Experimenting with Quantitative Evaluation Tools for Monitoring Operational Security

Rodolphe Ortalo, Yves Deswarte, *Affiliate*, *IEEE Computer Society*, and
Mohamed Kaâniche, *Affiliate*, *IEEE Computer Society*

Abstract—This paper presents the results of an experiment in security evaluation. The system is modeled as a privilege graph that exhibits its security vulnerabilities. Quantitative measures that estimate the effort an attacker might expend to exploit these vulnerabilities to defeat the system security objectives are proposed. A set of tools has been developed to compute such measures and has been used in an experiment to monitor a large real system for nearly two years. The experimental results are presented and the validity of the measures is discussed. Finally, the practical usefulness of such tools for operational security monitoring is shown and a comparison with other existing approaches is given.

Index Terms—Security assessment, operational vulnerabilities, privilege graph, quantitative evaluation.

1 Introduction

Security is an increasing worry for most computing system administrators. Computing systems are more and more vital for most companies and organizations, while these systems are made more and more vulnerable by new user requirements and new services (groupware and other information sharing facilities, interconnection to insecure networks, and powerful applications whose complexity may hide serious security flaws; etc.). On the other hand, for most users of current computing systems, security is not a main concern and they are not prepared, for the sake of security, to waive their system ease of use or to give up information sharing facilities. In such conditions, it is difficult to reach an acceptable degree of security since users play an important role in the computing system security: Even the best system, designed for the highest security, would be insecure if badly operated by casual users and a lax use of the most efficient protection mechanisms would introduce flaws that could be exploited by possible attackers.

Thus, one of the main tasks of most computing system administrators is to negotiate with users to make them change their careless behavior and improve the system security. And this is not an easy job: Why would a user renounce his bad habits if he considers that he does not own sensitive data or applications? It may be difficult for him to understand that the flaws he introduces in the system are endangering other user accounts, possibly with more sensitive information. The set of tools presented here aims at facilitating this administrator's task: By providing a quantitative assessment of the current system security level, these tools can help the administrator to identify those security flaws which can be eliminated for the best security improvement with the least effect on users. Such quantitative evaluation tools should also enable him to monitor the evolution of global system security with respect to modifications of the environment, of the configuration, of applications, or of user behavior.

The measurements delivered by the evaluation tools should represent as accurately as possible the security of the system in operation, i.e., its ability to resist possible attacks or, equivalently, the difficulty for an attacker to exploit the vulnerabilities present in the system and defeat the security objectives. Several characteristics can be deduced from these definitions:

- The security measure characterizes the security of the system itself, independently of the threats it has to face: The system is the same (and its security measure should be the same) whether there are many or few potential attackers with high or low competence and tenacity. But, of course, a given system (with a given security measure) will be more probably defeated by many competent, tenacious attackers than by few lazy ones!
- The security measure is directly related to security objectives: A system is secure as long as its main security objectives are respected, even if it is easy to perform illegitimate actions which do not defeat the objectives. For instance, a system can be secure even if it is easy for an intruder to read some public information.
- The main use of security measures is to monitor the security evolution of a given system rather than rate absolutely the security of different systems: It is more important to know if the security of a given system is improving or decaying than to compare the security of independent systems with different objectives, applications, users, environments, etc. Moreover, the security measures should be sensitive to system modifications that bring new

• *R. Ortalo, Y. Deswarte, and M. Kaâniche are with LAAS-CNRS 7, avenue du Colonel Roche, 31077 Toulouse cedex 4, France.*
E-mail: {ortalo, deswarte, kaaniche}@laas.fr.

Manuscript received 15 Aug. 1998; revised 10 Feb. 1999.
Recommended for acceptance by W.H. Sanders.
For information on obtaining reprints of this article, please send e-mail to: tse@computer.org, and reference IEEECS Log Number 109548.

vulnerabilities or correct previous ones that influence security significantly, and these measures should evolve accordingly.

A theoretical framework has been developed at LAAS to identify and compute such measures [1], [2], [3], [4]. This framework is based on a: 1) theoretical model, the privilege graph, exhibiting system vulnerabilities, 2) definition of security objectives, and 3) mathematical model based on Markov chains to compute the security measures. To demonstrate the feasibility of the approach, this theoretical framework has been implemented by a set of software tools which can compute the security measures of large Unix systems.

But, a major question is raised by such an approach: Do the obtained measures accurately represent system security? In this domain, no direct validation is expected: Real attacks on real systems are too rare for a precise correlation to be obtained between the computed measures and a success rate of attacks. Even a tiger team approach would probably be ineffective since such attacks are not necessarily representative of real attacks and, to obtain accurate results, tiger team attacks must be numerous and applied to a stable system [5], while our measures are intended to rate the dynamic evolution of system security. So, the only practical validation is experimental: We have chosen to observe the security measures computed on a real full-scale system during a long period and to analyze each significant measure change with respect to the events triggering these changes.

This paper presents this experiment. Section 2 presents a description of the theoretical framework. Section 3 presents the experiment itself and discusses the results. Finally, Section 4 draws a conclusion.

2 Presentation of the Approach

2.1 Formal Description of Operational System Vulnerabilities

It has been shown in [6] that the vulnerabilities exhibited by an operational computing system can be represented in a *privilege graph*. In such a graph, a node X represents a set of privileges owned by a user or a set of users (e.g., a Unix group). An arc represents a vulnerability. An arc exists from node X to node Y if there is a method allowing a user owning X privileges to obtain those of node Y. Three classes of vulnerabilities may be identified:

- A first class of arcs corresponds to privilege subsets directly issued from the protection scheme. For instance, in Unix systems, there is an arc from each node representing the privilege set of a group member to the node representing the privilege set of the group.
- A second class corresponds to direct security flaws, such as an easily guessed password or bad directory and file protections enabling the implantation of a Trojan horse.
- A third class of arcs contains the vulnerabilities which do not correspond to security flaws. Instead, they can result from the use of a feature designed, for instance, to improve security.

To illustrate the third class, we consider three examples based on Unix:

- *The* `.rhosts` *file*. To log in a Unix system, a password is normally required. However, there is a mechanism in Unix that allows remote trusted users to access a local system without supplying a password. This functionality is based on the `.rhosts` file, which enables the standard password-based authentication to be bypassed. Therefore, a user U1 can grant most of his privileges to another user U2 without disclosing his password. This is not a security flaw if U1 trusts U2 and needs U2 to undertake some tasks for him (less secure solutions would be to give his password or reduce his protections). But, if U2 also grants similar privileges to U3 (U2 trusts U3), then by transitivity, U3 can reach U1's privileges, even if U1 does not trust U3. This corresponds to a vulnerability associated with the privilege transfer mechanism provided by Unix. It is also noteworthy that any user who can write the `.rhosts` file of U1 can get the set of privileges of U1. It is clearly a bad idea to grant such permission and it corresponds generally to a direct security flaw.
- *The* `.xinitrc` *file*. When starting, the X Window system looks for a specific file in the user's home directory, called `.xinitrc`, to run as a shell script to start client programs. Daily practice shows that novice users have trouble configuring this file correctly. If a novice user trusts another user, more expert in X Window than himself, he may prefer using the expert's configuration file. An easy solution to do so is to establish a symbolic link between his own configuration file and the expert's. Afterward, the novice user will enjoy any enhancement made by the expert. From a security point of view, this can be a good solution. If the novice chooses inappropriate options or commands, however, this file will become a trapdoor, leaving his data unprotected. Using the file of the expert, who should be aware of the vulnerabilities, the novice's data security is indeed enhanced. However, he is at the mercy of this expert who can introduce a Trojan horse in his configuration file and then acquire most of the novice's privileges.
- *The* `setuid` *files*. In Unix, every process inherits its privileges from the user for whom the process is run. However, it is possible to let a process get the privileges of the owner of the program rather than the privileges of the user initiating the process. This is particularly useful when an operation needs more privileges than those held by the user. An example of this is the program `/bin/passwd` that changes user passwords: Every user must be able to change his own password but this operation requires writing in a protected file (usually the `/etc/passwd` file) to which no user has write access except the superuser; to do so, `/bin/passwd` uses the `setuid` facility to run with superuser privileges on behalf of less privileged users. This functionality has many other applications, all of them being

examples of granting sets of privileges by the owner of the program to the user of the program. As long as these `setuid` programs are correct and no low privileged user can create or modify such programs, the security is satisfactory. Indeed, this feature strengthens security since, without this feature, users should be granted more privileges constantly. But if a `setuid` program owner trusts another user and gives him write access to his program, he is at the mercy of this user.

In the same way as in the field of penetration analysis [7], these examples show also that vulnerabilities can be identified by analyzing the system configuration and, in particular, the file system: Each specific vulnerability corresponds to some specific information in a file or a directory.

Fig. 1 gives an example of such a privilege graph, with arcs being labeled by vulnerability classes. In this figure, each node corresponds to the set of privileges of users (A, B, F) or groups of users $(X_{admin}, P1)$. The node *insider* represents the minimal privileges of any registered user (e.g., the privilege to execute login, to change his own password, etc.).

Some sets of privileges are highly sensitive (e.g., the superuser privileges). These nodes are called "target" nodes since they are the most likely targets of attacks. On the other hand, it is possible to identify some nodes as the privileges of possible attackers; these nodes will be called "attacker" nodes. If a path exists between an attacker node (e.g., *insider*) and a target node (e.g., A), then a security breach can potentially occur since an attacker can exploit system vulnerabilities to obtain the target privileges.

In most real systems, such paths exist because a lot of possible vulnerabilities exist, even if an attacker cannot exploit most of them. For instance, all passwords can be guessed with some luck, but some passwords can be easily obtained by all "insiders" because they are in a dictionary and automatic tools such as `crack` [8] can identify them in a short time, while other passwords are much more complex and the only practical means to get them is by exhaustive search. This is true for each class of arcs. Some vulnerabilities are easily exploitable by an attacker (e.g., the arc corresponding to a group membership), while others may require knowledge, competence, tenacity, or luck. This means that, even if a path exists between an attacker node and a target node, the system security has a low probability of being defeated if an attacker needs a lot of cleverness, competence, or time to run through all the arcs composing the path. With the definition given in Section 1, a measure of the difficulty for the attackers to reach the targets would be a good measure of the security of the system.

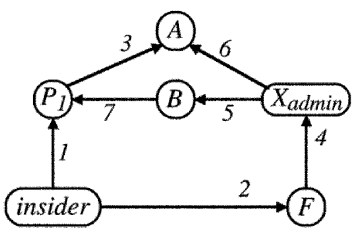


Fig. 1. Example of a privilege graph. Vulnerability Classes: 1) Y's `.rhosts` is writable by X; 2) X can guess Y's password; 3) X can modify Y's `.tcshrc`; 4) X is a member of Y; 5) Y uses a program managed by X; 6) X can modify a setuid program owned by Y; 7) X is in Y's `.rhosts`.

