Experimenting Quantitative Evaluation Tools for Monitoring Operational Security

Rodolphe Ortalo
ortalo@laas.fr

Yves Deswarte
deswarte@laas.fr

Mohamed Kaâniche
kaaniche@laas.fr

LAAS-CNRS & INRIA
7, avenue du Colonel Roche
31077 Toulouse cedex 4
France

Abstract

This paper presents the results of an experiment of security evaluation. The evaluation method used is based on previous work involving modeling the system as a privilege graph exhibiting the security vulnerabilities and on the computation of measures representing the difficulty for a possible attacker to exploit these vulnerabilities and defeat the security objectives of the system. A set of tools has been developed to compute such measures and has been experimented to monitor a large real system during more than a year. The experiment 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.

Keywords: quantitative security evaluation, privilege graph, Markov modeling.

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; powerful applications whose complexity may hide serious security flaws; etc.). On the other side, 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 system 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 here presented aims at facilitating this administrator's task: by providing a quantitative assessment of the current system security level, these tools can help him to identify those security flaws which can be eliminated for the best security improvement with the least incidence to users. Such quantitative evaluation tools should also enable him to monitor the evolution of the global system security with respect to modifications of the environment, of the configurations, of the applications or of the 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 up to: 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 security measure should evolve according to system modifications influencing its security: any modification can bring new vulnerabilities and/or correct previous ones and the security measure should be sensitive to such modifications. The main use of such measures is to monitor 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.

A theoretical framework has been developed at LAAS to identify and compute such measures [Dacier 1994, Dacier *et al*. 1996]. This framework is based on: 1) a theoretical model, the privilege graph, exhibiting the system vulnerabilities, 2) a definition of the security objectives, 3) a mathematical model based on Markov chains to compute the security measures. To demonstrate the practical 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: what is the validity of the obtained measures to represent accurately the 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 inefficient since such attacks are not necessarily representative of real attacks, and because for a good accuracy, tiger team attacks must be numerous on a stable system [Olovsson *et al*. 1995] while our measures are intended to rate the dynamic evolution of the 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.

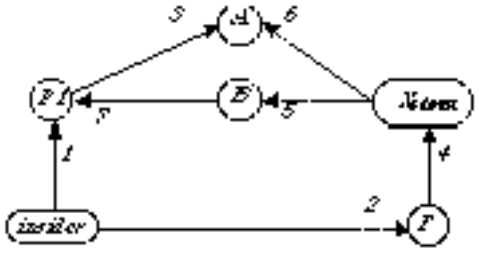
This paper presents this experiment. Section 2 presents a short description of the theoretical framework. Section 3 presents the experiment itself and discusses the results. Finally, Section 4 draws a conclusion.

2. Presentation of the Approach

2.1 Formal Description of Operational System Vulnerabilities

It has been shown in [Dacier and Deswarte 1994] 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 from node X to node Y indicates that a method exists for a user owning X privileges to obtain those of node Y. A vulnerability represented by an arc can be a direct security flaw, such as an easily guessed password or bad directory and file protections enabling the implantation of a Trojan horse. But the vulnerability is not necessarily a security flaw. Instead, it can result from the use of a feature designed to improve security. For instance, in Unix, the `.rhosts` file enables a user U1 to 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 protection). But if U2 grants some privilege to U3 (U2 is trusting U3), then by transitivity, U3 can reach U1's privileges, even if U1 does not trust U3. A third class of arcs is the representation of privilege subsets directly issued from the protection scheme, e.g., with Unix groups, 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.

Figure 1 gives an example of such a privilege graph with arcs being labeled by vulnerability classes.



1) X can guess Y's password; 2) X is in Y's `.rhosts`; 3) Y is a subset of X; 4) X can attack Y via Email; 5) Y uses a program owned by X; 6) X can modify a `setuid` program owned by Y.

Figure 1 Example of a privilege graph.

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. For example, we can define a node called “insider” which represents the minimal privileges of any registered user (e.g., the privilege to execute login, to change his own password, etc.). If a path exists between an attacker node and a target node, then a security breach can potentially occur since, by transitivity, a possible 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 most of them cannot be exploited by an attacker. 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` [Muffet 1992] 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 searching. 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 request 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 to be 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. To assess this measure, each arc in the privilege graph can be 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 is encompassing several characteristics of the attack process such as pre-existing attack tools, time needed to perform the attack, computing power available for the attacker, etc. [Brocklehurst *et al*. 1994]. For example, the effort needed to obtain a password can be assessed by the computing power and the time needed by `crack` to identify the password. For a Trojan horse attack, the effort can be assessed as the competence needed to design the Trojan horse, the time needed to implant it in a program which 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.

