEE, Los Alamitos (2002)
2. Mitnick, K.D., Simon, W.L.: The Art of Intrusion. Wiley Publishing Inc., Chichester (2006)

3. Schneier, B.: Secrets and Lies: Digital Security in a Networked World. Wiley, Chichester (2000)
4. Sindre, G., Opdahl, A.L.: Eliciting Security Requirements with Misuse Cases. Requirements Engineering 10(1), 34–44 (2005)
5. Karpati, P., Sindre, G., Opdahl, A.L.: Illustrating Cyber Attacks with Misuse Case Maps. Accepted to 16th International Working Conference on Requirements Engineering: Foundation for Software Quality, RefsQ 2010 (2010)
6. Karpati, P., Opdahl, A.L., Sindre, G.: Experimental evaluation of misuse case maps for eliciting security requirements. Submitted to 18th IEEE International Conference on Requirements Engineering, RE 2010 (2010)
7. Unified Modeling Language, `http://www.uml.org` (accessed 4.6.2010)
8. Internet Security Glossary,`http://www.apps.ietf.org/rfc/rfc2828.html` (accessed 22.6.2010)
9. Opdahl, A.L., Sindre, G.: Experimental comparison of attack trees and misuse cases for security threat identification. Information and Software Technology 51(5), 916–932 (2009)
10. Buhr, R.J.A.: Use Case Maps: A New Model to Bridge the Gap Between Requirements and Detailed Design. In: 11th Annual ACM Conference on Object-Oriented Programming Systems, Languages and Applications (OOPSLA 1995), Real Time Workshop, p. 4 (1995)
11. Markose, S., Xiaoqing, L., McMillin, B.: A Systematic Framework for Structured Object-Oriented Security Requirements Analysis in Embedded Systems. In: IEEE/IFIP International Conference on Embedded and Ubiquitous Computing, vol. 1, pp. 75–81 (2008)
12. Georg, G., Ray, I., Anastasakis, K., Bordbar, B., Toahchoodee, M., Houmb, S.H.: An aspect-oriented methodology for designing secure applications. Information and Software Technology 51, 846–864 (2009)
13. Redmill, F., Chudleigh, M., Catmur, J.: Hazop and software Hazop. Wiley, Chichester (1999)
14. IEC 61025: Fault tree analysis (FTA), IEC Standard (2006)
15. Runde, R.K., Haugen, Ø., Stølen, K.: The Pragmatics of STAIRS, Research Report 349 (January 2007)
16. Jürjens, J.: Secure Systems Development with UML. Springer, Heidelberg (2005)
17. Karpati, P.: `http://www.idi.ntnu.no/~kpeter/ExampleSheet_Group1.pdf` (accessed 24.6.2010)
18. Davis, F.D.: Perceived usefulness, perceived ease of use and user acceptance of information technology. MIS Quarterly 13, 319–340 (1989)
19. Cohen, J.: Statistical power analysis for the behavioral sciences, 2nd edn. Lawrence Erlbaum, New Jersey (1988)
20. Hopkins, W.G.: A New View of Statistics. University of Queensland, Brisbane (2001)
21. Wohlin, C., Runeson, P., Höst, M., Ohlsson, M.C., Regnell, B., Wesslén, A.: Experimentation in Software Engineering: An Introduction. Kluwer Academic, Norwell (2000)
22. Arisholm, E., Sjøberg, D.I.K.: Evaluating the effect of a delegated versus centralized control style on the maintainability of object-oriented software. IEEE Transactions on Software Engineering 30, 521–534 (2004)
23. Venkatesh, V., Morris, M.G., Davis, G.B., Davis, F.D.: User acceptance of information technology: Toward a unified view. MIS Quarterly 27(3), 425–478 (2003)