2.2 Single Vulnerability Rating

To assess this measure, each arc in the privilege graph is assigned a weight corresponding to the "effort" needed for a potential attacker to perform the privilege transfer corresponding to this arc. This notion of effort encompasses several characteristics of the attack process, such as the availability of attack tools, time needed to perform the attack, computing power available for the attacker, etc. [9]. For example, the effort needed to obtain a password can be assessed by estimating the computing power and the time needed by `crack` to identify the password. For a Trojan horse attack, the effort depends on the competence needed to design the Trojan horse, the time needed to implant it in a program that can be executed by the target user, and the time needed for the target user to activate it (the latter does not depend on the attack process, but only on the user behavior).

The effort weight assigned to an arc is thus a compound parameter which can be represented as a rate of success for the corresponding elementary attack. Ideally, the choice of the set of significant variables should be based on the analysis of all known intrusion scenarios and their associated relevant parameters. Unfortunately, such detailed intrusion data is not available. However, it is possible to rely on information provided by security experts and use general parameters to obtain a sensible and meaningful rating of each vulnerability considered. These data could further be complemented by additional information coming from user profiles built by intrusion detection tools [10].

The ITSEM [11] evaluation manual also provides a similar framework for such a quantification. In the ITSEC [12], three levels are defined to rate the strength of security mechanisms: *basic*, *medium*, and *high* (ITSEC, 3.6-3.8). The ITSEM provides guidance on more objective means for rating these strengths. Different factors are identified: *expertise*, *time*, *equipment*, and *collusion*[1] (ITSEM, 6.C.30). Some rules are further provided to obtain the strength of a particular mechanism once these factors have been evaluated (ITSEM, 6.C.31-6.C.33).

Clearly, several parameters should be considered to evaluate the weight assigned to each class of vulnerabilities. To be able to rank these classes according to the difficulty for an attacker to exploit them, a single value, the effort, incorporating the contribution of each significant parameter for the class considered, should be defined (e.g., using a method similar to the one proposed in the ITSEM). Precise estimation of the various parameters involved in the computation of this effort, using system analysis tools, would allow improving further the rating of each vulnerability.

The following section presents a model to compute the global privilege graph security measure from the elementary arc weights.

1. In the ITSEM, time and equipment correspond to two specific types of *resources* and collusion is a specific form of *opportunity* (other aspects of opportunity include *chance* and *detection*).

2.3 Attacker Behavior

In order to evaluate quantitative measures characterizing the operational security based on the privilege graph, it is necessary to identify the scenarios of attacks that may be attempted by a potential attacker to reach the target. Minimal assumptions describing the attacker behavior are necessary to build all these scenarios. Here, first, we assume that the attacker is sensible and he will not attempt an elementary attack that would give him privileges that he already possesses. Additional assumptions are required to characterize the progress of the attacker toward the target. Different models can be defined depending on the assumptions considered about the behavior of the attacker. The first model that can be considered is to assume that the attacker chooses the shortest path leading to the target (denoted as SP in the following), i.e., the one that has the lowest mean value of cumulative effort. The shortest path can be evaluated directly from the privilege graph, taking into account the rates assigned to the arcs.

We assume that, initially, the attacker has no information about vulnerabilities (except from the starting point in the graph) and only learns about vulnerabilities through successful attacks that give him more privileges. Indeed, the attacker would need a global view of the system vulnerabilities to be able to deliberately exploit the shortest path. Yet, to build the whole privilege graph, the attacker needs all the sets of privileges described in the graph (e.g., the `root` privileges). If the attacker already has these privileges, he does not need to build the graph! In fact, without such a global view of the system, the shortest path is simply not available to an attacker. Clearly, the shortest path assumption is not satisfactory. In the following, we will introduce two alternative assumptions and show that the corresponding security measures are more instructive for the security administrators than measure SP.

The attacker's privilege increases as a result of his progress toward the target can be characterized by a state-transition graph where each state identifies the set of privileges that he has already gained and transitions between states occur when the attacker succeeds in an elementary attack, allowing him to acquire new privileges. In order to fully characterize the *attack state graph*, we need to specify an additional assumption defining which steps will be attempted by the attacker at each stage of the attack. Two different assumptions are discussed hereafter, each of them corresponding to a specific attack model (i.e., attacker profile):

Total memory (TM) assumption. At each stage of the attack, all the possibilities of attacks are considered (i.e., those from the newly visited node of the privilege graph and those from the already visited nodes that he did not apply previously). At each step, the attacker may choose one elementary attack among the set of possible ones.

Memoryless (ML) assumption. At each newly visited node of the privilege graph, the attacker chooses one of the elementary attacks that can be issued from that node only (without considering the other attacks from the already visited nodes that he did not apply previously).

For both assumptions, it is assumed that the attack stops when the target is reached. We do not consider situations where attackers may give up or interrupt their process.

Fig. 2 plots the attack state graph corresponding to the example given in Fig. 1 when assumptions TM and ML are considered. It is assumed that *insider* is the attacker node and A is the target node. To improve the clarity of the figure, X_{admin} and *insider* are respectively referred to as X and I. It can be seen that the scenarios of attacks represented in Fig. 2b correspond to a subset of those identified in Fig. 2a.

To illustrate the main difference between assumptions TM and ML, let us assume that a new vulnerability is added to the privilege graph, such as an additional path is offered to the attacker to reach the target. For an attacker behaving according to TM, this new vulnerability will ease his attack. When he reaches the node corresponding to the new vulnerability, he has the possibility of choosing at each further stage of the attack, the easiest vulnerability that will lead him to the target (among the set composed of the new vulnerability and those identified during previous steps), whatever the difficulty of the new vulnerability. However, for an attacker behaving according to ML, this new vulnerability may have a positive or a negative impact on his attack. Indeed, if this vulnerability is easy, but followed by very difficult ones in the path leading to the target, the attacker does not have the possibility of backtracking and try other paths that probably will lead him to reach the target with less effort.

As an example, let us consider the privilege graph presented in Fig. 1 and the corresponding attack process graphs presented in Fig. 2. From the description of the attack methods mentioned in Fig. 1, we can see that vulnerability 1 is easier to exploit than vulnerability 2 (we assume it is easier to configure and activate a common privilege transfer mechanism of Unix than to perform a successful dictionary attack on a specific encrypted password). However, afterward, exploiting vulnerability 3 may be more difficult than exploiting successively vulnerabilities 2, 4, and 6 if A does not use the `tcsh` shell program in his default configuration.[2] Under assumption ML, we see, in Fig. 2, that, once the attacker has chosen to exploit vulnerability 1, he will not backtrack and try vulnerability 2. Therefore, in this case, the existence of vulnerability 1 may slow him, even though it is easier than vulnerability 2. However, with assumption TM, the attacker, even if he has chosen to exploit vulnerability 1 first, can decide later to exploit the path 2, 4, 6.

2.4 Mathematical Model

In order to be able to compare the evolution of the security measures corresponding to assumptions TM, ML, and SP, we need to specify the mathematical model that is used to evaluate the mean effort for an attacker to reach the target. Our investigations led us to choose a Markovian model that satisfies some intuitive properties regarding security evolution (see [2], [3] for further details). The Markov model is based on the assumption that the probability of success in a

2. Other common Unix shells (e.g., `bash`, `sh`, `csh`) have different initialization files (e.g., `.bash_profile`, `.profile`, `.cshrc`).

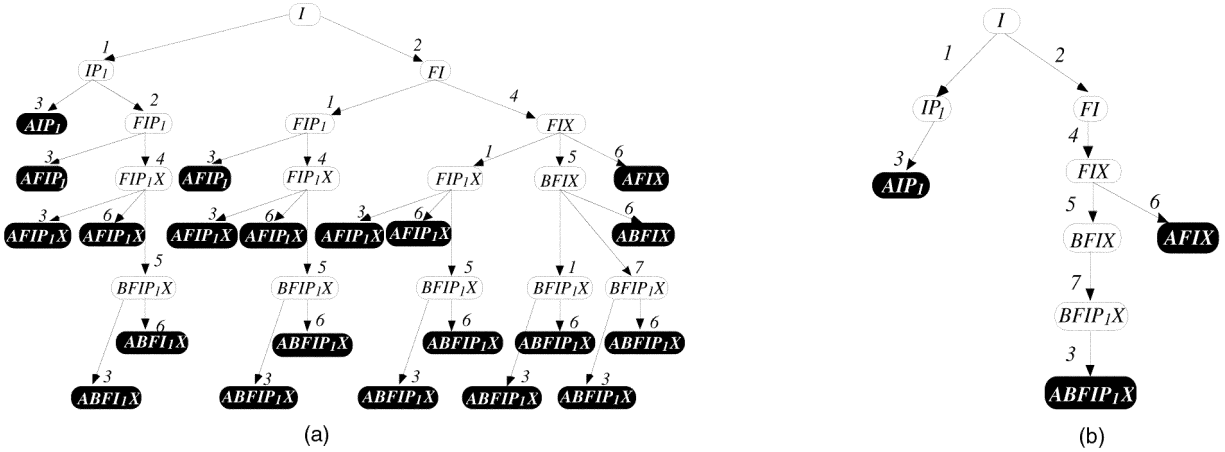


Fig. 2. Attack state graphs corresponding to the example of Fig. 1. (a) TM assumption; (b) ML assumption.

given elementary attack before an amount of effort "e" is spent is described by an exponential distribution given by:

$$P(e) = 1 - exp(-\lambda e),$$

where λ is the rate assigned to the attack. Practical considerations derived from the use of this distribution are the following:

- the potential attacker will eventually succeed in reaching the target, if a path leading to the target exists, provided that he spends enough effort;
- the mean effort to succeed in a given elementary attack is given by $1/\lambda$.

The last point is particularly valuable since the knowledge of the attack rates is sufficient to characterize the whole distribution. The first point deserves some clarification. In fact, as our aim is to evaluate system resiliency to successful attacks with respect to a specified target, we only consider scenarios of attacks that eventually lead to the target and not the scenarios which may be aborted during the attack process.

Based on the Markovian assumption, each transition in the attack state graph is rated with the success rate of the corresponding vulnerability. Various probabilistic measures can be derived from the model, among these, the mean effort for a potential attacker to reach the specified target, denoted as mean effort to security failure (METF), by analogy with mean time to failure (MTTF). This metric allows easy physical interpretation of the results: the higher the METF the better the security. Moreover, simple analytical expressions can be easily obtained and analyzed in order to check the plausibility of model results.

The METF is given by the sum of the mean efforts spent in the states leading to the target, which are weighted by the probability of visiting these states. The mean effort spent in state j, denoted as E_j, is given by the inverse of the sum of state j output transition rates:

$$E_j = 1 \Big/ \sum_{i \in out(j)} \lambda_{ji}\,. \tag{1}$$

$out(j)$ is the set of states reachable in a single transition from state j and λ_{ji} is the transition rate from state j to state i.