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

2.2 Operational Security Models and Measures

2.2.1 Assumptions

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. First, we assume that the attacker is sensible and he will not attempt an attack which would give him privileges he already possesses. Additional assumptions are required to characterize the progress of the attacker towards 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, i.e. the one which has the lowest mean value of cumulated effort. The shortest path can be evaluated directly from the privilege graph taking into account the rates assigned to the arcs. However, this assumption implicitly means that the attacker knows in advance the whole topology of the privilege graph. But, to build the whole privilege graph the attacker needs all the sets of privileges described in the graph. If the attacker already has these privileges, he does not need to build the graph! Clearly, the shortest path assumption is not satisfactory.

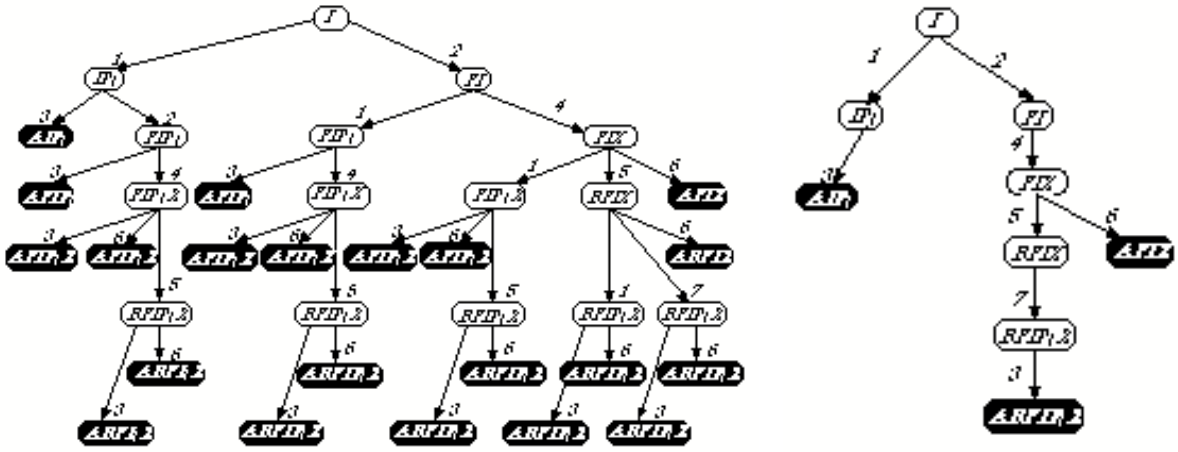
The attacker's privileges increase as a result of his progress towards the target can be characterized by a state-transition graph where each state identifies the set of privileges that he has already gained and transitions between states occur when the attacker succeeds in an attack allowing him to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

Total memory (TM) assumption: at each step of the attack process, 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 attack among the set of possible attacks.

Memoryless (ML) assumption: at each newly visited node of the privilege graph, the attacker chooses one of the 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 process stops when the target is reached. We do not consider situations where attackers may give up or interrupt their process.

Figure 2 plots the state graph attack process associated with the example given in Figure 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 Figure 2-b correspond to a subset of those identified in Figure 2-a.



(a) TM assumption (b) ML assumption

Figure 2 Attack process state graph associated to the example of Figure 1

2.2.2 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 which satisfies some intuitive properties regarding security evolution (see [Dacier *et al*. 1996] for further details). The Markov model is based on the assumption that the probability to succeed in a 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 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 clarifications. 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 state transition attack process 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 METF (Mean Effort To security Failure, by analogy with Mean Time To Failure). 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's output transition rates:

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

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:

$$METF_k = E_k + \sum_{i \in out(k)} P_{ki} \times METF_i \ ; P_{ki} = \lambda_{ki} \times E_k \quad (2)$$

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

2.2.3 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.

2.2.3.1 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 Figure 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.

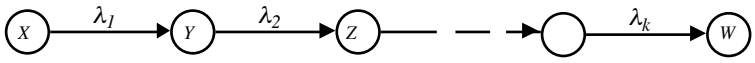


Figure 3 Markov model corresponding to a single path

2.2.3.2 Multiple Paths

As regards the SP assumption, the shortest path is obtained by identifying in the privilege graph all the direct paths 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 $METF_{SP}$ is:

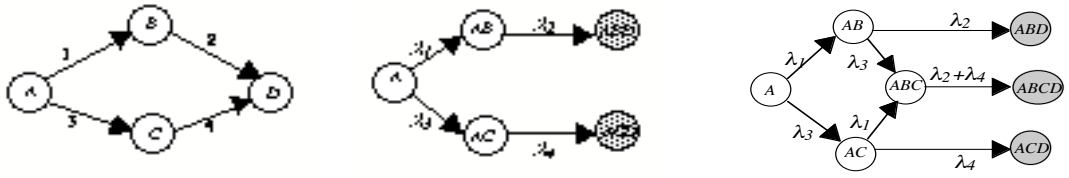
$METF_{SP} = \min \{U_1, \ldots, U_n\}$ where $U_k = \sum 1/\lambda_i$, λ_i is the rate assigned to the arc i that belongs to direct path k, n is the number of direct paths.

