Models and tools for quantitative assessment of operational security

M. Dacier[*]*, Y. Deswarte and M. Kaâniche*
LAAS-CNRS & INRIA
7 Avenue du Colonel Roche, 31077 Toulouse Cedex, France
Tel: +(33)61 33 62 00; Fax: +(33) 61 33 64 11
Email: dac@zurich.ibm.com; deswarte@laas.fr; kaaniche@laas.fr

Abstract
This paper proposes a novel approach to help computing system administrators in monitoring the security of their systems. The approach is based on modeling the system as a privilege graph exhibiting operational security vulnerabilities and on transforming this privilege graph into a Markov chain corresponding to all possible successful attack scenarios. A set of tools has been developed to support this approach and to provide automatic security evaluation of Unix systems in operation.

Keywords
Quantitative security evaluation, privilege graph, probabilistic assessment

1 INTRODUCTION

Computing system security relies mostly on users, operators and administrators, and even the best designed system, if badly operated, would be unsecure. Most authentication and protection mechanisms can be diverted by malicious or careless users, then allowing possible intruders to perform security breaches. This is not surprising since most users are less interested in security than in efficiency, flexibility and cooperation with other users.

A scrupulous system administrator should try to maintain the best security for his system with the least incidence on user operation. It is thus tremendously important to assess the operational system's security level and to monitor the evolution of this security level with respect to system configuration modifications, application or operation changes and environment evolution.

This paper develops an approach to evaluate the security of operational computing systems. Examples are taken from Unix™ systems, but the approach could be extended to other operating systems and to distributed systems as well. Section 2 shows how to model a computing system in a way exhibiting vulnerabilities that could be exploited by a possible

[*]Marc Dacier is now at IBM Zürich Research Laboratory, Säumerstrasse 4, CH-8803 Rüschlikon, Switzerland.

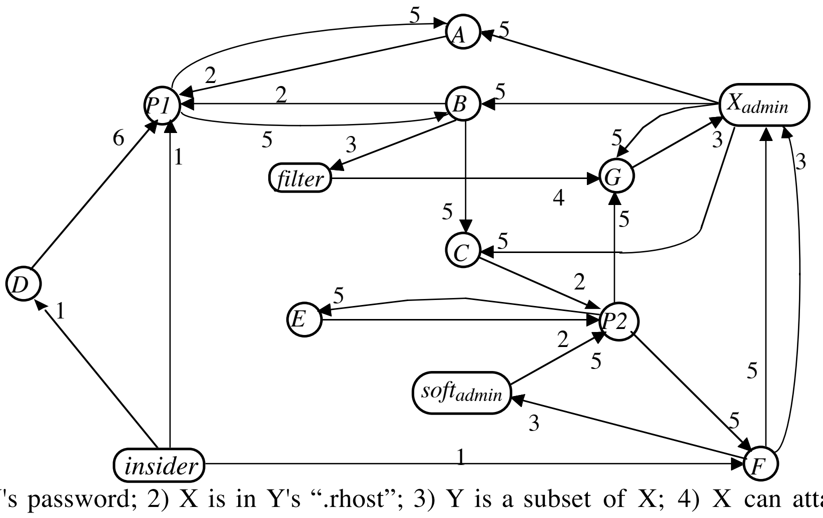
intruder. Section 3 presents a mathematical model which enables to compute probabilistic security measures corresponding to the difficulty for a possible intruder to perform a security breach. Finally, Section 4 describes a set of tools which have been developed to support this approach, and to provide automatic security evaluation of Unix systems in operation.

2 DESCRIPTION OF OPERATIONAL SYSTEM VULNERABILITIES

It has been shown in (Dacier and Deswarte, 1994) that vulnerabilities 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 represents a possibility for a user owning X privileges to extend his privileges to obtain those of node Y. Some sets of privileges are highly sensitive (e.g. the super user 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.

As a matter of fact, the vulnerability represented by an arc 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 to grant most of his privileges to another user without disclosing his password.

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



1) X can guess Y's password; 2) X is in Y's ".rhost"; 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.