Let us denote by METF_k the mean effort when state k is the initial state and P_{ki} the conditional probability transition from state k to state i, then:

$$\mathrm{METF}_k = \mathrm{E}_k + \sum_{i \in out(k)} \mathrm{P}_{ki} \times \mathrm{METF}_i \;;\; \mathrm{P}_{ki} = \lambda_{ki} \times \mathrm{E}_k \tag{2}$$

According to this model, the highest output conditional probabilities values correspond to the transitions with the highest success rates.

Clearly, the complexity of the METF computation algorithm is related to the size of the attack state graph and, thus, to the assumption adopted. For assumption TM, the number of paths leading to the target to be considered is higher than the number of paths corresponding to assumption ML.

2.5 Assumptions TM, ML, and SP—Expected Behaviors

In the following, we analyze the expected behaviors of the METF when assumptions TM, ML, and SP are considered.

2.5.1 Single Path

Let us consider first the example of a privilege graph containing a single path between the attacker node and the target node (see Fig. 3). In this case, the METF is given by

$$\sum_{j=1..k} 1/\lambda_j\,,$$

where k is the number of arcs in the path and λ_j is the success rate associated to the elementary attack j. The same value of the METF is obtained when either assumption TM, ML, or SP is considered. Clearly, as the number of arcs increases, the METF increases and the security improves. Also, when the values of λ_j increase, the METF decreases and the security degrades.

2.5.2 Multiple Paths

As regards the SP assumption, the shortest path is obtained by identifying, in the privilege graph, all the direct paths

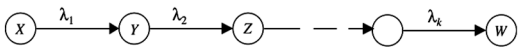


Fig. 3. Markov model corresponding to a single path.

from the attacker node to the target node and evaluating the minimum value of the METF among the values computed for each direct path. A direct path from the attacker to the target is such that each node that belongs to this path is visited only once. The expression of $\mathrm{METF_{SP}}$ is:

$$\mathrm{METF_{SP}} = \min\{\mathrm{U}_1, \ldots, \mathrm{U}_n\},$$

where $\mathrm{U}_l = \sum 1/\lambda_i$, λ_i is the rate assigned to the arc i that belongs to direct path l, n is the number of direct paths.

The METF values corresponding to assumptions TM or ML can be obtained by processing the corresponding attack state graph. Let us consider the example of Fig. 4, where A is the attacker and D is the target. The privilege graph (Fig. 4a) indicates the presence of two paths leading to the target. The Markov models corresponding to assumptions ML and TM are given in Fig. 4b and Fig. 4c, respectively. Application of (1) and (2) leads to the following expressions:

$$\mathrm{METF_{ML}} = \frac{1}{\lambda_1+\lambda_3} + \frac{\lambda_1}{\lambda_1+\lambda_3}\times\frac{1}{\lambda_2} + \frac{\lambda_3}{\lambda_1+\lambda_3}\times\frac{1}{\lambda_4}$$

$$\mathrm{METF_{TM}} = \frac{1}{\lambda_1+\lambda_3} + \frac{\lambda_1}{\lambda_1+\lambda_3} \times \left(\frac{1}{\lambda_2+\lambda_3} + \frac{\lambda_3}{\lambda_2+\lambda_3}\times\frac{1}{\lambda_2+\lambda_4}\right) + \frac{\lambda_3}{\lambda_1+\lambda_3}\times\left(\frac{1}{\lambda_1+\lambda_4} + \frac{\lambda_1}{\lambda_1+\lambda_4}\times\frac{1}{\lambda_2+\lambda_4}\right).$$

It could be seen that, for any value of λ_1, λ_2, and λ_3, the expression of $\mathrm{METF_{TM}}$ is always lower than $1/\lambda_1 + 1/\lambda_2$ (which corresponds to the case where only the first path exists), and to $1/\lambda_3 + 1/\lambda_4$ (which corresponds to the case where only the second path exists). This result illustrates the fact that the addition of new paths leading to the target in the privilege graph surely leads to a decrease of $\mathrm{METF_{TM}}$ that indicates security degradation. This result can be generalized; further details can be found in [2].

However, assumption ML leads to a different behavior since $\mathrm{METF_{ML}}$ may increase or decrease depending on the values of the parameters. For instance, $\mathrm{METF_{ML}}$ is lower than $1/\lambda_1 + 1/\lambda_2$ only if $1/\lambda_4 < (1/\lambda_1 + 1/\lambda_2)$, i.e., when the mean effort spent in obtaining the privileges of node D from node C is lower than the mean effort corresponding to the initial path. This is due to the fact that, with assumption ML and contrary to assumption TM, when the attacker chooses a given path, he never backtracks until he reaches the target. If the modifications introduced in the privilege graph lead to some additional paths which are shorter than those derived from the initial privilege graph, then $\mathrm{METF_{ML}}$ decreases; otherwise the $\mathrm{METF_{ML}}$ may increase.

From the previous discussion, it can be concluded that $\mathrm{METF_{TM}}$ is always lower than the mean effort calculated based on the shortest path only ($\mathrm{METF_{TM}} \leq \mathrm{METF_{SP}}$). For assumption ML, $\mathrm{METF_{ML}}$ may be lower or higher than $\mathrm{METF_{SP}}$ depending on the values of the parameters and the structure of the privilege graph.

The last property that is worth mentioning concerns the comparison of $\mathrm{METF_{ML}}$ with $\mathrm{METF_{TM}}$. Since the attack scenarios corresponding to assumption ML are a subset of those obtained with assumption TM, it can be proven that, for the same privilege graph, assumption ML leads to higher METF values than assumption TM: $\mathrm{METF_{ML}} \geq \mathrm{METF_{TM}}$.

2.6 Discussion

Based on the results of the previous section, Table 1 summarizes the expected behavior for measures $\mathrm{METF_{ML}}$ and $\mathrm{METF_{TM}}$ when an event happens such that the number of paths between the attacker and the target set of privileges increases, due to one new vulnerability appearing in the system.

When only one modification of the privilege graph occurs, we should expect that:

- if the number of paths increases because of the addition of a new vulnerability, $\mathrm{METF_{TM}}$ decreases since this new path weakens the security of the target;
- when the shortest path between the attacker and the target decreases, $\mathrm{METF_{TM}}$ decreases and shows a degradation of security;
- as discussed in the previous section, two kinds of behavior may be observed for $\mathrm{METF_{ML}}$:
  - if the new path decreases the probability of taking another relatively easy path to the benefit of a longer new one, $\mathrm{METF_{ML}}$ may increase (indicated as behavior 2 in Table 1);
  - otherwise, $\mathrm{METF_{ML}}$ should have the same evolution as $\mathrm{METF_{TM}}$. It should decrease as the number of paths increases and reveal a degradation of security (behavior 1).

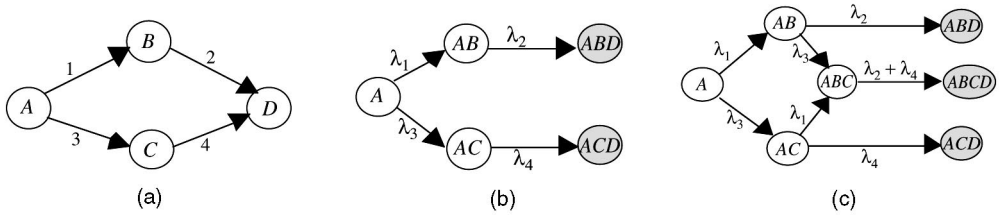


Fig. 4. Multiple paths—example. (a) privilege graphs; (b) assumption ML; (c) assumption TM.

TABLE 1
Typical Behaviors

	$METF_{TM}$	$METF_{ML}$	
Number of Paths ↑	↓	↓	Behavior 1
Number of Paths ↑	↓	↑	Behavior 2

By analyzing the variation of the measures together with the modifications introduced in the privilege graph, the security administrators can assess whether these modifications have a significant impact on security. Based on these results, they can identify the most critical paths in the privilege graph and take appropriate decisions: either correct some system weaknesses (when security decreases) or keep the system configuration unchanged if the risks induced by the modifications are not significant (either security increases or only a small decrease in the METF is observed).

With the Markov model, both measures $METF_{TM}$ and $METF_{ML}$ exhibit the expected behavior (see Section 2.3). With respect to assumptions TM and ML a potential question is which type of attacker profile is more realistic. It is difficult to answer this question because of lack of real data. In fact, both behaviors correspond to different attacker profiles and are plausible. In the experiment presented in the following section, we will see that both models provide to security administrators valuable information about security evolution.

Concerning the shortest path, it is clear that the information provided by this measure is incomplete, as only one path in the privilege graph is taken into account. Therefore, the security variation due to the presence of the other paths will not be identified if only the shortest path is computed to monitor the operational security.

3 Experiment

In order to validate the measures presented previously, we developed several tools integrated into a single prototype called ESOPE. Using ESOPE, an experiment has been carried out on a real computer system.

3.1 Tools Description

The set of tools supporting the experiment presented in this section corresponds to the main steps of the evaluation process:

1. **Definition of the security policy.** For each security objective chosen for the system, the relevant security targets (sets of privileges that must be protected) and the potential attackers (sets of privileges against which targets should be protected) are identified. Each attacker-target pair corresponds to two sets of nodes in the privilege graph for which one quantitative evaluation is needed. A tool has been developed to describe formally the security objectives from which all pairs are identified and gathered into a file.
2. **Probing the system and building the privilege graph.** We have developed a tool named ASA, for Automatic Security Advisor, which looks for known vulnerabilities in the Unix system under study and builds the related privilege graph. The tool runs with extended privileges in order to be able to analyze all parts of the system. So far, ASA uses many procedures included in the COPS package [13]. More precisely, like COPS, some Unix scripts scan the Unix file system, gathering information about the access permissions of several files either for each user or for specific directories. A crack program is run to guess user passwords using a standard dictionary. Each time a vulnerability is detected in the system, an arc is added to the privilege graph under construction. The output of the ASA tool is therefore a privilege graph describing all known vulnerabilities of the target system at the time of the snapshot. After probing the system, the privilege graph built is recorded in an archive.
3. **Evaluation.** Subsequently, another tool computes the security measures presented in Section 2 for each security objective. These computations can be applied to either a single privilege graph or a whole archive.
4. **Identification of security-relevant events.** Last, to ease the analysis of the security measures computed, a tool identifies, for each significant variation of a measure, the security events that have caused it. More precisely, it looks for the arcs involved in the paths between the attacker and target nodes that changed between two consecutive privilege graphs. This helps to identify the event(s) that caused this measure evolution. An example of the output of this tool is given in Appendix A.

3.2 Target System Description

The system under observation in this experiment is a large distributed computer system composed of several hundred different workstations connected to a local area network. There are about 700 users sharing one global file system. During the experiment, the total number of users changed frequently due to the arrival and departure of temporary staff (a frequent event for the target system). The probing of security vulnerabilities was made on the global file system. In this experiment, the system was observed for 21 months on a daily basis, starting in June 1995 until March 1997. The archive of privilege graphs contains 674 items (one for each day).

