Towards Operational Measures of Computer Security: Experimentation and Modelling

Tomas Olovsson
Erland Jonsson
Department of Computer Engineering
Chalmers University of Technology
S-412 96 Göteborg
SWEDEN

olovsson/jonsson@ce.chalmers.se

Sarah Brocklehurst
Bev Littlewood
Centre for Software Reliability
City University
London EC1V 0HB
England

s.brocklehurst/b.littlewoood@csr.city.ac.uk

Abstract.

This paper discusses similarities between reliability and security with the intention of finding probabilistic measures of *operational security* similar to those that we have for reliability of systems. Ideally, a measure of the security of a system should capture quantitatively the intuitive notion of 'the ability of the system to resist attack', described by the parameter *effort*.That is, it should reflect the degree to which the system can be expected to remain free of security breaches under particular conditions of operation, including those of attack. Current security *levels*, e.g., those of the Orange book, at best reflect the extensiveness of safeguards introduced during the design and development of a system. Even though we might expect a system developed to a higher level than another system to exhibit 'more secure behaviour' in operation, this cannot be guaranteed. In particular, we can not assess the actual operational security from knowledge of such a level.

We have carried out two realistic intrusion experiments intended to investigate the empirical issues that arise from this probabilistic view of security assessment. More specifically, they investigated the problems of measuring *effort* and *reward* associated with security attacks and breaches. In the first, *pilot experiment*, the intention was to see whether experiments of this type, in which a number of under-graduate students were allowed to attack a system under controlled circumstances, were at all feasible, and if so, to get valuable information on how they should be carried out. In the second *full-scale experiment*, we aimed at getting enough data to be able to start a methodology development, a methodology by which operational security measures could be derived. During this latter experiment 181 activity reports were submitted, resulting in 63 successful breaches, and reflecting a total expenditure of 594 man-hours. The breaches were classified into 6 different categories, based on which kind if security flaw was exploited and the underlying functionality and nature of these flaws is discussed. In a short concluding discussion on quantitative assessment, it is recognized that, even if effort is meant to be composed of many different parameters, various time parameters, such as working time, on-line time and CPU time, form an important base for the measure.

Table Of Contents here

1. Introduction

Our overall aim in this work is to develop a quantitative theory of operational security, similar to that which now exists for reliability (for hardware and, more recently, for software). The work presented in this paper, however, concentrates on the more limited objective of finding appropriate methods for collection of data, by means of which the operational security of a system could be measured. Here, an immediate parallel would be that of reliability modelling. Current approaches to measuring and predicting system reliability are based on the definition of reliability on terms of probability of failure-free operation for a specified legth of time [Littlewood 1989].The advantage of this operational approach is that it allows us to obtain measures of the actual *behaviour* of a system, as seen by the user, rather than merely of some static properties. Users are likely to be more interested in knowing the reliability of their system, than in knowing that it possesses certain structural properties and attributes, or that it was developed under a particular regime. These static properties of the system and its development process, undoubtedly, do influence the user-perceived operational reliability, but they are not sufficient in themselves to determine this reliability.

The present position in security seems to be that current ‘measures’, or rankings, of security of systems are merely analogous to the static properties and attributes discussed above. The security levels of various Security Evaluation Criteria, such as the European ITSEC [ITSEC 1991] and the American Orange Book [NSCS 1985], for example, are based on such static factors. It is clear that those factors will influence operational security, and they may very well be beneficial in producing secure systems, but they do not facilitate quantitative *evaluation* of the actual achieved operational security. Indeed, it is not even possible to know whether a particular system with a higher security level in such a scheme is, in some truly operational sense, more secure than another system with a lower security level.

Thus, it is evident that a more appropriate measure of security is needed, and the work on the theoretical side would be to try to make a rigourous definition of such a measure. This presents considerable difficulties, which seem to be much more serious than is the case for reliability, and even a superficial analysis leaves you with several open questions. Therefore, it has been judged beneficial to launch parallel empirical investigations. Thus, we have carried out a series a of practical intrusion experiments, the two first of which are presented in this paper. In the first, the pilot experiment [Olovsson et al. 1993], the intention was to see whether experiments of this type were at all feasible, and if so, how they should be carried out. In the second experiment, data on the intrusion process was gathered, data that is aimed to reflect the *effort* expended by the intruder in trying to make a breach into the system [Littlewood et al 1994]. The intention is that this data should be used, in the short run for the methodology development, and in the long run as a data bank for the actual calculation of statistical operational measures. A compilation of this data is found in [Olovsson and Jonsson 1995].

There does not seem to be a large literature on probabilistic treatment of system security. Lee [Lee 1989] proposes to model levels of security in terms of the probability distribution of the ‘level of threat’ required to achieve a penetration involving information of a given classification. Thus $r_{c1,c2}(t)$ denotes the probability that a system at level c_1 will allow a breach involving information of a given classification at level c_2 when the system is subjected to a level of threat *t*. Hierarchical relationships between security levels can then be represented in the obvious way by inequalities between these functions. He also introduces probability distributions for the amounts of damage resulting from various illegal transfers of information. He claims that various conclusions can be drawn from this kind of model concerning the greatest security risks and the rules of combination for system security levels. Denning [Denning 1987] considers a statistical approach to detecting when the operational environment switches from ‘non-threatening’ to ‘threatening’. Bishop [Bishop 1989] suggests, in what he calls a ‘common sense security model’, that it is possible to empirically assign some of the probabilities discussed in this paper, but gives no details.

It is also worth noting that up to now security work seems to have concentrated to a large extent on the problems arising in circumstances where very high security is demanded. Certainly it appears to be the case that some of the major funding agencies for security research are concerned with issues of national security. Problems of this kind in security are similar to the problems of safety-critical systems in reliability, where very high levels of reliability often need to be assured. The work on ultra-high reliability evaluation, however, has so far been notably unsuccessful, at least for software and design reliability [Littlewood 1991], in comparison with the achievements in the modelling and evaluation of more modest levels [Brocklehurst and Littlewood 1992]. This lesson suggests that, in these initial stages of attempting to model operational security, we should restrict ourselves to systems for which the security requirements are also relatively modest. It is only for such systems that sufficient numbers of breaches could be observed for the empirical studies that seem necessary.