The METF values corresponding to assumptions TM or ML can be obtained by processing the corresponding state transition attack process. Let us consider the example of Figure 4 where A is the attacker and B is the target. The privilege graph (Figure 4-a) indicates the

presence of two paths leading to the target. The Markov models corresponding to assumptions ML and TM are given in Figure 4-b and 4-c respectively. Application of equations (1) and (2) leads to the following expressions:

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

$$METF_{TM} = \frac{1}{\lambda_1 + \lambda_3} + \frac{\lambda_1}{\lambda_1 + \lambda_3} \times [\frac{1}{\lambda_2 + \lambda_3} + \frac{\lambda_3}{\lambda_2 + \lambda_3} \times \frac{1}{\lambda_2 + \lambda_4}] + \frac{\lambda_3}{\lambda_1 + \lambda_3} \times [\frac{1}{\lambda_1 + \lambda_4} + \frac{\lambda_1}{\lambda_1 + \lambda_4} \times \frac{1}{\lambda_2 + \lambda_4}]$$



(a) Privilege graph (b) Assumption ML (c) Assumption TM

Figure 4 Multiple paths—example

It could be seen that, for any value of λ_1, λ_2 and λ_3, the expression of $METF_{TM}$ 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 $METF_{TM}$ which indicates security degradation. This result can be easily generalized, further details can be found in [Dacier 1994].

However, assumption ML leads to a different behavior since $METF_{ML}$ may increase or decrease depending on the values of the parameters. For instance, $METF_{ML}$ 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 contrarily 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 $METF_{ML}$ decreases, otherwise the $METF_{ML}$ increases.

From the previous discussion, it can be concluded that $METF_{TM}$ is always lower than the mean effort calculated based on the shortest path only ($METF_{TM} \leq METF_{SP}$). For assumption ML, $METF_{ML}$ may be lower or higher than $METF_{SP}$ depending on the values of the parameters and the structure of the privilege graph. It is noteworthy that, for both assumptions ML and TM, the shortest path is always the major contribution to the final value of the METF due to the fact that the probability that the shortest path is selected by the attacker is higher than the probability of choosing another path.

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

2.2.4 Discussion

Based on the results of the previous section, Table 1 summarizes the expected behavior for measures $METF_{ML}$ and $METF_{TM}$ when: (1) the number of paths between the attacker and the target set of privileges increases, or when (2) the shortest path between them decreases.

In both cases, it is important to note that 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. In operational systems, it is frequent that only one modification of the privilege graph occurs at a time. If the privilege graph is constructed each time a modification occurs, then it is likely that the typical behaviors reported in Table 1 will be always satisfied.

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, $METF_{TM}$ decreases since this new path weakens the security of the target,
- as discussed in the previous section, two kinds of behavior may be observed for $METF_{ML}$:
  - if the new path decreases the probability of taking another relatively easy path to the benefit of a longer new one, $METF_{TM}$ may increase (behavior 1.2).
  - otherwise, $METF_{ML}$ should have the same evolution as $METF_{TM}$: it should decrease as the number of paths increases and reveal a degradation of security (behavior 1.1).
- when the shortest path between the attacker and the target decreases, both measures $METF_{TM}$ and $METF_{TM}$ decrease and show a degradation of security (behavior 2).

	$METF_{TM}$	$METF_{ML}$	Behavior
Number of Paths ↗	↘	↘	1.1
Number of Paths ↗	↘	↗	1.2
$METF_{SP}$ ↘	↘	↘	2

Table 1 Typical behaviors

Clearly, assumption TM allows easier interpretation of security evolution. By analyzing the variation of this measure 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).

As regards assumption ML, the increase of $METF_{ML}$ when the number of paths increases may be considered as unrealistic as it could mean that the security increases when additional vulnerabilities are introduced. This kind of measure behavior is due to the ML attacker profile which assumes that when the attacker chooses a given path he never leaves it. If the top

events appearing in the selected path correspond to easy attacks, then the attacker is inclined to choose this path. Then, if the following attacks require too much effort to succeed, the mean effort to reach the target will increase. The main question is whether this type of attacker profile is realistic or not. It is difficult to answer this question because of lack of real data. In the experiment presented in the following section, we will show that valuable information about security evolution can be provided to security administrators even when only model ML is considered. Indeed, as we are mainly interested in METF variation rather than in the absolute values of this measure, any significant variation of the METF has to be thoroughly examined.

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

3.1 Tools Description

The experiment presented in this section has been conducted using a set of tools. The main steps of the evaluation process are:

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. So far, ASA is using many procedures included in the COPS package [Farmer and Spafford 1990]. More precisely, like in 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 may be recorded in an archive. This archive will be regularly augmented by using classical Unix tools such as `cron` to allow automatic and periodic analysis of the system.
3) **Evaluation:** Subsequently, another tool computes the security measures presented in Section 2.2 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 sets 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 Annex A.