In the target system of this experiment, security is not a main concern of the users. Since no critical information is stored in the system, it is not necessary to enforce a strong global security policy, even if some users or the system administrators might worry about it for personal reasons or for safety reasons. This explains the significant number of known vulnerabilities that will be shown hereafter. It is noteworthy that most vulnerabilities persist and are accepted because they often provide useful and convenient functionalities.

Furthermore, our main objective being to validate the behavior of the security measures, we only observed the "natural" evolution of the system. We did not try to improve the system security by convincing the users to remove the vulnerabilities we had identified.

3.3 Experiment Settings

3.3.1 Security Objectives

Evaluating security measures requires that relevant sets of target(s) and attacker(s) be defined. These pairs are related to the security objectives that one would like the system to fulfill. For a Unix system, one important target to protect against attacks is the `root` account and, more generally, every account allowed to obtain superuser privileges. Another interesting target to study is the group of all system administrators, giving access to all the data they share. As an attacker, we choose the "insider" set of privileges defined in Section 2.1. Table 2 summarizes these case studies.

For the analysis of the second security objective, one problem appears due to the existence of superusers in Unix. A superuser is able to get all the privileges of any other user in the system. Thus, when we consider a target set of users to protect, this mechanism implicitly leads us to include the superuser in it (as this set of privileges includes any other set). If we had considered that, security objective 2 would have included objective 1 since if one sequence of privilege transfer methods enables defeating objective 1, it also defeats objective 2. In order to have completely distinct case studies, we did not consider vulnerabilities linked to superuser properties for objective 2. We then removed from the privilege graphs all the instantaneous arcs going directly from the superuser to the `admin_group` set of privileges.

TABLE 2
Security Objectives

	Attacker	Target
objective 1	`insider`	`root`
objective 2	`insider`	`admin_group`

TABLE 3
Attack Success Rate Levels

Name	Weight
`level1`	10^{-1}
`level2`	10^{-2}
`level3`	10^{-3}
`level4`	10^{-4}

3.3.2 Vulnerabilities

From all the known vulnerabilities in Unix, we monitored 13 of the most common, including: password checking (with `crack` software); user-defined privilege transfer methods (`.rhosts`); incorrect/exploitable permissions on `setuid` files, `.rhosts` files or initialization files (`.profile`, `.cshrc`, etc.); incorrect path search that allows Trojan horse attacks; etc. A more detailed review of all the classical Unix vulnerabilities can be found in [14]. In addition to this, specific arcs are defined to represent the first class of vulnerabilities identified in Section 2.1: the inclusion of privilege sets, for example, between one user node and all the nodes of the Unix groups he belongs to.

Security state modifications, or *events*, occur when vulnerabilities are either created or eliminated (arcs in the privilege graph are added or deleted) or when the value associated with one vulnerability (the weight assigned to an arc) changes. Such events occurred frequently during the experiment.

3.3.3 Quantification

For the experiment, we defined a four-level classification scale to rate the different vulnerabilities (see Table 3), where the levels differ from each other by one order of magnitude: `level1` corresponds to the easiest elementary attacks and `level4` to the most difficult ones.

The various levels assigned to each elementary attack are reasonable, taking into account the method involved, but they remain rather arbitrary. Evaluating precisely the success rate of the various attacks present in the system (see Section 2.2) would have required additional tools (such as for recording user profiles) that are not currently available in our prototype. However, this is not a serious drawback as this experiment aims primarily at validating relative changes in the security measures over time rather than rating the security of the system on an absolute scale.

3.4 Experiment Results

The results of this experiment corresponding to objectives 1 and 2 are presented in Fig. 5 and Fig. 6, respectively. The measures presented are: the number of paths found between attacker and target nodes, $\mathrm{METF}_{\mathrm{SP}}$, $\mathrm{METF}_{\mathrm{TM}}$, and $\mathrm{METF}_{\mathrm{ML}}$, corresponding, respectively, to the METF for the shortest path and for the TM and ML attacker behaviors. The numbers indicated on the graphs refer to the complete list of security events, given in Appendix A.

When the number of paths between the attacker and the target increases, the size of the attack state graph associated with assumption TM may grow very fast (depending on the topology of the privilege graph) and our tools are not always able to build it. This precluded the computation of measure $\mathrm{METF}_{\mathrm{TM}}$ in some cases. Thus, the thick line curves in the two graphics present some gaps (e.g., in November 1995).

For each significant measure variation, the cause has been analyzed and a detailed description is given in Appendix A. Many of the events mentioned in Appendix A are indicated on the figures for convenience. The events marked in gray correspond to events that did not have a strong impact on the measures for the given objective, but implied a noticeable variation for the other objective.

3.5 Experiment Feedback

In the following, we analyze a subset of the events included in Appendix A to check the assumptions and expected behaviors discussed in Section 2.

Some events deserve to be analyzed more precisely in order to verify the various assumptions made on $\mathrm{METF}_{\mathrm{TM}}$ and $\mathrm{METF}_{\mathrm{ML}}$: no. 2, no. 7, no. 11, no. 16, no. 21, no. 29, and no. 56 indicated on Fig. 5 and Fig 6. We will also inquire further on events no. 6, no. 13, no. 29, and no. 79 that had a great impact on the evolution of the measures as well as on the number of paths.

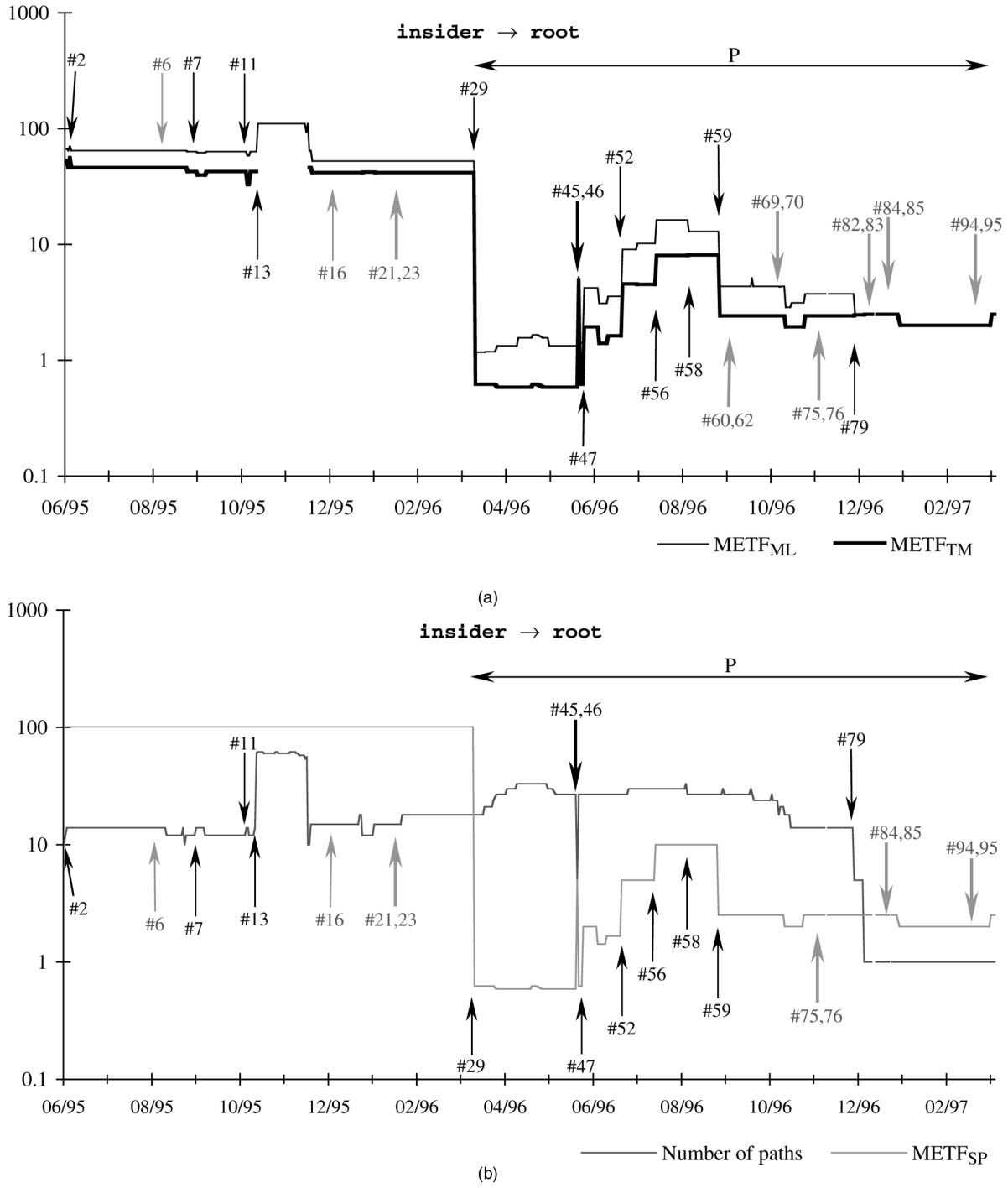


Fig. 5. Measures evolution for objective 1.

3.5.1 Events no. 2, no. 7, no. 11 for Objective 1 and no. 21 for Objective 2

Events no. 2, no. 7, and no. 11 for objective 1, and event no. 21 for objective 2 exhibit a global behavior of type 1 (see Table 1). Each of these events satisfies the conditions in which such behavior should be observed: They add one new vulnerability to those already available to the attacker to reach the `root` or the `admin_group` target, therefore increasing the total number of possible paths between the attacker node and the target. Furthermore, the shortest path does not evolve because these new paths are not shorter than the previous shortest one. More precisely, the vulnerabilities corresponding to these events are described in Table 4 (extracted from Table 5).