Section 2 deals with some further resemblances and differences between reliability and security. In section 3 we discuss the effort and reward parameters of our operational security model, and their influence on the breach process. Section 4 gives the practical conditions for the experiments, defining the actors and the reporting. Section 5 and 6 gives details of the pilot and full-scale experiments respectively, and section 7 discusses to what extent the results of these rather limited experiments could be used for quantitative modelling.

2. The Reliability Analogy

In reliability, the important point is that we express our confidence in a system probabilistically in terms that reflect naturally its ability to operate successfully. In particular, there is an acceptance that there is a suitable ‘time’ variable, and it is assumed that failures in time occur in a random process (albeit perhaps a complex one). This natural randomness of the failure process arises in reliability for several reasons. In the case of hardware reliability it is due to the complexity and unpredictability of the involved circuits and the operational profiles, as well as from the operational environment. In the case of software reliability, there is an inherent determinism of the errors in the programs, since all of them are design errors. In this case, the randomness comes from the natural unpredictability of the operational environment.

In the reliability context, the input space is the totality of all inputs that might ever be encountered. Thus for a process-control system it might be a many-dimensional space representing all the possible settings of sensors and controls, together with all possible machine states. We can informally think of a fault as represented by a subset of this space: when an input is selected from the subset a failure will result. The input space is typically very large and complex, and we never know with certainty which inputs will be selected in the future. Neither do we know which inputs will, when selected, trigger a design fault and so result in a failure. We take account of this natural uncertainty in probabilistic measures of reliability such as the *reliability function* (the probability of failure-free operation for a specified time *t*), the *rate of occurrence of failures* (ROCOF), the *mean time to next failure*, etc. In the security context the input space is again the set of all possible ‘inputs’ to the system, including both those involved in normal operational use and those which result from *intentional attacks* upon the system.

Thus, reliability theory can be thought of as the description of processes involving failure events taking place in time. In the case of security, a naive view would be that the process of security breaches could be modelled likewise. However, even if there are some parallels, there are also important distinctions that would require a different approach. In particular, it is quite clear that time is not an appropriate variable for the intrusion process, as discussed below. Also, it is a matter of discussion what is the most adequate correspondance to a security breach. The classical notion of a security breach as a security ‘failure event’ leads to some further problems as regards security modelling. It could therefore be discussed, if a breach could just as well be regarded as a security ‘fault’ [Jonsson and Olovsson 1994]. Perhaps the solution to this problem lies in a totally new approach to the composite area of reliability and security [Jonsson and Olovsson 1992].

3. Effort and Rewards

3.1 Effort

In the case of security, time alone is not an appropriate variable. In particular, elapsed time when a system is not under attack is irrelevant for the evaluation of resistance to deliberate attacks. Instead, we need to consider a variable which captures the effort expended in attacking the system, and informally we expect a system which requires more *effort* to be expended until it is successfully breached, to be ‘ more secure’. We might with these terms define operational security analogous to reliability: the security function would be the probability of surviving without a security breach for the expenditure of a given effort by an attacker; other measures such as the rate of security breaches per unit effort, mean effort to the next security breach, etc, have obvious meanings.

Clearly, different systems will vary in their ability to resist a particular expenditure of effort from an attacker. Factors influencing this resistance include how the system is configured, the presence of security enhancing mechanisms, the quality of the system design and how the system is operated. The idea is that this ‘ability to resist attacks’ can be estimated by means of measuring the effort expended in order to achieve a breach. A complications is that effort is a variable composed of several factors: the attacker’s education, skill and experience as well as time, money and other resources spent by the attacker are important factors. It is important that any differences between the ability of different attackers to break into the system is captured in the effort variable. It should capture the intuitive notion that the more effort is invested in attacking the system, the greater is the chance of achieving a breach. The major motivation for the experimentation described in this paper, is to find out whether it is possible to devise an effort measure with these properties.

3.2 Breaches and Rewards

The classical notion of a security breach as a security ‘failure event’ is sometimes too simplistic for our purposes. Firstly, it is clear that the value of different security breaches to a single attacker may vary enormously and this notion of reward should be incorporated into the measures. Secondly, whilst an attacker of a system may certainly gain something of value from these breach events, they may also acquire reward continuously - for example by gradually learning about the system. For these reasons we believe that, instead of the failure events in time that characterise the reliability process, the most general description of the security process would be in terms of a reward process, involving both discrete and continuous increments, against effort expended.

In our experiments we want to collect empirical data on breach events in order to learn about the nature of the reward process in security. Unfortunately, there are various practical limitations to what we could ever expect to achieve here. Although it would be desirable to include all types of breach events, some cannot be allowed in an experiment in which a real system is used due to the possible consequences for other users of the system.

The reward an attacker would get from breaking into a system determines his/her motivation and affects whether he/she is willing to spend the effort that is needed in order to perform the attack. Examples of such rewards are personal satisfaction, gain of money, revenge or simply pure curiosity. A reward may also be negative. An example of a negative reward could be the consequences of detection.

Considerations about the reward processes suggests that each attacker has a subjective view of his/her rewards, which is in general different from the system owner's view of these same rewards. For example, the loss due to a breach event for a system owner is likely to be different from the reward to the attacker. Thus, the system owner would normally only attach *negative* reward, i.e. actual loss, to (illegal) attacker activity. Furthermore, the subjective view of an attacker may be different from other attackers' views in similar circumstances. However, each attacker may be expected to apportion their effort optimally according to their view of their (potential) rewards, while the owner may be expected to respond to attacker activity in accordance with his view of his potential losses.

Note that the attacker may also attach some reward to legal activity, e.g. activity that allows the attacker to learn something about the system that would be useful in other, illegal, attacks.

4. Conditions for the Experimentation

Two experiments have been conducted: a pilot experiment and a first full-scale experiment. This section defines the common conditions under which they were performed: the actors, the system and what we hoped to achieve. The differences between the experiments are referred to under each separate paragraph.