3.2 Target System Description

The system under observation in this experiment is a large distributed computer system of more than a 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 have changed frequently due to the arrival and departure of temporary staff, a frequent event for the target system. The probing of security vulnerabilities is made on the global file system. In this experiment, the system has been observed during 13 months on a daily basis, starting in June 1995 until the end of July 1996. The archive of privilege graphs contains 385 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 might worry about it for personal reasons, or sometimes system administrators for safety reasons. This explains the important 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 convince the users to remove the vulnerabilities we had identified to improve the system security.

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 one would like the system to fulfill as much as possible. On a Unix system, one important target to protect against attacks is the `root` account, and more generally, every account allowing to obtain superuser[1] privileges. Another interesting target to study is one particular Unix group: the group of all system administrators, giving access to all the data they share. To select a precise attacker we choose the "insider" set of privileges defined in section 2. Table 2 summarizes these case studies.

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

[1] In Unix, the superuser privilege bypasses the access control mechanisms.

Table 2 Security objectives

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, objective 2 would have included objective 1, as if one sequence of privilege transfer methods enables to defeat 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.

3.3.2 Vulnerabilities

From all the known vulnerabilities in Unix, we monitored 13 among 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 [Spafford and Garfinkel 1991]. In addition to this, specific arcs labeled "instantaneous" correspond to inclusions 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 to one vulnerability changes (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 (see Table 3), where the different levels differ from each other by one order of magnitude, to rate the different vulnerabilities: `level1` corresponds to the easiest elementary attacks, and `level4` to the most difficult ones.

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

Table 3 Attack success rate levels

In this experiment, the various levels assigned to each attack are rather arbitrary. Evaluating precisely the success rate of the various attacks present in the system 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 the security measures behavior rather than precisely rate the security of the system.

3.4 Security Measures

3.4.1 Results

The different results of this experiment corresponding to objectives 1 and 2 are presented in Figures 5 and 6 respectively. The measures presented are: the number of paths found between attacker and target sets, $METF_{SP}$, $METF_{TM}$ and $METF_{ML}$.