As can be seen in Fig. 5 and Fig. 6 and in the detailed table of Appendix A, $\mathrm{METF}_{\mathrm{TM}}$ always decreased as the result of occurrence of each of these events, showing a degradation of security. The amplitude of this evolution is variable (depending on the difficulty related to the new vulnerability and on its relative position with respect to previously existing paths). In fact, this has been verified for

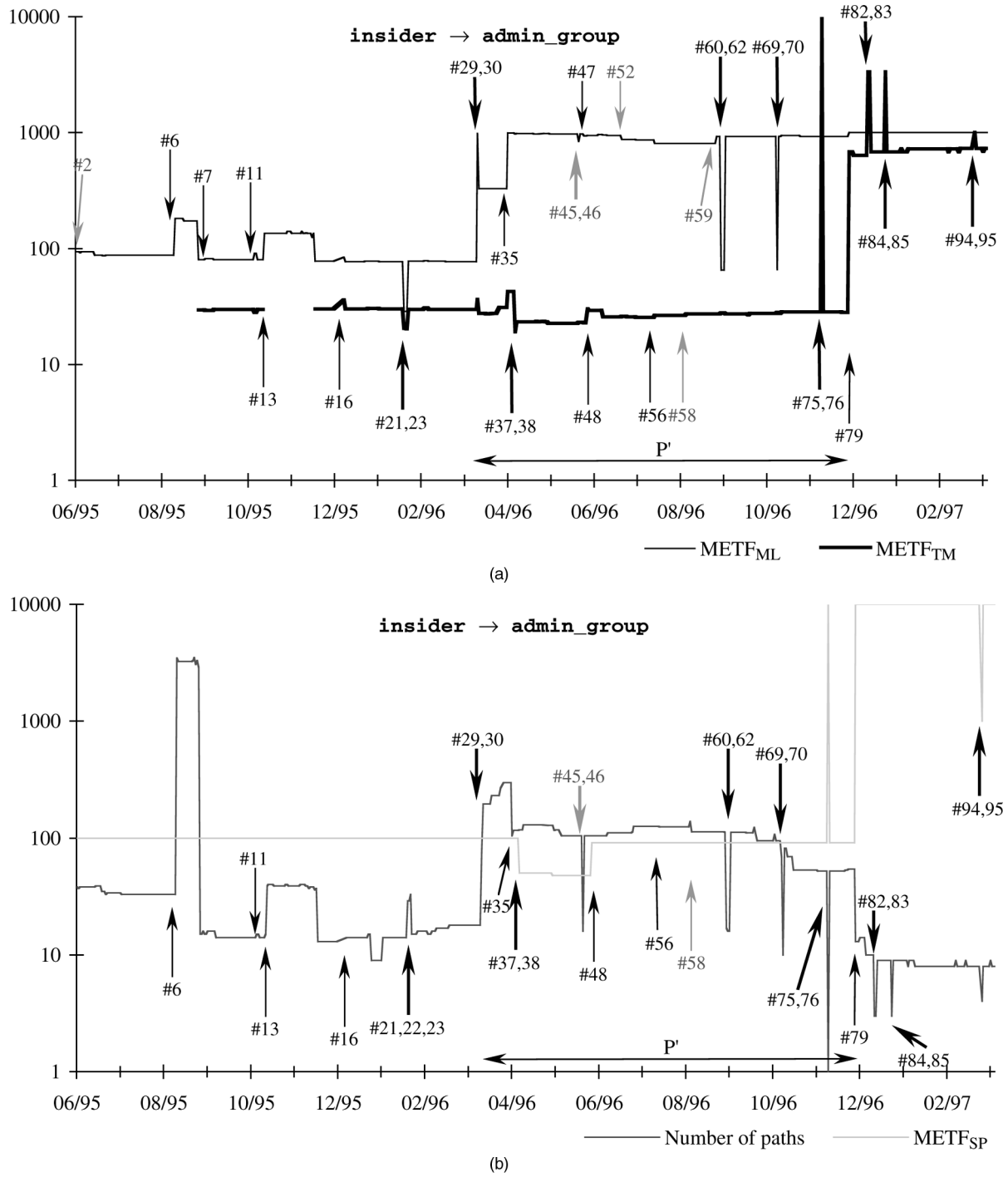


Fig. 6. Measures evolution for objective 2.

$METF_{TM}$ on every single degradation of the security, but, sometimes, the relative variation of this measure is very small and is not visible on the plots. Therefore, in the whole experiment, the behavior of $METF_{TM}$ was in agreement with the expectations. Moreover, for each of the events no. 2, no. 7, no. 11, and no. 21, $METF_{ML}$ evolution is similar to the evolution of $METF_{TM}$.

3.5.2 Events no. 11 and no. 56 for Objective 2

For objective 2, event no. 11 has a different impact on the security measures that illustrates behavior 2. In this case, an increase in the number of paths between the attacker and the target led to a decrease of $METF_{TM}$ and an increase of $METF_{ML}$. We expected that the $METF_{ML}$ and $METF_{TM}$ measures would not always evolve in the same direction. This happens when a secondary path appears that lengthens a previous path. It influences the $METF_{ML}$ that indicates an improvement by reducing the probability of selecting a fast path, while $METF_{TM}$, only affected by the fact that a new path has been created, indicates a degradation.

TABLE 4
Examples of Vulnerabilities Leading to Behavior 1

Objective	Event	Date	Problem
1	#2	18 Jun 95	One user grants write permissions to everyone for his home directory (allowing to implant a Trojan horse).
1	#7	7 Sep 95	Another user grants write permissions to everyone on his `.login` initialization file, allowing a major Trojan horse attack that would be activated at his next login.
1	#11	18 Oct 95	A third user grants write permission to everyone on his `.rhosts` file, enabling an immediate attack via the identity transfer mechanisms of Unix.
2	#21	30 Jan 96	A previously non-privileged user becomes a new member of the group `admin_group`.

The opposite variation appears for event no. 56 with respect to objective 2. In this case, a small decrease in the number of paths is associated with an increase of $\mathrm{METF}_{\mathrm{TM}}$ and a decrease of $\mathrm{METF}_{\mathrm{ML}}$. This corresponds to the fact that a secondary path disappears. Such events confirm that the evolution of both measures may be different in some circumstances due to the different assumptions made on the attacker behavior.

3.5.3 Event no. 29 and Period P for Objective 1

At the beginning of March 1996, a strong decrease of the shortest path between the attacker and the target occurred (event no. 29, objective 1). Fig. 5 shows that $\mathrm{METF}_{\mathrm{ML}}$ and $\mathrm{METF}_{\mathrm{TM}}$ decrease as a result of this evolution.

The period P that followed event no. 29 also exhibits an interesting behavior. During this period, it can be seen by comparing the two curves of Fig. 5 that $\mathrm{METF}_{\mathrm{TM}}$ was nearly equal to $\mathrm{METF}_{\mathrm{SP}}$.[3] The influence of the shortest path is so important here that its length directly controls the value of $\mathrm{METF}_{\mathrm{TM}}$ and the behavior of $\mathrm{METF}_{\mathrm{ML}}$. We are in the case where it is possible for the attacker to reach the target in a few very easy steps.

In such a situation, it is clear that the target is not well protected. Furthermore, as its security is directly affected by the vulnerabilities appearing in the shortest path, it would be mandatory to react and disable such vulnerabilities. However, in practice, this situation lasted until the end of the experiment. As can be seen in Table 5, Table 6, and Table 7, even though some of the most dangerous vulnerabilities were disabled (see no. 45, no. 47, no. 52, no. 56, no. 79), not all of them were eliminated, and some of them reappeared (see no. 46, no. 59). The origin of this situation has not been completely clarified. However, it is probably due to the local installation of a new complex piece of software.

3.5.4 Events no. 29, no. 79, and Period P' for Objective 2

The impact of event no. 29 on objective 2 corresponds to an important increase of the number of paths between the attacker and the target `admin_group`. However, most of these new paths are secondary paths, usually difficult to exploit. This change in the topology of the privilege graph induces a major increase of $\mathrm{METF}_{\mathrm{ML}}$, while $\mathrm{METF}_{\mathrm{TM}}$ indicates a minor degradation. This behavior is similar to the one described in Section 3.5.2. We can see that the absolute values of the two measures differ significantly during all period P'. However, they both react similarly to security events occurring during this period. Event no. 79 corresponds to the correction of many vulnerabilities associated to this security objective (see in Section 3.7). At the end of period P', most of the secondary paths were eliminated.

3.5.5 Event no. 6 for Objective 2 and Event no. 13 for Objectives 1 and 2

On both Fig. 5 and Fig 6, we also observed a similar phenomena: sudden important increases of the number of paths (at November 1995 for objective 1, and at the end of August 1995 and November 1995 for objective 2). The problem involved was an incorrect positioning of the write permission for the `others` or `group` fields of Unix permissions on one user's important initialization files (`.rhosts` for event no. 6, `.profile` and `.xinitrc` for event no. 13). These specific events opened a path to the target for every user in the system and, thus, allowed the *insider* user to use any vulnerable user as a starting point.

This phenomenon is normal and is a very security-relevant event, but interferes with the evaluation of security for two reasons:

1. first, the dramatic increase in the number of paths precludes the computation of $\mathrm{METF}_{\mathrm{TM}}$ (that should have shown a decrease in security);
2. second, all the new paths being longer than the previous ones, $\mathrm{METF}_{\mathrm{ML}}$ increases. In fact, the "insider" attacker is much more likely to choose a long path and spend a lot of effort in the system before reaching the target, and the $\mathrm{METF}_{\mathrm{ML}}$ is sensitive to that.

However, in these cases, the dramatic increase of the number of paths between the attacker and the target indicates directly that thorough security analysis must be performed.

3. In fact, the difference between measures TM and SP is very small ($\sim 10^{-3}$). Of course, this is not directly visible on the plot.

3.5.6 Event no. 16 for Objective 2

In order to validate our modeling assumptions, we do not consider the simultaneous occurrence of several modifications of the privilege graph (addition or deletion of vulnerabilities or modification of the rates assigned to the vulnerabilities). Different simultaneous modifications may influence diversely the system security and, thus, it is difficult to predict the type of behavior to be observed for the security measures. For instance, the consequence of event no. 16 on objective 2 seems to contradict the conclusions of Section 3.5.1. It shows an increase of the number of paths between the attacker and the target while both security measures $\mathrm{METF}_{\mathrm{TM}}$ and $\mathrm{METF}_{\mathrm{ML}}$ increase. But, this evolution is due to the occurrence of several security events within the period of one observation of the system (one day). Such a situation occurred more than once during the experiment.

In fact, when looking more closely at the definition of event no. 16 in Table 5, it can be seen that three vulnerabilities have been disabled for two different users and that one user has enabled a new one. The first two events should have a positive influence on security while the last one should have an opposite effect (it increases the number of paths). The evolutions of the measures seem to indicate that the last one has the least impact.

3.5.7 Consecutive Symmetrical Events

Some of the events indicated on Fig. 5 and Fig. 6 are grouped in pairs. These events correspond to opposed modifications in the system, occurring within a few days of each other. Such events occur frequently and can have a significant impact on the security measures (see nos. 21-23, nos. 45-46, nos. 60-62, nos. 69-70, nos. 82-83, nos. 84-85, and nos. 94-95).

It is noteworthy that many of these events correspond to a security improvement followed shortly by a degradation. The improvement is generally due to some access rights modification performed by the users themselves. But, these corrections appeared inconvenient to these users and they canceled them. This observation has confirmed that some of the vulnerabilities existing on the target system correspond to functions that users need. Therefore, the efficient and persistent elimination of these specific vulnerabilities is not a simple problem and should involve a negotiation with the users who require specific functionalities, but may compromise security when trying to obtain them.

3.6 Comparison of the Various Measures

In addition to the results discussed in Section 3.5, we make, in the following, some other remarks about the comparison of the different measures shown in Fig. 5 and Fig. 6.