There are three different kinds of actor involved in the experimentation: *the Attackers*, *the System administrator* and *the Coordinator*. Each of the actors plays a different role in the experiment and with respect to the target *System*. Furthermore, each actor is subject to a set of rules and restrictions, as well as a desirable behaviour. These rules and restrictions are discussed below.

4.1 The System

The target system was the same for both experiments: a set of 24 SUN ELC diskless workstations (22 in the pilot) connected to one file-server, all running SunOS 4.1.2. The system was during this time in operational use for laboratory courses taken by undergraduate students at the Department of Computer Engineering at Chalmers. The attackers were legal users of the system and were given normal user privileges and had physical access to all workstations except the file server. The system itself was configured as a 'standard' configuration as specified by the supplier and supervised by an experienced system administrator. No special security improvements were introduced for the experiment, and therefore the system presented only a modest level of security. The system had all standard monitoring and accounting features enabled in order to allow us to monitor the activities on each user account and to measure the resources each attacker spent during the breach process.

It should be emphasised that the choice of system was not the result of any specific preference. It was rather due to the fact that this was the system that was readily available for laboratory use. However, we realised that it was not a disadvantage that the type of system that we had available was rather common, and could therefore be assumed to be quite representative of a 'normal' system. Also, it is our intention to conduct experiments on different systems in the future to avoid general conclusions being drawn from system-specific results.

4.2 The Attackers

Attacker profile. A non-trivial problem was to find potential attackers. We were aiming for attackers that either already were or at least in the future would become the 'normal' users of the system, i.e. users without any special knowledge of operating system details and security issues. It is important to note that we *did not want professional crackers* who already knew about most weaknesses in the system. Professional crackers would give us information about where and how our particular system needed to be tightened. Such experiments or investigations have indeed been performed [Attanasio et al 1976], but from a modelling point of view, it is essential that the attackers were representative of a more general distribution of users, and using 'security experts' would radically reduce the value and representativeness of data.

A possible and attractive solution turned out to be using university students. Firstly, they were a good approximation of 'normal users' since they would very soon be working as such in industry, and secondly, we believed that it would not be a problem to convince university students to participate in such an experiment.

At the same time, when starting up the pilot experiment we were unsure of to the extent to which the students would really succeed in breaking into the system. An experimental result with no or very few breaches would definitely be an experimental failure. Even if this would be a good result from a system security point of view, it would not give us any data, nor would it give much information of how to perform such experimentation. To reduce this risk we decided to use only members of the Chalmers University Computer Club for the pilot experiment. The rationale for this decision was that since they had a special interest in computers, it was quite probable that they should be successful in this task. And conversely, if they did not succeed, 'regular' students would be unlikely to succeed. In the full-scale experiment we would then, depending on the outcome of the pilot, be prepared to continue with 'regular' students.

Rules for the Attackers. The attackers were told (rather informally) that a security breach occurs *whenever they succeed in doing something they were not normally allowed to do*, for example to read or modify a protected file, to use another user's account or to falsify or disturb normal system function. In general, it was our intention to restrict attacker action as little as possible. Despite this it was necessary to pose some restriction on their activity. One obvious restriction was that the attackers (or, in the case of the second experiment, the attacker *teams*) were forbidden to cooperate with others, since this would impair the collected data. Also, for obvious reasons, they were not allowed to cause physical damage to the system or to tamper with the hardware in order to reduce system availability. Finally they were told that activities that could cause disruption of system service to the normal users to the system had to be performed under supervised control, which in general meant that the activity was either performed during evenings or performed with the presence of an experiment coordinator.

It was essential for the attackers to be given a general description of the overall objectives of the experiment so that they had a complete understanding of why the rules should be obeyed, and why and in what way they should report their actions. Afterwards, it turned out that this requirement really motivated the attackers to work and follow the rules, and it become a major factor for the success of these experiments.

4.3 The Coordinator

The coordinator’s role was to monitor and coordinate all activities during the experiment. In particular he had to make sure that the attackers and the system administrator were complying with the experimental rules. This meant that, as much as possible, the coordinator continually monitored all activity throughout the experiment and that he was normally available for consultation by the attackers and the system administrator. He was also to make sure that the activity of attackers would not interfere with each other.

4.4 The System Administrator

The system administrator was supposed to behave as realistically as possible. He was aware of the fact that the experiment was taking place, but he would monitor the system in the usual way and not intensify his search for security violations or other unwanted user behaviour (the extended logging imposed on the system was for data collection use only). Also, whenever he found a security violation on the system, he should contact the Coordinator who recorded the observation. Thus, the system administrator reported all attacking behaviour he observed and all security related actions he performed during the experiment.

4.5 Anticipated Results

The ultimate goal for these experiments was to collect data for modelling operational security. Some questions were especially interesting to find answers to:

- Can we derive our single quantified measure of effort expended by an attacker, unifying the different factors (time, expertise, etc)?
- What is the distribution of the effort that needs to be expended to achieve a breach, for a single attacker?
- What is the nature of the more general stochastic process of reward from the attackers’ and owner’s views (i.e. when rewards can be continuously accrued as well as being associated with discrete breach events).

It should be noted that some of these problems would require extensive experimentation and innovative new approaches before even an approximate result could be anticipated. A more extensive discussion of modelling issues can be found in the previous chapter of this book.

In addition to the issues related to modelling (i.e., security evaluation) we hope to make some interesting qualitative observations. It is anticipated that answers, or at least partial answers, to these questions should be possible to find:

· How hard is it for regular users to break into their own (Unix) system? Will they be able to break in at all?

· Would it be possible for other users to work on the attacked system during the experiment? (A preferred situation from a modelling point of view, but also preferred since the available system was in operational use.)

· How do regular users approach such a task, i.e., what do they do when they are asked to break into their own system?

· Are only previously known vulnerabilities in the system exploited by these attackers or are new ones found?

· What kind of security breaches occur? Is the first security breach followed by new independent breaches, or is it more likely that the first breach is followed by new breaches where the knowledge collected from earlier breaches is used?

· How many and what types of security breaches were (or could have been) detected by the system owner? How many and what types of breaches could have been detected if we added non-standard hardware or software?

· How many of the attacks and breaches could have been carried out by attackers that were not users on the system?