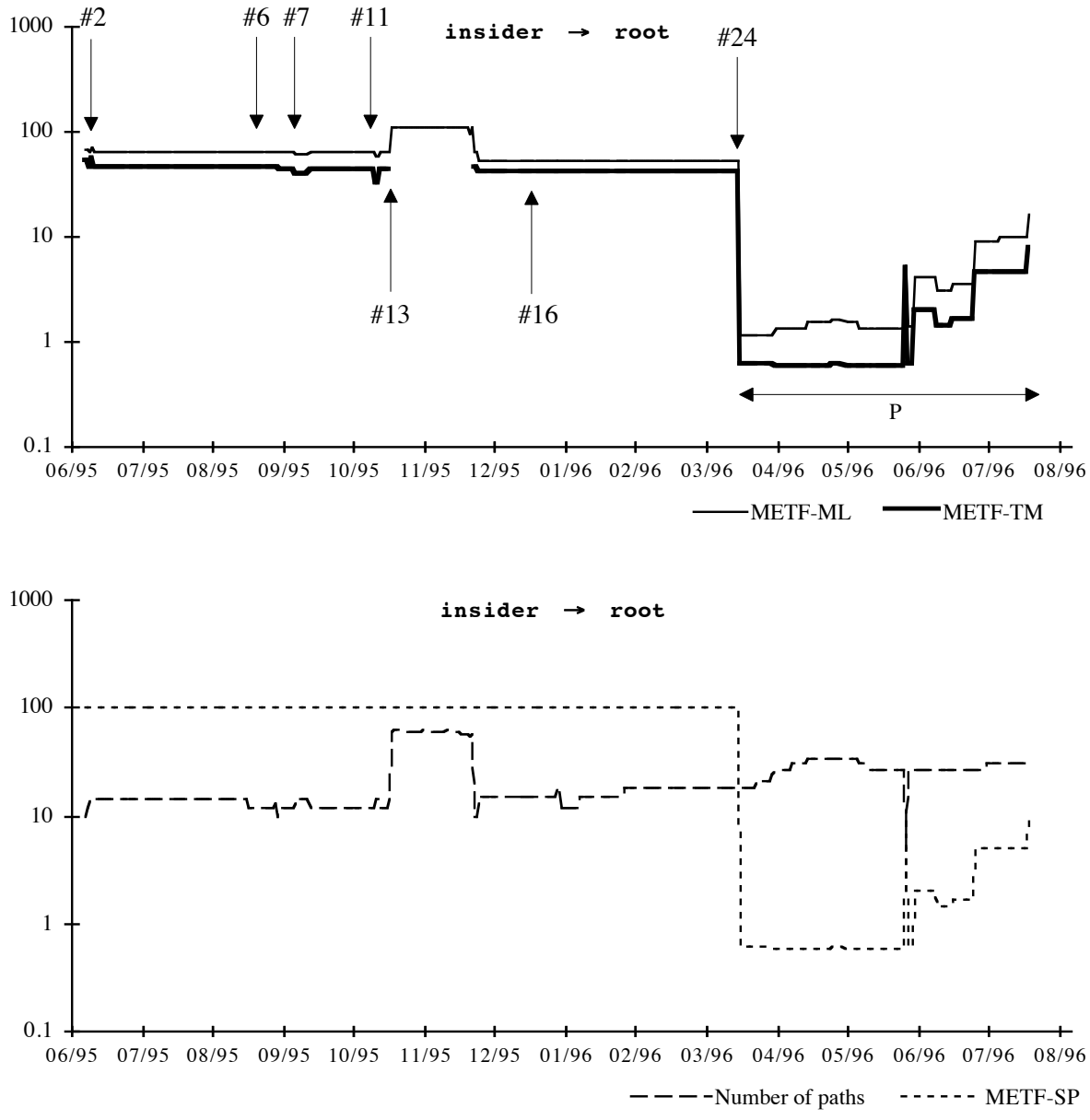


Figure 5 Measures evolution for objective 1

$METF_{TM}$ can only be computed when the number of paths between the attacker and the target is relatively small. Thus, the thick line curves in the two graphics sometimes present gaps due to the uncomputability of this measure (unfortunately, very large gaps appear in

Figure 6). For each significant measure variation, the cause has been analyzed and a detailed description is given in Annex A.