3.6.1 Shortest Path

During the period covered by the experiment, $\mathrm{METF}_{\mathrm{SP}}$ was less sensitive to system modifications than $\mathrm{METF}_{\mathrm{ML}}$ and $\mathrm{METF}_{\mathrm{TM}}$. This measure provides interesting information about the easiest path of the weighted graph representing the system; however, it is not dynamic enough to be useful for operational monitoring of the security evolution. As indicated in Section 2.5, in comparison with $\mathrm{METF}_{\mathrm{TM}}$, the value of the shortest path does not take into account the fact that several equivalent paths could be available. In fact, more than its length, it is the nature of this path and of the vulnerabilities involved that will be of interest to improve the security, as it is the path that seems to have the major impact on $\mathrm{METF}_{\mathrm{ML}}$ and $\mathrm{METF}_{\mathrm{TM}}$.

3.6.2 Number of Paths

The number of paths between the attacker and the target is a sensitive indicator (it varies a lot), but it seems difficult to use alone for operational security monitoring.

First, it can be noticed that a security event leading to a decrease or an increase in the number of paths between the attacker and the target does not necessarily lead to a significant variation of the other security measures (see nos. 1, 18, 19, 20, 22 of Table 5). Theoretically, it seems possible to ignore such security events that have a minor influence on the mean effort to be spent by an attacker to reach the target. We are in the situation where the impact of the addition or deletion of arcs in the privilege graph is relatively small compared to the global effort values, even if the number of paths varies.

On the contrary, we have identified some events that led to a significant evolution of $\mathrm{METF}_{\mathrm{ML}}$ or $\mathrm{METF}_{\mathrm{TM}}$, whereas the number of paths changed slightly. See nos. 2, 3, 4, 11, 12, 16, 17, 21, 23 in Table 5. We are, therefore, able to detect that these particular events have an important influence on the security of the system without significantly affecting the number of paths between the attacker and the target.

Globally, we can see that the number of paths existing between the attacker and the target is a measure that would raise a significant number of alarms, among which some may be relatively uninteresting. Moreover, not all important security events would raise an alarm. Consequently, this measure seems more difficult to use than $\mathrm{METF}_{\mathrm{ML}}$ and $\mathrm{METF}_{\mathrm{TM}}$ and it is less reliable.

3.6.3 $\mathrm{METF}_{\mathrm{TM}}$ and $\mathrm{METF}_{\mathrm{ML}}$

The measures $\mathrm{METF}_{\mathrm{ML}}$ and $\mathrm{METF}_{\mathrm{TM}}$ exhibit an interesting behavior, with stability periods separated by several variations. As can be seen in Appendix A, each of these variations can be related to a security-relevant event. However, we can also see that $\mathrm{METF}_{\mathrm{TM}}$ cannot be computed all the time, which is a main drawback, and that $\mathrm{METF}_{\mathrm{ML}}$ sometimes exhibits a delicate behavior in which it shows an increase of the effort needed by the attacker to reach the target when the number of paths between them increases (behavior 2).

However, it seems possible to rely on $\mathrm{METF}_{\mathrm{ML}}$ to reveal the degradation of the security of the target and to react adequately to the most significant security events (contrary to the number of paths).

Among all the measures, $\mathrm{METF}_{\mathrm{TM}}$ seems to exhibit the most plausible behavior. Additional work would be needed to reduce the complexity of the algorithm used to compute it and then to obtain the missing values for a complete comparison with $\mathrm{METF}_{\mathrm{ML}}$.

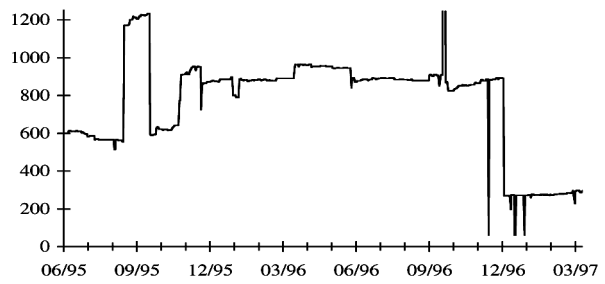


Fig. 7. Evolution of the total number of vulnerabilities.

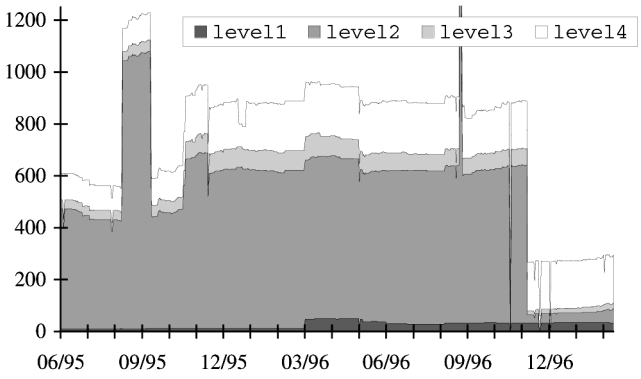


Fig. 8. Evolution of the distribution of vulnerabilities.

3.7 Comparison with Other Security Monitoring Tools

Usually, tools dedicated to operational security analysis, such as COPS or SATAN, limit their action to a direct checking of the system that ends with a list of all the known vulnerabilities present in it, possibly sorted in different classes according to their severity. As the prototype presented in this paper is heavily based on such tools (see ASA description in Section 3.1), we gathered these data and it is also interesting to compare the information provided on security at the end of this first step of the prototype and after the complete evaluation method is performed.

Fig. 7 plots the evolution of the number of known vulnerabilities in the system during the whole experiment. Fig. 8 shows the same results, but details the distribution of the vulnerabilities audited among the various severity levels considered.

First, we can note that a significant decrease in the total number of vulnerabilities has been observed in the target system on 4 December 1996. Some of these vulnerabilities, those related to objectives 1 and 2, appear in event no. 79 of Table 7. This evolution is due to a system wide operation performed by the system administrators on this date on the most sensitive configuration files of all system users in order to enforce restrictive access rights. The purpose of this action was to eliminate common user misconfigurations accumulated in the system. This action was initiated after the system administrators used some classical security analysis tool, and reviewed their reports. However, we can note that, even though many vulnerabilities were disabled, a significant number of them remain. Furthermore, the paths allowing a potential attacker to defeat either security objective 1 or 2 were not particularly targeted and the impact of this global operation on the security measures presented in Fig. 5 and Fig. 6 is variable. Only $\mathrm{METF}_{\mathrm{TM}}$ revealed a significant improvement for objective 2.

Second, if we were to use directly the information provided by Fig. 7 or Fig. 8 to monitor the security of the system, we can see that the number of alarms we would face is very significant. In fact, each time a new security event occurs in the system, we would be obliged to analyze it precisely, even if it is a minor event, because we do not know exactly its influence on the security objectives. Probably, to reduce the number of alarms, one would try to take into account the severity level of the new vulnerabilities involved. However, we can see, by comparing Fig. 8 and Fig. 5 and Fig. 6, that an evolution of the number of severe vulnerabilities (`level1` or `level2`) present in the system and a decrease of the overall security are not always correlated. Finally, we have seen previously that global corrections do not automatically bring significant improvements.

Of course, our intention is not to depreciate the value of the results obtained by such automatic tools. It is an essential first step to handle operational security monitoring and our evaluation method is mostly based on the data provided by these tools. However, and it is a well-known problem, the number of alarms raised by such tools is significant and they cannot always be taken care of easily. The evaluation measures presented in the previous section enable the security administrator to extract from all these variations the ones that would really need reaction. Therefore, the results obtained by our evaluation method are complementary to those derived from classical security analysis tools.

4 Conclusion

In this paper, we have presented an approach to the quantitative evaluation of the security of operational systems. The evaluation is based on a theoretical model, called the privilege graph, which describes the system vulnerabilities that may offer opportunities to potential attackers to defeat some security objectives. We have studied several modeling assumptions and discussed the validity of these assumptions based on an experimental study performed on a real system. Three different models are discussed corresponding to three assumptions about the attacker behavior: SP, TM, and ML. The experimental results show that both assumptions TM and ML are satisfactory because the behavior of the corresponding measures provides useful feedback to the security administrators in order to monitor the security of their system, i.e., evaluate the impact of the system vulnerabilities on the security objectives and identify the vulnerabilities which may lead to security degradation. Unfortunately, the security measure associated with assumption TM cannot always be computed due to the complexity of the algorithm. On the other hand, the computation of the measure related to assumption ML is easier. However, these two measures correspond to different assumptions on the attacker behavior. In an operational system, both measures should be used when possible, and any of their variation should be carefully analyzed. Finally, it

TABLE 5
Analysis of Security Events for Objectives 1 and 2, Events 1-28

				Objective 1			Objective 2		
Date	#	Users involved	Description	ΔNP	ΔML (%)	ΔTM (%)	ΔNP	ΔML (%)	ΔTM (%)
1995									
17 Jun	1	U1	U1's password becomes guessable.	+2	~0	~0	-1	~0	—
18 Jun	2	U2	U2's home directory writable.	+2	-3	-13	+1	+1	—
19 Jun	3	U3, U4	Potential Trojan horse attack on U3's `.login` and U4's `.tcshrc` becomes less probable (change of starting shell).	0	+9	+22	0	-2	—
20 Jun	4	U3, U4	(Opposite of #3.)	0	-8	-18	0	+2	—
29 Jun	5	U21	U21's home directory becomes writable.	0	0	0	+1	-7	—
24 Aug	6	U12, many others	U12 provides attack means to nearly six hundred other users.	0	0	0	+3445	+108	—
7 Sep	7	U5	Trojan horse attack on U5's `.login` becomes possible.	+2	-2	-8	+244	~0	—
9 Sep	8	U13, U14	U13's password is no longer guessable. U14 changes the false group ownership of some of his initialization files. (This disabled #6).	0	0	0	-2867	-54	—
14 Sep	9	U6	U6's home directory writable.	+2	-2	-6	1	+3	-1
20 Sep	10	U6	(Opposite of #9.)	-2	+2	+7	-2	-2	+1
18 Oct	11	U7	U7's `.rhosts` becomes writable	+2	-8	-27	+1	+13	-5
20 Oct	12	U7	(Opposite of #11.)	-2	+8	+29	-1	-12	+6
25 Oct	13	U5	Trojan horse attacks on some of U5's initialization files become possible for members of his group.	+46	+74	—	+24	+70	—
29 Nov	14	U5,U8	U5 removed from U8's `.rhosts`. (This disabled #13.)	-46	-41	—	-24	-43	—
1 Dec	15	U9	New attack possible via a system file.	+5	-19	-9	0	0	0
19 Dec	16	U15, U16, U17	Some trojan horse attacks possible on U15 and U16 disabled. U17's password becomes guessable.	0	0	0	+1	+8	+20
20 Dec	17	U15, U16	Attacks on U15 and U16 are now possible again. (Same vulnerability as formerly disabled in #8.)	0	0	0	0	-8	-17
1996									
3 Jan	18	U10	U10's password guessable.	+3	~0	~0	1	~0	~0
5 Jan	19	U1, U10, U15, U16, U17, U18, U19, U20	Objective 1: U10's and U1's password are no longer guessable. (Opposite of #1 and #18.) Objective 2: U1, U10, U15, U16, U17, U18, U19, U20 passwords are no longer guessable. (This corrects one problem in #16.)	-6	~0	~0	-6	+1	+2
13 Jan	20	U1, U10, U15, U16, U17, U18, U19, U20	Objective 1: U1's password guessable. (Same vulnerability as #1.) Objective 2: U1, U10, U15, U16, U17, U18, U19, U20 passwords are again guessable. (Same problems as in #19 and #16.)	+3	~0	~0	+5	-1	-2
30 Jan	21	`root`	`root` becomes a member of `admin_group`.	0	0	0	+15	-63	-31
1 Feb	22	U11	U11's `.mailrc` init file writeable.	+3	~0	-1	+4	~0	~0
2 Feb	23	`root`	`root` is no longer a member of `admin_group`. (Opposite of #21).	0	0	0	-18	+170	+47
7 Feb	24	U55	Objective 2: U55 is a new target user and his password is guessable.	0	0	0	+1	~0	~0
13 Feb	25	U25	Objective 2: a potential Trojan horse is disabled by U25 (who changed is command path search settings). U25 is no longer a target.	0	0	0	-1	+1	+3
16 Feb	26	U25	(Opposite of #25.)	0	0	0	+1	-1	-3
19 Feb	27	U40	Objective 2: U40 is a new target user and his password is guessable.	0	0	0	+1	~0	~0
28 Feb	28	U56	Objective 2: U56 is a new target user and his password is guessable.	0	0	0	+1	~0	~0