· Is it an advantage for attackers to cooperate, for example to work in groups of two? Will two cooperating attackers produce more security breaches than two independently working attackers? Cooperation may favour sharing of knowledge and resources at the expense of lack of diversity.

We hoped that if the type of experimentation described here turned out to present a practicable way towards quantitative modelling of security, we would continue with further experiments which would clarify and give even more detailed answers to these questions regarding quantitative modelling. Also, hopefully future such experiments could be the basis for comparing systems with respect to security properties and could give system owners a possible idea, i.e. a measurement, of how secure their systems are.

4.6 Reporting

In addition to the automatic logging and recording of data, the attackers were required to perform extensive reporting. The reason for this was twofold: firstly, much of the information of interest was not available for automatic recording and secondly, we wanted to be able to compare results from automatic and manual reporting to be able to estimate the magnitude of mis-reporting, i.e., how erroneous the reporting process was. There were three principal manual reports: the *background report*, the *attack*- and *breach report* (which in the full-scale experiment were combined into a single *activity report*), and an *evaluation report*.

The *background report* was submitted before the experiment started. In this the attackers had to document their background (formal education, prior experience with computers and computer security, etc.) together with their interest and motivation for participating in the experiment. They were also asked to estimate their knowledge in computer security compared to all other last year students.

Each *activity report* contained data for one specific activity such as working-time, whether spent at the computer or elsewhere, on-line time, used resources (e.g. books, manuals, other computers, etc) and when the activity took place. Here the attacker would also note the motivations for his activity as well as other observations. If they managed to perform a security breach or gained some other sort of substantial reward, they had to give a full description of this, as well.

When the experiment was completed, the attackers had to fill in another questionnaire anonymously, the *evaluation report*. The goal of this report was to get some cross-checking of the experimental results. For example, they were asked in how many attacks they got help or hints from other attacking groups, which was something that was against the rules.

Also, in the very end, the attackers were asked to write a *final report* where each attacker described his/her activities and results quite freely. This report was delivered a few weeks after finalization of the experiment and served as the attackers' personal evaluation and reporting of their work. The final report turned out to be a very helpful means for checking the contents of the other reports.

5. The Pilot Experiment

5.1 Goal

The first experiment, the pilot, was conducted during spring 1993. The purpose was to investigate *if* it was at all possible to collect data for quantitative security modelling in the proposed way, and in such a case to find out *how* the data collection experimentation should appropriately be organised. Examples of open issues at this time were that of attacker behaviour and success (*could* they break in?), and their ability and willingness to report correctly.

5.2 The Attackers and their Motivation

In the pilot experiment we decided to use members from Chalmers University Computer Club, i.e. students who were expected to be more skilled than regular students. The reason for this choice was to increase the probability of successful breaches.

In order to encourage the attackers to try to make breaches (and not to spend as much time as possible with the project) and to adhere to the rules of the experiment, two types of reward were given. Firstly, the most successful attacker was to receive a small personal 'medal' type gratitude. We hoped that this would motivate the attackers to work in the 'desired' way, where being successful would not necessarily mean 'performing many breaches', but rather to come up with innovative ideas and to document his actions in a purposeful way. Secondly, a small gift was handed over to the Chalmers Computer Club, a gift that could be used by all members of the Club and which would be our appreciation for the help we received.

5.3 Results from the Pilot Experiment

In the beginning of the experiment, we had 13 active attackers working independently. However, during the experiment the number of active attackers decreased, mainly due to lack of motivation. The 13 attackers submitted 37 attack reports showing roughly 50 hours of expended working time. However, later investigations showed that this figure was grossly underestimated and the real working time was more than twice as high. The expended working time was unevenly distributed among the different attackers: the 3 most active attackers accounted for more than 85% of the reported time. The automatic accounting system recorded 73 000 commands which were executed by the attackers, out of a total of 800 000 commands, indicating that the attackers accounted for less than 10% of the total system activity during this time.

The attackers' reports contained 25 successful attacks, showing that it was indeed possible for students to break into a Unix system. Most breaches were 'standard' breaches in the sense that they were already reported in literature or available over Internet. Broadly speaking they go into three different groups (for a further discussion of some of these methods, see the following discussion of the full-scale experiment):

- achieving root privileges by means of a single-user boot-up, possibly followed by using the SUID-mechanism to transfer root privileges to the server.
- finding out user passwords using the *Crack* program, i.e., a dictionary attack.
- using the Xkey snooping program to monitor the key strokes of other users, thus hoping to get passwords or other interesting information.
- The use of security-enhancing program such as *Cops* to find vulnerabilities.
- There were also some less standard attempts made:
- one attacker found out that the screen devices for some workstations were default readable (and writable), meaning that it was possible to monitor all the output on the screen.
- one attacker planted a Trojan Horse for other users in a faked 'ls'-command.

A detailed description of breaches and effort expended, can be found in [Brocklehurst et al 1994]. It should be noted that there are remedies for many of these attacks. However, the system administrator was either not aware of the vulnerability or, in a few cases, a vendor-provided solution was received but had not yet been installed.

5.4 Conclusions of the Pilot Experiment

The amount of data received from this pilot experiment was, as expected, too sparse to allow any statistical modelling, but the experiment showed that it should be possible to conduct a full scale experiment that would yield real data. There were several important conclusions drawn from this experiment:

- students *are indeed able to break into a standard Unix system*, even though they were given such a short time for the task. Several of them performed many different breaches.
- students *can* be used in this kind of experiment. They were quite interested in participating in the experiment and they showed a remarkable understanding of and compliance to the experimental rules.
- the system *will* be operable to other users at the same time.
- However, attackers had a tendency to leave the experiment due to lack of ideas and/or motivation. This emphasised the importance of motivation for the attackers in future experiments.

- Also, the required reporting turned out to be too extensive. Many of the questions, e.g. regarding personal reward of a breach, were answered with insufficient detail or not at all. It also became evident that the understanding of what constituted an ’attack’ and ’breach’ differed between the attackers, which led to incomplete reporting of attacker activities. Therefore, in the full-scale experiment, we reduced the number of questions and merged the attack report and the breach report into one single *activity report*, containing all information we wanted to know about an activity.