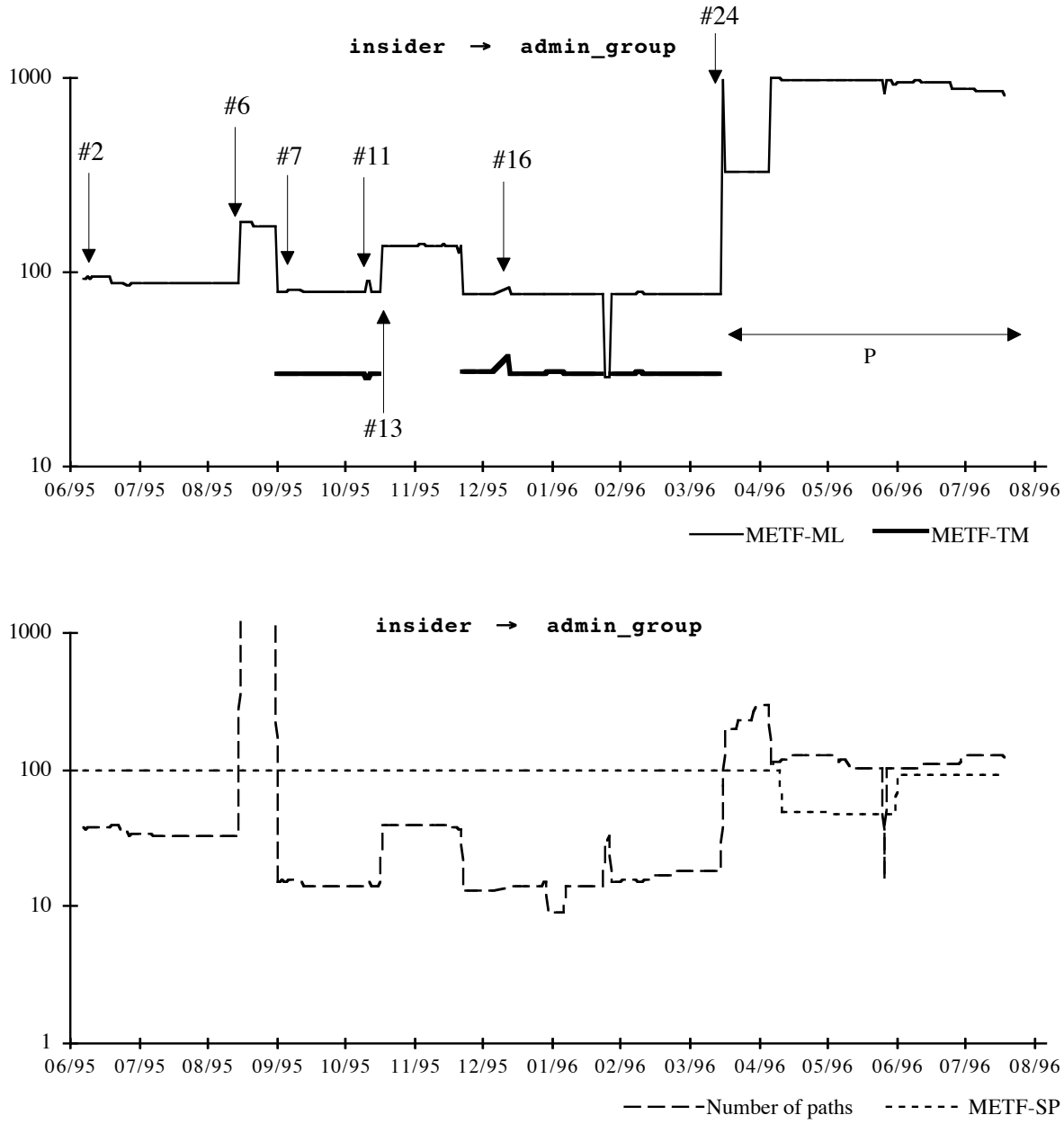


Figure 6 Measures evolution for objective 2

3.4.2 Experiment Feedback

Some events deserve to be analyzed more precisely in order to verify the various assumptions made on $METF_{TM}$ and $METF_{ML}$: #2, #7, #11, #16 and #24, indicated on Figures 5 and 6. We will also inquire further events #6 and #13 that had a great impact on the evolution of the measures as well as on the number of paths.

3.4.2.1 Events #2, #7 and #11 for Objective 1

For objective 1, the events #2, #7 and #11 exhibit a global behavior of type 1.1 (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` 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 A-1):

Event	Date	Problem
#2	18 Jun 95	One user grants write permissions to everyone for his home directory (allowing to implant a Trojan horse that careless users could activate for example).
#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.
#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.

Table 4 Examples of vulnerabilities leading to behavior 1.1

As can be seen in Figures 5 and 6 and in the detailed table of Annex A, in front of such events, $METF_{TM}$ always decreases, 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 $METF_{TM}$ 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 #2, #7 and #11, $METF_{ML}$ evolution is similar to the evolution of $METF_{TM}$.

3.4.2.2 Event #11 for Objective 2

For objective 2, the event #11 has a different impact on the security measures that illustrates behavior 1.2. In this case, an increase in the number of paths between the attacker and the target has lead to a decrease of the $METF_{TM}$ 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 shows 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, shows a degradation.

3.4.2.3 Event #24 and Period P for Objective 1

Event #24 is a good example illustrating behavior 2. At the beginning of March 1996, a strong decrease of the shortest path between the attacker and the target of objective 1 occurred. Figure 5 shows that $METF_{ML}$ and $METF_{TM}$ decrease as a result of this evolution.

Period P that followed event #24 also exhibits an interesting behavior. During this period, it can be seen by comparing the two curves of Figure 5, that $METF_{TM}$ was nearly equal to the

length of the shortest path[2]. The influence of the shortest path is so important here that its length directly controls the value of $METF_{TM}$ and the behavior of $METF_{ML}$. 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.

3.4.2.4 Event #6 and #13 for Objectives 1 and 2

On both Figures 5 and 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` field of Unix permissions on an important initialization file. This opened a path to the target for nearly every user in the system, and thus provided the "insider" user with as much intermediate initial paths as there were vulnerable users in the overall system.

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

- first, the dramatic increase in the number of paths precludes the computation of $METF_{TM}$ (that should have shown a decrease in security);
- second, all the new paths being longer than the previous ones, $METF_{ML}$ 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 $METF_{ML}$ is sensitive to that. This is a normal evolution of $METF_{ML}$, but may not be a satisfying indication of the overall security evolution.

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.

3.4.2.5 Event #16 for Objective 2

In order to validate the assumptions made on the behavior of the measures, we need to consider their evolution when only atomic events occur since the evolution cannot be predicted when several conflicting events occur. For instance, in the results relative to objective 2, event #16 seems to contradict the conclusions of Section 3.4.2.1: it shows an increase of the number of paths between the attacker and the target while both security measures $METF_{TM}$ and $METF_{ML}$ increase. But this evolution is due to the occurrence of several simultaneous security events within the period of one observation of the system (1 day). Such situation occurred more than once during the experiment.

In fact, when looking more closely to the different events that caused this evolution (Table A-1), 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

[2] In fact, measure TM was slightly smaller differing from the length of the shortest path by $\sim 10^{-3}$. Of course, this is not direclty visible on the plot.

number of paths). The evolutions of the measures seem to indicate that the last one has the least impact.

3.4.3 Comparison of the Various Measures

Apart from the conclusions derived from this experiment on the behavior of $METF_{TM}$ and $METF_{ML}$, we are also able to make a few other remarks relative to the comparison of the different measures shown in Figures 5 and 6.

3.4.3.1 Shortest Path

During all the period covered by the experiment, the shortest path evolved only 3 times (once for objective 1 and twice for objective 2). This measure is giving an interesting information about the easiest path present in the system, however it is not dynamic enough to be useful for operational monitoring of the security evolution. As indicated in Section 2.2.4, in comparison with $METF_{TM}$, 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 can be of interest to improve the security as it is the path which has the major impact on $METF_{ML}$ and $METF_{TM}$.