is concluded that the shortest path, the number of vulnerabilities and the number of paths are not sufficient to characterize the operational security evolution.

The experimental results presented in this paper and the modeling assumptions considered constitute a preliminary investigation about the feasibility of security evaluation of operational systems, taking into account their dynamic evolution, and about the benefits of these kinds of evaluations. Currently, work is in progress at LAAS to extend the monitoring tools used in this

TABLE 6
Analysis of Security Events for Objectives 1 and 2, Events 29-56

				Objective 1			Objective 2		
Date	#	Users involved	Description	ΔNP	ΔML (%)	ΔTM (%)	ΔNP	ΔML (%)	ΔTM (%)
21 Mar	29	U2, U3, U11, U15, U16, U20, U22, U23, U25, U26, root	Objective 1: Attacks on U2, U3 or U11 disabled. (Opposite of #2, #3 and #22 respectively.) New attack possible via a system file for U22. U23's password guessable. Several direct attacks on root are possible for everyone. Objective 2: Attacks on U15 and U16 are now disabled again. (Same vulnerability as formerly re-enabled in #17.) Attacks on U2, U3, U11 and U20 are disabled. U20 is no longer a target. Vulnerable user U23 acquires more privileges (new member of a group) and his password becomes guessable. U25 is vulnerable to a trojan horse attack made by U26.	0	-98	-98	+57	+1173	+22
22 Mar	30	U15, U16	Attacks on U15 and U16 are now enabled again. (Same vulnerability as in #16, #17 and #29.)	0	0	0	+120	-66	-24
28 Mar	31	U27	U27's password becomes guessable.	+3	~0	-2	+37	~0	~0
29 Mar	32	U52	Objective 2: U52's password is no longer guessable. U52 is no longer a target.	0	0	0	-1	~0	+2
3 Apr	33	U28	U28's password becomes guessable.	+3	~0	-2	+37	~0	~0
5 Apr	34	U32, U24, root	Objective 1: U24's home directory writable. One direct attack on root enabled. Objective 2: U32 is no longer a target. U24's home directory writable. One direct attack on root enabled.	+3	+13	~0	+30	+1	+12
11 Apr	35	U15, U16	Attacks on U15 and U16 are now disabled again.	0	0	0	-194	+200	+37
12 Apr	36	U1	U1's password becomes guessable.	+3	~0	~0	+12	~0	-1
16 Apr	37	U33, U34	U33 and U34 are new target users. U33 and U34 are vulnerable to some trojan horse attacks on their initialisation files.	0	0	0	+2	-2	-55
17 Apr	38	U33	U33 is no longer vulnerable (cf #37).	0	0	0	-1	+1	+22
19 Apr	39	U6	U6's home directory writable.	+3	+16	~0	+12	~0	-1
29 Apr	40	root	One direct attack on root is disabled.	0	+6	+6	0	0	0
6 May	41	root, U55	(Opposite of #40.) *Objective 2* U55's password is no longer guessable. U55 is no longer a target user.	0	-6	-6	-1	~0	+2
9 May	42	U34	*Objective 2* U34 disables two potential trojan horse attacks.	0	0	0	0	~0	-2
10 May	43	U6	U6's home directory no longer writable.	-3	-14	~0	-12	~0	+1
15 May	44	U1, U35	*Objective 1*: U1's password no longer guessable. *Objective 2*: U35 is no longer a target user.	-3	~0	~0	-13	~0	~0
30 May	45	root, U9, U29, U30, U22, U26	Most direct attack on root are disabled. Neither U9, U22, U26, U29 nor U30 can now attack it via an (improbable) trojan horse.	-22	+292	+740	-88	-14	+2
31 May	46	root, U9, U29, U30, U22, U26	(Opposite of #45.)	+22	-73	-87	+88	+17	~0
3 Jun	47	root	Most direct attack on root are disabled.	0	+196	+68	0	-4	-1
5 Jun	48	U34	*Objective 2* U34 disables one potential trojan horse attacks (idem #42).	0	0	0	0	+2	+32
13 Jun	49	root	Two direct attacks on root are disabled.	0	-26	-28	0	+1	~0
16 Jun	50	U32	U32 is again a target and his home directory is now writable by member of some other group than his own.	0	0	0	+7	-1	-12
19 Jun	51	root	One direct attack on root is disabled again.	0	+15	+16	0	~0	~0
29 Jun	52	root, U26	Most remaining direct attack on root are disabled. Most attack from U26 to root are disabled.	0	+154	+129	0	-8	~0
3 Jul	53	U36, U37	U36 and U37 are new target users. U36's and U37's passwords are guessable.	0	0	0	+2	~0	~0
4 Jul	54	U31	U31's password becomes guessable.	+3	~0	~0	+13	~0	~0
9 Jul	55	U31, U26	U31's password no longer guessable. U23's home directory becomes writable.	0	+13	~0	0	~0	-1
22 Jul	56	root	One direct attack on root is disabled.	0	+59	+76	-1	-7	+4

TABLE 7
Analysis of Security Events for Objectives 1 and 2, Events 57-80

				Objective 1			Objective 2		
Date	#	Users involved	Description	ΔNP	ΔML (%)	ΔTM (%)	ΔNP	ΔML (%)	ΔTM (%)
12 Aug	57	U38, U39	U38 is a new user whose password is guessable. *Objective 2*: U39 is a new target user. U39's password is guessable.	+3	~0	~0	+14	~0	~0
13 Aug	58	U4 ,U38	U4's account is terminated. U38's password is no longer guessable.	-6	+20	+1	-26	~0	+3
3 Sep	59	root, U39, U40	Three direct attacks on root are enabled. *Objective 2*: U39's password is no longer guessable. U40 is a new target user. U40's password is guessable.	0	-67	-70	0	+15	~0
6 Sep	60	U25, U26, U41	U41's password is guessable. *Objective 2*: U25 is no longer vulnerable to a Trojan horse attack made by U26 (cf #29).	+3	~0	~0	-95	-93	+2
7 Sep	61	U41	U41's password is no longer guessable.	-3	~0	~0	-1	~0	~0
9 Sep	62	U25, U26	*Objective 2*: U25 is again vulnerable to a Trojan horse attack made by U26 (cf #60).	0	0	0	+96	+1322	-2
25 Sep	63	U6	U6's home directory writable.	+3	+18	~0	+13	~0	-2
26 Sep	64	U17, U24, U28, U42, U43, U44,	U42's is a new user whose password is guessable. U24's home directory is writable. U28's password is guessable. *Objective 2*: U17's password is guessable. U43 and U44 are new target users and their passwords are guessable.	-3	-15	~0	-16	~0	+2
27 Sep	65	U42	U42's password is no longer guessable.	-3	~0	~0	-13	~0	~0
9 Oct	66	U45	U45 is a new user whose password is guessable.	+3	~0	~0	+13	~0	~0
10 Oct	67	U45	U45's password is no longer guessable.	-3	~0	~0	-13	~0	~0
14 Oct	68	U27, U46	U27 and U46 accounts are terminated (their passwords were guessable).	-6	~0	~0	-26	~0	~0
15 Oct	69	U25, U26, U47	U47's password is guessable. *Objective 2*: U25 is no longer vulnerable to a Trojan horse attack made by U26 (cf #62).	+3	~0	~0	-59	-93	+2
16 Oct	70	U25, U26	*Objective 2*: U25 is again vulnerable to a Trojan horse attack made by U26 (cf #69).	0	0	0	+72	+1317	-2
18 Oct	71	root, U6	U6's home directory is no longer writable. One direct attack on root is enabled.	-3	-34	—	-13	+2	+3
22 Oct	72	U29	U29's home directory is writable by members of his group.	-4	+10	~0	-16	~0	+7
31 Oct	73	root, U22, U48	One direct attack on root is disabled (cf #71). Now, only U48 can attack U22 via a Trojan horse place in his home directory. However, U48's password is guessable.	0	+19	+24	0	-1	~0
8 Nov	74	U40	U40's password is no longer guessable (cf #59).	0	0	0	-1	~0	~0
15 Nov	75	root, U9, U21, U22, U23, U25, U26, U30, U32, U34, U48, U49, U50, U51	Multiple vulnerabilities concerning the initialization files and home directory permissions of several users disabled. Several atttacks on root disabled.	0	0	0	-51	+977	+35164
16 Nov	76	*Idem #75*	(Opposite of #75.)	0	0	0	+51	-91	-99
27 Nov	77	U52	*Objective 2*: U52 is a new user and his password is guessable.	0	0	0	+1	~0	~0
29 Nov	78	U53	*Objective 2*: U53 is a new user and his password is guessable.	0	0	0	+1	~0	~0
4 Dec	79	root, U9, U22, U21, U23, U48, U30, U34, U49	One vulnerability on root opened to U22 is disabled. U48 can no longer attack U22 (cf #73). Several Trojan horse attacks possible on U9, U23 and U30 are disabled. *Objective 2*: U21 and U49 home directories are no longer writable. Some incorrect permissions of U34 initialization files are corrected.	-9	-34	+2	-41	+8	+2311
7 Dec	80	U54	U54 is a new user and his password is guessable.	0	0	0	+1	~0	-6