6. The Full-scale Experiment

6.1 Goal

Since the pilot experiment had showed the potential feasibility of the method as such, a full-scale experiment was performed. The goal of the full-scale experiment was to gather enough data for a quantitative modelling attempt. We hoped that we could use the data to develop a quantitative methodology, by means of which quantitative conclusions could be made. However, it should be emphasised that those conclusions would *not be representative* for a more general class of systems, attackers or environments. The statistical basis would be far too limited for any such generalisations.

Further, we hoped to learn more about the attacking process and system vulnerabilities.

6.2 The Attackers and their Motivation

The full-scale experiment was conducted during a 4 week period in November and December 1993. As a result of the successful outcome, in terms of breaches, of the pilot experiment, we engaged ’regular’ last year students as attackers in the second experiment, and offered them to do a project work within a course in Applied Computer Security. The conditions for this experiment were similar to those for the pilot experiment, except that one security improvement had been carried out: the system hardware had been equipped with a password to prevent users performing a single user boot-up sequence. Furthermore, the reporting system was simplified as described above.

This time, the students were grouped into groups of two, with the intention to increase the motivation and the persistence of the attacking process. There were 24 attackers (i.e. 12 groups) participating in the experiment and we expected each group to spend around 40 hours of effective working time during a 4 week calendar period. However, there was no absolute requirement to spend exactly this amount of time. Their goal was to create as many and as valuable security breaches as possible, not to sit down and wait for 40 hours to pass by. The attackers were supposed to meet with the Coordinator twice a week and discuss their progress.

The major motivation of the attackers was that the experiment was a compulsory part of the course they were taking, and that well performed work, including a good final report, could result in a higher mark on the course. We also found that most attackers were genuinely interested in learning more about security by testing and attacking a real system and that they were interested in discussing their results long after the experiment had ended.

6.3 Some Numerical Results

During the full-scale experiment we received 181 activity reports describing a total of 63 security breaches of various kinds. The number of breaches is an approximate figure, since the limit of what to consider to be a security breach or not, is a matter of definition (this problem is further discussed below). The groups reported 481 hours of working time, i.e. time where at least one group member was active, and altogether 594 man-hours were spent, which is an average working time of 50 hours per group.

An informal interpretation of these figures are that an “average” computer science student can perform one security breach every seventh hour, on an average. However, there were great variations between students, and between the breach rates early in their activity and those later on.

Furthermore, 281 hours of on-line time at the target system and 65 hours on other systems was reported. A comparison with the automatic logging shows that the reported numbers are somewhat underestimated (35 hours less), but the discrepancy is much lower that in the pilot experiment, a fact that can probably be accredited to the improved reporting system.

6.4 Types of Breaches

The experiment resulted in 63 breaches. We have divided these breaches into five categories, mainly depending on what kind of vulnerability they exploited. The definition of each category and examples of breaches are given in the following. The table below shows a summary of breaches and attempted breaches per category. Note that these figures are only valid given the present tentative classification and definition of breach and breach attempt.

Table 1: Number of breaches per category

Category	Attempts	Breaches
Execution of security enhancing programs	22	18
Spoofing programs with SUID privileges	14	8
OS and administration related problems	19	18
User related problems	11	8
Snooping	9	6
Other	many	5

Execution of security enhancing programs. The intended use of security enhancing programs is to help the system administrator to maintain security in his system, and many of them are freely available over the Internet. These programs look for specific or general (potential) vulnerabilities in the system and output a list of warnings. The administrator is supposed to run those programs on a regular basis and take proper action. However, in many cases this is not done regularly, which means that an intruder has an automatic tool for finding vulnerabilities.

Examples of used programs are *Crack*, *Cops*, *Tiger* and *ISS* (*Internet Security Scanner*). Crack is a password guessing program performing a so-called dictionary attack. It exploits the fact that users have a tendency not to chose random passwords, but passwords that could be found in a dictionary or encyclopedia, possibly with minor modifications. Cops and Tiger scans the system searching for a number of different potential problems. ISS does the same thing but from the ’outside’ of the system.

These programs revealed several vulnerabilities in the system: a number of accounts with badly selected passwords, files with insufficient protection, incorrect system configuration, etc. In some cases, these programs were modified to exploit some specific characteristic. A good example of this is the modification of the crack program to look especially for passwords with a special distribution of consonants and vowels. In this way the computer generated passwords given to all students working in the laboratory exposed a greater risk to be guessed.

Spoofing programs with root privileges. Many Unix programs are executed with root (super-user) privileges, which are often necessary to accomplish the function of the program. This is implemented by using the SUID (Set User ID on execution) facility in Unix, which is a service that permits the user of a program to run it *with the privileges of the owner* of the program, instead of the users' own privileges. If the owner is root (the system administrator), a user can execute the program with the system administrator's rights. Normally, such SUID rights are only given to a few trusted programs. However, in some cases these programs contains bugs so that they can be spoofed to perform an unintended function, and this function may be selected so that a breach occurs.

Since this is a very general methodology for attacking a system, the breaches in this category are quite diverse. One example is a problem with the program Xterm, a virtual terminal program in the X-Windows system, which executes with root privileges. A facility of this program is that it can log all terminal output into a user-specified file. Due to a serialisation error in the procedure for checking this write privilege [Landwehr et al 1994], the output can be diverted to any file in the file system.

Another attack made use of the X-window server to remove a directory, even though it was not empty. The result was a number of 'lost' files and subdirectories that would continue to exist and occupy disk space, but yet could not be referenced. This breach provides a means to fill the available disk space of the system, whilst from a system administrator's view it appears to be almost empty. It can be characterised as an availability breach. It is possible for the system administrator to recover the lost files once he figures out what has happened, something that is not evident. (The administrator of the target system would later have done a restoration of the file-system from a back-up, which apart from a lot of work and making the system unavailable, could have resulted in lost user files.)

Operating system and administration related problems. Many problems were found which either correspond to bugs within the operating system or to bad management of the system. The target operating system had several files, directories and devices that by default were readable and writable! Some attackers discovered that a couple of such devices could be a suitable target for compromising system security. For example, one group found out that the system back-up tape for the previous week was installed and readable. In this way all information on the tape was unprotected and available to the attackers.