3.4.3.2 Number of Paths

The number of paths between the attacker and the target is a sensitive information (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 of the number of paths between the attacker and the target does not necessarily lead to a significant variation of the other security measures (see #1, 18, 19, 20, 22, of Table A-1). Theoretically, it seems possible to ignore such security events that have little influence on the mean effort to be spent by an attacker to reach the target. We are in the case 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 $METF_{ML}$ or $METF_{TM}$ whereas the number of paths changed slightly: see #2, 3, 4, 11, 12, 16, 17, 21, 23 in Table A-1. We are therefore able to detect that these particular events have an important influence on the security of the system while they do not significantly affect 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 an important number of alarms among which some may be relatively uninteresting. Moreover, not all important security events would lead to the raise of an alarm. Consequently, this measure seems more difficult to use than $METF_{ML}$ and $METF_{TM}$ and it is less reliable.

3.4.3.3 $METF_{TM}$ and $METF_{ML}$

The measures $METF_{ML}$ and $METF_{TM}$ exhibit an interesting behavior, with stability periods separated by several variations. As can be seen in Annex A, each of these variations can be related to a security-relevant event. However, we can also see that $METF_{TM}$ cannot be computed all the time, which is a main drawback; and that $METF_{ML}$ sometimes exhibit 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 1.2). This weakens the confidence we can have in $METF_{ML}$, all the more when a single security event such as #6 or #13 can lead to a great increase of that measure.

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

Among all the measures, $METF_{TM}$ 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 values that we miss here for a complete comparison with $METF_{ML}$.

3.4.4 Comparison with Other Security Monitoring Tools

Usually, tools dedicated to operational security analysis such as COPS or SATAN limit their action to a direct check 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.

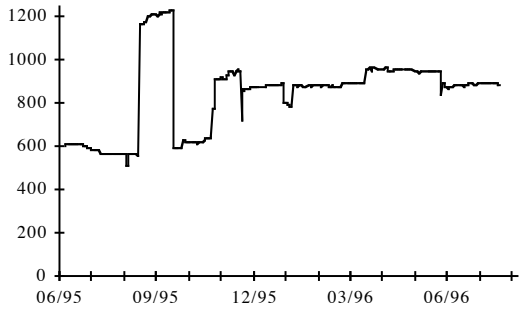


Figure 7 Evolution of the total number of vulnerabilities

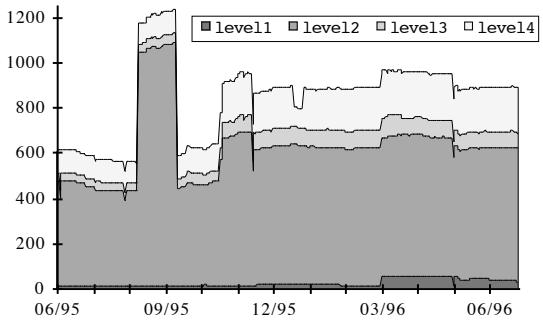


Figure 8 Evolution of the distribution of vulnerabilities

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

If we were to use directly the information provided by Figure 7 or 8 to monitor the security of the system, we can see that the number of alarms we would be faced to would be very important. In fact, each time a new security event occurs in the system, we would be obliged to analyze it more precisely, even if it is a minor event, because we do not know exactly its influence on the security objectives. Probably, in front of such alarms, one would try to take into account the severity level of the new vulnerabilities involved. However, we can see, by comparing Figures 8, 5 and 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. In fact, as the evolution of the number of vulnerabilities does not integrate

the position of the new ones in the paths from an attacker to a target, it is difficult to judge directly if the overall security of the target has really been compromised.

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 it forms the basis upon which our own evaluation is done. However, and it is a well known problem, the number of alarms raised by such tools is important, and all of them 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 aiming at 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 experiment results show that assumption TM is satisfactory because the behavior of the corresponding measure 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 to assumption TM can not be always computed due to the complexity of the algorithm. On the other hand, the computation of the measure related to assumption ML is easier. However, it is more difficult for the security administrators to identify the appropriate actions to be done on the system based on the observed behavior of this measure only. In fact, in this case, any variation of the security measure should be carefully analyzed whereas, for assumption TM, only negative variation of the measure can be considered. Finally, it 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. Further work is needed to improve the accuracy of the measures considered in order to improve our confidence in them, and help the security administrators in better monitoring the security of their systems.

References

[Brocklehurst *et al*. 1994] S. Brocklehurst, B. Littlewood, O. T. and E. Jonsson, “On Measurement of Operational Security”, in *9th Annual IEEE Conference on Computer Assurance (COMPASS'94),* (I. C. Society, Ed.), pp.257-266, Gaithersburg, 1994.