TABLE 8
Analysis of Security Events for Objectives 1 and 2, Events 81-97

				Objective 1			Objective 2		
Date	#	Users involved	Description	ΔNP	ΔML (%)	ΔTM (%)	ΔNP	ΔML (%)	ΔTM (%)
11 Dec	81	U26, U50, U51	U51 can no longer attack U26 and U50 via a Trojan horse. U51 is no longer vulnerable to a Trojan horse attack.	-4	+1	+1	-4	~0	~0
17 Dec	82	root, U25, U26, U32, U50, U53	Several vulnerabilities concerning U25, U26, U32, U50, U53 are disabled. Several direct attacks on root are disabled.	0	0	0	-7	+233	+422
19 Dec	83	root, U25, U26, U32, U50	(Opposite of #82, except for U53.)	0	0	0	+6	-70	-80
29 Dec	84	root, U25, U26, U32, U50	(Opposite of #83.)	0	0	0	-6	+233	+390
30 Dec	85	root, U25, U26, U32, U50	(Opposite of #84, vulnerabilities are finally left enabled.)	0	0	0	+6	-70	-80
1997									
4 Jan	86	root	One direct attack on root is enabled.	0	-20	-20	0	0	0
9 Jan	87	U47	U47's account is terminated.	0	0	0	-1	~0	+6
10 Jan	88	U57	*Objective 2*: U57 is a new user, his password is guessable.	0	0	0	+1	~0	-6
14 Jan	89	U57	(Opposite of #88.)	0	0	0	-1	~0	+6
11 Feb	90	U58	*Objective 2*: U58 is a new user, his password is guessable.	0	0	0	+1	~0	-6
12 Feb	91	U58	(Opposite of #90.)	0	0	0	-1	~0	+6
18 Feb	92	U1	*Objective 2*: U1's password is guessable.	0	0	0	+1	~0	-6
19 Feb	93	U52	*Objective 2*: U52's password is no longer guessable.	0	0	0	-1	~0	+7
1 Mar	94	U1, U19, U32, U54, U59	*Objective 2*: Passwords of U1, U19, U54 and U59 are no longer guessable. U32 is no longer a target user.	0	0	0	-4	~0	+38
2 Mar	95	U1, U19, U32, U54, U59	(Opposite of #94.)	0	0	0	+4	~0	-27
7 Mar	96	root, U60	One direct attack on root is disabled. *Objective 2*: U60 is a new user and his password is guessable.	0	+25	+25	+1	~0	-6
8 Mar	97	U60	*Objective 2*: U60's password is no longer guessable	0	0	0	-1	~0	+6

experiment to analyze a networked Unix system. Modern systems contain many different interconnected computers and each of these computers may exhibit local vulnerabilities, e.g., associated with locally defined user accounts. Global users accounts may still exhibit global vulnerabilities available through numerous workstations. Finally, the vulnerabilities related to networking functionalities, as well as the trust relationships between the different computers, are an important factor with an increasing influence on the opportunities available to an attacker. A new version of ESOPE currently provides a convenient architecture for performing a detailed analysis of a system composed of numerous workstations. This architecture is based on a meta-object approach that successfully provides distribution transparency, and allows monitoring programs to execute on remote computers with minor modifications. ESOPE also allows the computation of the security measures presented in this work. The integration of additional network-related vulnerabilities will be addressed soon. A graphical user interface based on the Amulet toolkit [15] is also currently developed. An important issue that remains to be addressed is the extension of this tool to heterogeneous systems, to take into account the vulnerabilities of different operating systems. This tool will bring additional experimental results concerning the security measures evolution in such a complex computer system. Such results will help to improve the accuracy of the measures considered in order to improve our confidence in them. We hope this tool will help the security administrators in better monitoring the security of their systems.

Appendix A

Detailed Analysis of Security Events

A detailed description of each cause of a major security measure variation appearing in Fig. 5 and Fig. 6 is given in Table 5, Table 6, Table 7 and Table 8. Not all the security events that have occurred during the experiment are

addressed here. We only considered those that led to a significant evolution of one of the measures. In this table, ΔNP designates the absolute variation of the number of paths, and ΔML and ΔTM the relative variation of $METF_{ML}$ and $METF_{TM}$. "—"means that the value was not computable, and "~ 0" that the relative variation is less than 0.5 percent.

ACKNOWLEDGMENTS

The authors are grateful to Marc Dacier, now at IBM Zürich Research Laboratory, for his remarks on an early version of this paper and for his pioneering work on quantitative evaluation of security. We wish to thank the anonymous referees for their useful comments which helped to significantly improve this article. We also acknowledge the support of the Computing and Instrumentation service at LAAS, in particular, Marc Vaisset and Matthieu Herrb. This work was partially supported by UAP Assurances and by the European ESPRIT Project 20072 "Design for Validation" (DeVa).

REFERENCES

[1] R. Ortalo, Y. Deswarte, and M. Kaâniche, "Experimenting with Quantitative Evaluation Tools for Monitoring Operational Security," *Proc. Dependable Computing for Critical Applications 6 (DCCA'6),* M. Dal Cin, C. Meadows, and W.H. Sanders, eds., Grainau, Germany, Mar. 1997.

[2] M. Dacier, "Towards Quantitative Evaluation of Computer Security," doctoral thesis (in French), Institut National Polytechnique de Toulouse, no. 971, LAAS Report 94488, Dec. 1994.

[3] M. Dacier, Y. Deswarte, and M. Kaâniche, "Models and Tools for Quantitative Assessment of Operational Security," *Proc. 12th IFIP Information Systems Security Conf. (IFIP/SEC'96),* pp. 177-186, S.K. Katsikas and D. Gritzalis, eds., Samos, Greece, May 1996.

[4] R. Ortalo, "Quantitative Evaluation of Information Systems Security," doctoral thesis (in French), Institut National Polytechnique de Toulouse, no. 1418, LAAS Report 98164, May 1998.

[5] T. Olovsson, E. Jonsson, S. Brocklehurst, and B. Littlewood, "Towards Operational Measures of Computer Security: Experimentation and Modelling," *Predictably Dependable Computing Systems,* B. Randell, J.-C. Laprie, H. Kopetz, and B. Littlewood, eds., pp. 555-569. Berlin: Springer-Verlag, 1995.

[6] M. Dacier and Y. Deswarte, "Privilege Graph: An Extension to the Typed Access Matrix Model," *Proc. Third European Symp. Research in Computer Security (ESORICS'94),* pp. 317-334, D. Gollman, ed., Brighton, U.K., Lecture Notes in Computer Science 875, Springer-Verlag, 1994.

[7] S. Gupta and V.D. Gligor, "Experience with a Penetration Analysis Method and Tool," *Proc. 15th Nat'l Computer Security Conf.,* NIST & NCSC, vols.1/2, pp. 165-183, Baltimore, Oct. 1992.

[8] A.D.E. Muffet, "Crack Version 4.1—A Sensible Password Checker for Unix," publicly available by ftp with the crack4.1 software at ftp.cert.org, 1992.

[9] B. Littlewood, S. Brocklehurst, N. Fenton, P. Mellor, S. Page, D. Wright, J. Dobson, J. McDermid, and D. Gollmann, "Towards Operational Measures of Computer Security," *J. Computer Security,* vol. 2, nos. 2/3, pp. 211-229, 1993.

[10] D. Anderson, T. Lunt, H. Javitz, A. Tamaru, and A. Valdes, "Safeguard Final Report: Detecting Unusual Program Behavior Using the NIDES Statistical Component," SRI Project 2596, Dec. 1993.

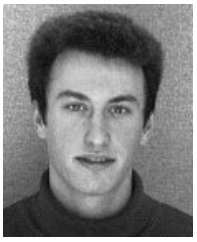
[11] ITSEM, *Information Technology Security Evaluation Manual,* 262 p., Office for Official Publications of the European Communities, Luxembourg, 1993.

[12] ITSEC, *Information Technology Security Evaluation Criteria,* 163 p., Office for Official Publications of the European Communities, Luxembourg, 1991.

[13] D. Farmer and E.H. Spafford, "The COPS Security Checker System," *Proc. Summer Usenix Conf.,* 971 p., Anaheim, Calif., 1990.

[14] S. Garfinkel and E. Spafford, *Practical Unix & Internet Security.* O'Reilly & Assoc., 1996.

[15] B.A. Myers, R.G. McDaniel, R.C. Miller, A.S. Ferrency, A. Faulring B.D. Kyle, A. Mickish, A. Klimovitski, and P. Doane, "The Amulet Environment: New Models for Effective User Interface Software Development," *IEEE Trans. Software Eng.,* vol. 23, no. 6, pp. 347-365, June 1997.



Rodolphe Ortalo received the certified engineer degree from Supélec in 1994, the diploma for further studies in computer science from the University of Orsay in 1994, and a PhD degree in computer science from the National Polytechnic Institute of Toulouse (INPT) in 1998. He prepared his PhD dissertation at LAAS-CNRS, in the Fault Tolerance and Dependable Computing Research Group, under the direction of Yves Deswarte. This research work addresses the problem of quantitative evaluation of information systems security. Dr. Ortalo is currently performing a one-year postdoctoral study at ONERA, in collaboration with Frédéric Cuppens. His current research interests include formal security policy specification issues, vulnerabilities description, and intrusion detection.

Yves Deswarte received the certified engineer degree from ISEN, Lille, in 1972 and the computer science specialization engineer degree from the ENSAE, Toulouse, in 1973. He is currently Directeur de Recherche at LAAS-CNRS (currently on sabbatical leave at Microsoft Research in Cambridge, United Kingdom). Formerly, he was an R&D engineer at a major French computer manufacturing company, and joined INRIA and LAAS-CNRS in 1979. He has been a member of the LAAS research group on fault tolerance and dependable computing since 1984. Since 1973, he has been a main contributor to the design of many fault-tolerant computing systems. His research activities are devoted to dependable computing systems and, more precisely, to the design of fault-tolerant and secure distributed systems, protection of distributed object systems and security evaluation. He is the author of more than 50 publications in the domain of fault-tolerance and security. He is chair of the SEE Technical Committee on Dependable Computing Systems and has chaired the International Steering Committee of the European Symposium on Research in Computer Security (ESORICS). He is a member of the ACM and an affiliate of the IEEE Computer Society.

Mohamed Kaâniche received the certified engineer degree from the National School of Civil Aviation of Toulouse, France, in 1987, the diploma for further studies in computer science and automation from the University of Toulouse in 1988, and a PhD degree in computer science from the National Polytechnic Institute of Toulouse (INPT) in 1992. He is currently Chargé de Recherche of CNRS, the French National Organization of Scientific Research. He joined LAAS-CNRS in 1988 as a member of the research group on fault tolerance and dependable computing. From 1997-1998, he was a visiting research assistant professor at the University of Illinois at Urbana-Champaign (on leave from LAAS). His current research activities focus on dependability modeling and evaluation of fault-tolerant computing systems, software reliability growth evaluation, and operational systems' security assessment. He has published more than 30 papers on these subjects in international journals and conferences. Dr. Kaâniche is an affiliate of the IEEE Computer Society.