A couple of attacker groups exploited the insufficient checking of sender that 'sendmail' (the mail delivery program) performs. Thus, it is very simple to send email with a faked sender name. As a consequence, the source of an incoming email could never be trusted based on the contents of the "From-field" only. A similar bug affects the Internet News service. Still another vulnerability of this kind that was discovered is that a remote copy command (*rcp*) given to a device will crash the system.

Most problems in this category had already been warned about by CERT. However, one vulnerability found by the attackers was that the file */etc/utmp* was writable. This file contains login records for the current users of the system, and opened a possibility to introduce false information into the system, which could lead to a variety of breaches. This vulnerability was found and reported by CERT about 3 months after the experiment was finished (see APPENDIX A: Cert Advisory CA-94:06).

User related problems. This category includes all problems that are directly due to user action or lack of action. The target system was in daily use and more than 800 different student accounts existed. Therefore, with a high probability, some students would unintentionally leave files and directories readable and writable, when they should not have been. This opens up many possible attacks, e.g., just changing or deleting the contents of files or reading files that were supposed to be confidential. Another obvious attack is to use the unintended write permission to plant Trojan horses.

A quite unconventional and interesting and successful attack was carried out by one group, who sent a message to all normal users of the system (they got a special permission from the Coordinator to perform this attack.) The message claimed that all passwords had been revealed due to an operating system failure, and required all users to immediately change their passwords, and *to reply back informing about their new password*. The message was signed by something that could be interpreted as ‘System administrator’. Two users actually sent their password back (!) but a more important observation was that users begun to change their passwords from the machine generated (random) passwords, into simple, easily guessed passwords. This simple attack clearly shows the key role of the users in determining the security of a system.

Snooping. Snooping attacks aim at confidentiality. We have put two types of attacks in this category. The first one exploits the fact that the X-Windows server offers a service that permits other programs to ‘listen’ to events that occur, unless protected by an authentication file (*.Xauthority*). This makes it possible to use programs to record keystrokes, and thus collect confidential data.

The second one is a pure passive attack against the network connecting the workstations. These attacks are very hard to detect and could have been performed outside the lab as well. On the target system network, messages are sent in clear-text including passwords. Five groups realised that it must be possible to listen to the network and tried this approach. Three groups succeeded to filter out relevant data from the extensive network traffic. For example, one group found the root password, two user passwords and two passwords to external systems. The possible outcome of this attack is much greater than these passwords, since given enough time, they would have recorded all passwords ever sent on this network! Two other groups could listen to the network, but due to the extensive traffic did not manage to extract any sensitive data.

Other breaches. Finally, there are a number of breaches that do not naturally go into any of the above categories. The most interesting one, found by one group, is that the command *kbd_mode*, whose intended function is to reset the console keyboard, could be used remotely to (silently) disable another user’s keyboard.

6.5 The Attacking Process

How do the attackers approach their task? By far, the most common method is to make use of the Internet, which seems to be an almost inexhaustible source of information. In some cases books, manuals and journals were used. Thus, most breaches were ‘known’ on the Internet. The major exceptions were the *utmp* and the *kbd_mode* attacks, one of which was later reported. Example of information sources on the Internet are BBS’s, the News service and World Wide Web (WWW) pages. Furthermore, the Computer Emergency Response Team (CERT) is an organisation that monitors computer security and break-in activities, and sends out alerts and ‘fixes’ for security vulnerabilities. This experiment clearly stresses the importance of forcing system administrators to follow these recommendations.

6.6 Discussion

The experiment shows that there are many ways to enter a standard Unix system even for a regular user. However, since no extended security enhancements had been carried out on the target system, many of the vulnerabilities could have been removed. Some required an extended security ’kit’ to be installed, e.g. NIS+ which is available from some vendors. However, not many sites use them. If they had been used here, the number of successful attacks would have been smaller, but many loopholes would still remain. Furthermore, if the system were threatened by ‘professional’ attackers, they would probably penetrate it much more easily then did the students. Also, for many - even experienced - system administrators, it may not be evident how a vulnerability should indeed be removed. The attempt, before starting the full-scale experiment, to remove the possibility to gain *root* privileges by means of carrying through a single-user boot-up sequence clearly failed.

An overall conclusion from a practical security point of view is that sensitive information should not be stored on this kind of system, unless extensive security-enhancing precautions are taken.

7. Lessons for Quantitative Assessment

7.1 Measuring Effort

The pilot experiment provided valuable lessons which successfully allowed the second experiment to provide extensive data on ‘effort’. More events were observed in the second experiment and the recording of the data was much fuller and more complete. In addition, a flexible data-base structure is being built to allow easy investigation of some of the theoretical issues.

Much of the data collection was concerned with trying to establish the feasibility of identifying a single measure of effort. Most relevant and easily quantifiable was effort represented by three different (but associated) measures of *time*. The first time variable is the working time expended by the attacker, for example by learning about the system (on- or off-line), or by manually searching files for vulnerabilities.

Other time variables are associated with those attacks where the attacker executed some software, e.g. *Crack*. Here it was possible to measure the time that the attacker was using a machine, the *on-line time* as well as the *execution time (*or *CPU time)* of that machine on the attack(s). (In some cases the programs were executed on other systems than the target machines, in which case this possibility disappears.) Some attackers, for example, executed software on many machines in parallel and the use of these additional hardware resources can be taken into account by measuring the total execution time across the machines.

The three time measures together form a important component of our intuitive notion of effort. Obtaining a single measure of effort from these poses some difficulties that are currently being addressed as the data from the experiments is analysed. An obvious way forward is to take a simple linear combination of the three different times spent on an attack and treat this as the ‘effort’ spent. The parameters of the linear combination have to be chosen in some way that ‘normalises’ the disparate ‘times’. One way of doing this that is under investigation is to try to assess, subjectively, the different values (e.g. in monetary terms) of each different time unit.

These difficulties are compounded, of course, when we take account of aspects of effort other than time. For example, some attackers reported having used external resources (e.g. documentation, friends, ...) off-line in order to learn about possible ways to break into the system, but it is not presently clear how to combine these with time-based effort. However, in many attacks the software used by the attackers was publicly available (e.g. *Crack*, *Xkey*), and could thus be regarded as having zero cost. In that case the times the attackers spent retrieving, learning about and setting up such software and monitoring any results could be sufficient to characterise the effort involved, as described above.