[Dacier 1994] M. Dacier, *Towards Quantitative Evaluation of Computer Security,* Doctoral Thesis, LAAS Report 94488 (in French), Institut National Polytechnique de Toulouse, December 1994.

[Dacier and Deswarte 1994]M. Dacier and Y. Deswarte, "The Privilege Graph: an Extension to the Typed Access Matrix Model", in *European Symposium in Computer Security (ESORICS'94),* (D. Gollman, Ed.), Lecture Notes in Computer Science, 875, pp.319-334, Springer-Verlag, Brighton, UK, November 1994.

[Dacier *et al*. 1996] M. Dacier, Y. Deswarte and M. Kaâniche, "Models and Tools for Quantitative Assessment of Operational Security", in *12th International Information Security Conference (IFIP/SEC'96),* (S. K. Katsikas and D. Gritzalis, Ed.), pp.177-186, Chapman & Hall, Samos (Greece), May 1996.

[Farmer and Spafford 1990]D. Farmer and E. H. Spafford, "The COPS Security Checker System", in *the Summer Usenix Conference,* Anaheim, CA, USA, 1990.

[Muffet 1992] 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.

[Olovsson *et al*. 1995] T. Olovsson, E. Jonsson, S. Brocklehurst and B. Littlewood, "Towards Operational Measures of Computer Security: Experimentation and Modelling", in *Predictably Dependable Computing Systems,* Basic Research Series, (B. Randell, J.-C. Laprie, H. Kopetz and B. Littlewood, Ed.), pp.555-569, ISBN N°3-540-59334-9, Springer-Verlag, Berlin, Germany, 1995.

[Spafford and Garfinkel 1991] E. Spafford and S. Garfinkel, *Practical Unix Security,* 483p., O'Reilly & Associates (Inc.), 1991.

Annex A - Detailed analysis of security state modifications

A detailed description of each cause of a major security measure variation appearing in Figure 5 and 6 is given in Table A-1. Not all the security events that have occurred during the experiment are addressed here: we only inquired into those that led to an important relative evolution of one of the measures. In this table, ΔNP, ΔML and ΔTM designate the absolute variation of the number of paths, and the relative variation of $METF_{ML}$ and $METF_{TM}$ respectively. "—" means that the value was not computable, and "~0" that the relative variation is less than 0.5%.

				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, and 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	—
1 Feb	22	U11	U11's `.mailrc` init file writeable.	+3	~0	-1	+4	~0	—
2 Feb	23	root	root is no longer a member of admin_group. (Opposite of #21).	0	0	0	-18	+170	—
21 Mar	24	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 Mar	25	U15, U16	Attacks on U15 and U16 are now enabled again. (Same vulnerability as in #16, #17 and #24.)	0	0	0	+120	-66	—
28 Mar	26	U27	U27's password becomes guessable.	+3	~0	—	+37	~0	—
3 Apr	27	U28	U28's password becomes guessable.	+3	~0	—	+37	~0	—
5 Apr	28	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	—
11 Apr	29	U15, U16	Attacks on U15 and U16 are now disabled again.	0	0	0	-194	+200	—
12 Apr	30	U1	U1's password becomes guessable.	+3	~0	~0	+12	~0	—
16 Apr	31	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	—
19 Apr	32	U6	U6's home directory writable.	+3	+16	~0	+12	~0	—
10 May	33	U6	U6's home directory no longer writable.	-3	-14	~0	-12	~0	—
15 May	34	U1, U35	*Objective 1*: U1's password no longer guessable. *Objective 2*: U35 is no longer a target user. U1's and U35's passwords are no longer guessable.	-3	~0	~0	-13	~0	—
30 May	35	`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	—
31 May	36	`root`, U9, U29, U30, U22, U26	(Opposite of #35.)	+22	-73	-87	+88	+17	—
3 Jun	37	`root`	Most direct attack on `root` are disabled.	0	+196	+68	0	-4	—
13 Jun	38	`root`	Two direct attacks on `root` are disabled.	0	-26	-28	0	+1	—
16 Jun	39	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	—
19 Jun	40	`root`	One direct attack on `root` is enabled again.	0	+15	+16	0	~0	—
29 Jun	41	`root`, U26	Most remaining direct attack on `root` are disabled. Most attack from U26 to `root` are disabled.	0	+154	+129	0	-8	—

3 Jul	42	U36, U37	U36 and U37 are new target users. U36's and U37's passwords are guessable.	0	0	0	+2	~0	—
4 Jul	43	U31	U31's password becomes guessable.	+3	~0	~0	+13	~0	—
9 Jul	44	U31, U26	U31's password no longer guessable. U23's home directory becomes writable.	0	+13	~0	0	~0	—
22 Jul	45	`root`	One direct attack on `root` is disabled.	0	+59	+76	-1	-7	—

Table A-1 Analysis of security events for objectives 1 and 2