An alternative way out of the difficulty of combination would be to generalise the underlying theoretical model so that effort can be represented as a vector. However, this has the disadvantage of losing the conceptual simplicity of the original approach, where different types of effort could be imagined to be equivalent via some 'common currency' such as cost. Our present view is that a single measure, whilst somewhat idealistic, is the best way forward.

In most cases carrying out a single attack involved the attacker in a combination of different kinds of activity, e.g. some learning and planning, followed by writing some software, followed by executing the software, etc. The reporting in the second experiment was sufficiently complete that it is possible to associate effort measures (i.e. times) to the different activities that constitute an attack. This introduces another type of variation between attackers - how they *apportion* their effort - that seems worthy of further attention.

7.2 Breach Events and Rewards

From the accounts of their attacking activity, and from the reports of the breaches they claimed to have made, it was possible to learn something about the nature of the attackers' reward processes. Firstly, it was clear that each attacker generally associated very different rewards with the achievement of different successes. Not surprisingly, attackers associated a very high reward with getting root on the server ('the ultimate goal' for many of them) whilst, for example, associating a lesser reward with getting root on a local machine. Again, fairly obviously, in cases where one (*partial*) breach was used to achieve some further goal - for example local root to achieve root on the server, or discovery of some other vulnerability in the system which later led to getting root on the server - the attackers attached higher reward to the ultimate breach than to the prior partial breach. The main other type of success with which the attackers clearly associated substantial rewards was that of obtaining passwords. Other events with which the attacker associated rewards were generally to do with learning about security issues in the system. In some cases the attackers seemed to attach reward to having proved that a particular attack method would work even though no higher objective was achieved (e.g., proving that their implementation of a Trojan horse could catch passwords, but not subsequently actually getting any passwords from this method).

In the pilot it was sometimes possible to rank different successes in order of the amount of subjective reward they represented to the attacker, but this was generally not the case, particularly when successes were of different types. For example, for ordinary user accounts where an attacker is indifferent between these users, whilst most attackers would rate getting many passwords for different accounts more highly than getting just one, it was not clear how they would value getting many passwords of ordinary users compared with getting the system owner's password. Ranking of even more disparate kinds of successes, e.g. local root versus some users' passwords, seems even more of a problem.

Both the pilot and the second experiment confirmed our conjecture that different attackers *did* seem to have different rewards associated with the occurrence of the same successes (in other words different attackers had different motivation driving their attacking behaviour). For example, it was apparent that some attackers seemed to get more personal satisfaction from being in a position to do something malicious than others, some found learning particularly important and attached more reward to this than other attackers, others seemed to get a lot of intellectual satisfaction from having done something clever. These results confirm the necessity to adopt a subjective approach to the probability modelling.

It was noticeable that the attackers found it much harder to assess continuous accrual of reward even than they did the reward associated with single breach events. Thus it seems particularly difficult to assess the reward associated with gradual learning about a system.

8. Conclusions

These experiments have demonstrated the feasibility *in principle* of measuring effort expended in attacking a system, and thus eventually obtaining a quantitative, probabilistic theory of security akin to that which has been available for many years for reliability. The pilot experiment was invaluable in pointing to difficulties such as incomplete reporting, so that these were overcome in the second experiment. The result was that we obtained accurate measures of the different times associated with attacks, and with different activities of attackers that constituted the attacks. Although even the second experiment was quite small, it succeeded in obtaining a considerable amount of data of this kind.

Further work is now needed on ways to combine the different components of ‘effort’, such as these different times, into single measures of effort as required by the probabilistic theory of security assessment. This work is currently underway.

Measures of reward present even harder problems than effort. Our approach to this was to be as comprehensive as possible in our data collection, so that there exists a data-base containing details of the nature of all reported breaches as well as the constituents of effort discussed earlier. This data-base structure will itself be a valuable resource in future experiments. In the meantime, the results of the second experiment are being analysed using the subjective judgements of the experimenter as the measures of reward.

In conclusion, we recognise that there remain formidable difficulties in this kind of quantitative assessment of security. On the other hand, we have shown that it is possible to come close to defining the required effort-based measures of security, even in the small confines of the experiments that we have conducted. We believe that it is worthwhile to continue with the work in the expectation that the remaining problems can be overcome after further investigation.

9. References

[Attanasio et al 1976] C. R. Attanasio. P. Markstein and R. J. Phillips, “Penetrating an Operating System: A Study of VM/370 Integrity”. IBM Systems J., 15 (1), pp. l02-16, 1976.

[Bishop 1989] R. Bishop, “Computer Security - A Common Sense Model”, Computer (5 October), pp.42-3, 1989.

[Brocklehurst and Littlewood 1992] S. Brocklehurst and B. Littlewood, “New ways to Get Accurate Reliability Measures”, IEEE Software, vol. 9, No. 4, pp. 34-42, 1992.

[Brocklehurst et al 1994] S. Brocklehurst, B. Littlewood, T. Olovsson and E. Jonsson, “On Measurement of Operational Security”, in COMPASS 94 (9th Annual IEEE Conference on Computer Assurance), (Gaithersburg), pp.257-66, IEEE Computer Society, 1994.

[Denning 1987] D. E. Denning, “An Intrusion-Detection model”, IEEE Trans. Software Engineering, 12 (2), pp.222-32, 1987.

[Herschberg 1988] I. S. Herschberg, “Make the Tigers Hunt for You”, Computers and Security, 7, pp. 197-203, 1988.

[ITSEC 1991] Information Technology Security Evaluation Criteria (ITSEC): *Provisional Harmonized Criterivp2a, December 1993*. ISBN 92-826-7024-4.

[Jonsson and Olovsson 1992] E. Jonsson, T. Olovsson, “On the Integration of Security and Dependability in Computer Systems”, IASTED International Conference on Reliability, Quality Control and Risk Assessment, Washington, Nov. 4-6, 1992. ISBN 0-88986-171-4, pp. 93-97.

[Jonsson and Olovsson 1994] E. Jonsson, T. Olovsson, “Security in a Dependability Perspective”, Nordic Seminar on Dependable Computing Systems 1994 (NSDCS’94), Lyngby, Aug. 24-26, 1994. pp. 175-186.

[Landwehr et al 1994] C. E. Landwehr, A. R. Bull, J. P. McDermott and W. S. Choi, “A Taxonomy of Computer Program Security Flaws”, ACM Computing Surveys, 26 (3) 1994.

[Lee 1989] T. M. P. Lee, “Statistical Models of Trust: TCBs versus People”, in IEEE Symposium on Security and Privacy, (Oakland), pp. 10-19, IEEE Computer Society Press, 1989.

[Littlewood 1989] B. Littlewood, “Predicting Software Reliability”, Phil Trans Royal Soc London, A 327, pp. 513-49, 1989.

[Littlewood 1991] B. Littlewood, “Limits to evaluation of Software Dependability”, in Software Reliability and Metrics (Proc. of 7th Annual CSR Conference, Garmisch-Partenkirchen, London. Elsevier, pp. 81-110, 1991.

[Littlewood et al 1994] B. Littlewood, S. Brocklehurst, N.E. Fenton, P. Mellor, S. Page, D. Wright, J.E. Dobson, J.A. McDermid and D. Gollmann, “Towards operational measures of computer security", Journal of Computer Security, vol. 2, no. 3.

[NSCS 1985] NCSC, Department of Defense Trusted Computer System Evaluation, National Computer Security Center, Department of Defense, No DOD 5200.28.STD, 1985.

[Olovsson et al. 1993] T. Olovsson, E. Jonsson, S. Brocklehurst, B. Littlewood, “Data Collection for Security Fault Forecasting: Pilot Experiment”, Technical Report No

167, Department of Computer Engineering, Chalmers University of Technology, 1992 and ESPRIT/BRA Project No 6362 (PDCS2) First Year Report, Toulouse Sept. 1993, pp. 515-540.

[Olovsson and Jonsson 1995] T. Olovsson, E. Jonsson, “A Practical Security Intrusion Experiment - Compilation of Recorded Data”, Technical Report No 246, Department of Computer Engineering, Chalmers University of Technology, 1995.

APPENDIX A: Cert Advisory CA-94:06

```
=============================================================================
CA-94:06                        CERT Advisory
                               March 21, 1994
                       Writable /etc/utmp Vulnerability

=============================================================================

The CERT Coordination Center has received information concerning a
vulnerability that exists on systems where the file /etc/utmp is writable
by any user on the system.

This vulnerability is being actively exploited; please review CERT Advisory
CA-94:01 "Ongoing Network Monitoring Attacks."

The problem is known to affect Sun Microsystems, Inc. SunOS 4.1.X and
Solaris 1.1.1 operating systems. Solbourne Computer, Inc. and other Sparc
products using SunOS 4.1.X or Solaris 1.1.1 are also affected. Solaris 2.x
is not affected by this problem.

Patches can be obtained from Sun Answer Centers worldwide.  They are also
available via anonymous FTP from ftp.uu.net in the /systems/sun/sun-dist
directory, and in Europe from ftp.eu.net in the /sun/fixes directory.

CERT queried several vendors in addition to Sun.  The following vendors
reported that their operating systems, as distributed by the vendor, are
not affected by this problem:

Convex Computer Corporation           Digital Equipment Corporation
Data General Corporation              Hewlett-Packard Company
IBM                                   Intergraph
Motorola, Inc.                        NeXT, Inc.
Pyramid Technology Corporation        Sequent Computer Systems
Sony Corporation

Currently, we are not aware of /etc/utmp being writable on other systems.
If your operating system is not explicitly mentioned above, and if you
determine that /etc/utmp is writable by someone other than root, we
encourage you to contact your vendor.

If /etc/utmp on your system is writable only by the root account, you need
not be concerned about the vulnerability.

CERT recommends that sites check their /etc/utmp file to be sure it is not
writable by users other than root.  If it is generally writable, you should
obtain patches from the system vendor or protect /etc/utmp as described below.

-----------------------------------------------------------------------------

I.   Description

     If the file /etc/utmp is writable by users other than root,
     programs that trust the information stored in that file can
     be subverted.

II.  Impact

     This vulnerability allows anyone with access to a user account
     to gain root access.

III. Solution
```

The solutions to this vulnerability are to either (a) protect the file, or (b) patch all the programs that trust it.

A. To protect the file, make /etc/utmp writable only by root:

```
# chown root /etc/utmp
# chmod 644 /etc/utmp
```

B. Patches from Sun Microsystems

Program	Patch ID	Patch File Name
in.comsat	100272-07	100272-07.tar.Z
dump	100593-03	100593-03.tar.Z
syslogd	100909-02	100909-02.tar.Z
in.talkd	101480-01	101480-01.tar.Z
shutdown	101481-01	101481-01.tar.Z
write	101482-01	101482-01.tar.Z

Program	BSD Checksum	SVR4 Checksum	MD5 Digital Signature
in.comsat	26553 39	64651 78	912ff4a0cc8d16a10eecbd7be102d45c
dump	52095 242	41650 484	cdba530226e8735fae2bd9bcbfa47dd0
syslogd	61539 108	38239 216	b5f70772384a3e58678c9c1f52d81190
in.talkd	47917 44	32598 88	5c3dfd6f90f739100cfa4aa4c97f01df
shutdown	46562 80	56079 159	bfc257ec795d05646ffa733d1c03855b
write	61148 41	48636 81	f93276529aa9fc25b35679ebf00b2d6f

If you believe that your system has been compromised, contact the CERT Coordination Center or your representative in Forum of Incident Response and Security Teams (FIRST).

Internet E-mail: cert@cert.org
Telephone: 412-268-7090 (24-hour hotline)
CERT personnel answer 8:30 a.m.-5:00 p.m. EST(GMT-5)/EDT(GMT-4), and are on call for emergencies during other hours.

CERT Coordination Center
Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213-3890

Past advisories, information about FIRST representatives, and other information related to computer security are available for anonymous FTP from info.cert.org.