3 A PROBABILISTIC APPROACH FOR SECURITY ASSESSMENT

The privilege graph highlights system vulnerabilities which an intruder can take advantage of. Some of the identified flaws have to be eliminated, however, some others may correspond to functionalities required by the users. In this situation, the security administrator must enhance the security of the system whilst preserving, as far as possible, its ease of use. In this sense, deciding which flaws must be eliminated and which ones can be ignored should be based on the evaluation of the risks induced by the residual flaws. Hence, it is important to develop a method which enables to evaluate the impact of residual flaws on operational security.

In the following, we present an example (the maze analogy) which outlines the main properties that we believe should be satisfied by the model chosen for security evaluation. Then, we propose a modeling framework for security evaluation which allows these intuitive properties to be satisfied.

3.1 The “maze” analogy

The privilege graph can be compared to a maze where the nodes correspond to rooms and each arc between two nodes corresponds to a door that allows access from one room to another. The doors are transparent so that one can see the content of the rooms that can be directly accessed from them. Each door can be opened in a single direction, and once opened it cannot be closed. The doors are equipped with locks of different qualities: each lock can be characterized by a parameter which determines the success rate of the breaking process applied to it. One of the rooms contains a target which may be accessed by an attacker placed in an another room of the maze. The attacker has no knowledge of the whole topology of the maze and he has to try to reach the target. However, the attacker knows all the mechanisms that are available to break the locks. He can apply one mechanism to each lock and wait until the lock is broken. During this waiting time he can apply other mechanisms to try to open other doors.

Let us assume that the attacker has two main qualities which characterize any human being:

- *Memory*: he remembers all the rooms he has already visited and the rooms that can be accessed from them.
- *Common sense*: he will not try to break a lock allowing access to a room he has already visited from another room.

Different situations are studied hereafter.

Direct path. Let us suppose that there is only one path leading to the target such that each room has only one door to get into the following room (direct path). Thus, the difficulty cumulated by the attacker to reach the target is proportional to the number of doors he has to open and also to the reliability of each lock he has to break. In this case, the cumulated difficulty can be estimated by adding the difficulties associated with each lock he has to break.

Multiple path. Let us assume that there are two direct paths leading to the target. Undoubtedly, the target is less protected than in the previous case due to the existence of the second path: the attacker can simultaneously try to break different locks. Then, his rate of success depends on the failure rates of the locks which are simultaneously addressed.

Major impact of the shortest path. The shortest path is the one which allows to reach the target with the lowest cumulated difficulty. It is likely that the least reliable lock will be easier to break than the others; then, the lowest cumulated difficulty up to the target is mainly given by the shortest path. However, since the attacker does not know the whole topology of the maze, he may decide to choose another path if he believes that it will lead him to reach the target. At each step where different possibilities are offered to him, he has to make a choice. Then, all the paths have to be considered to estimate the cumulated difficulty up to the target. Intuitively, the cumulated difficulty will be of the order of magnitude of the shortest path divided by the number of paths having this value.

Security evaluation: intuitive properties. The maze analogy allows us to derive some intuitive properties that should be satisfied by the quantification model to be chosen for security evaluation based on the privilege graph:
P1: Security increases if the "length of the paths" leading to the target increases.
P2: Security decreases if the "number of paths" leading to the target increases.
P3: Security is mainly affected by the shortest path leading to the target.

3.2 Quantification variables for security assessment

The first step towards security assessment is to define significant variables that can be used to characterize and evaluate security. As discussed in (Dacier *et al.*, 1995), we believe that "Effort" and "Time" spent by a potential attacker to reach the security target are sufficient to characterize the potential intrusion process. Indeed, some kinds of attacks, for instance Trojan Horses, require a small amount of effort to be spent by the attacker for their implementation, however a long period of time may pass before the attack is successful. Other attacks are complex, but once implemented their effect is immediate. Combinations of both kinds of attacks are also possible. As a consequence, we have chosen to derive evaluations with respect to both variables and then to analyse simultaneously the results obtained.

Two main properties are stated hereafter concerning security evolution with respect to "Time" and "Effort"; further justifications and examples can be found in (Dacier *et al.*, 1995):

- *Security is directly proportional to the time needed by an attacker to succeed in his attack.*
- *Security is directly proportional to the effort required for the implementation of an attack.*

Estimation of "Time" and "Effort". As most potential attacks rely on well known breaking techniques we believe that it is possible for security experts who have a deep knowledge of the system under study to build a database gathering identified breaking rules and then assign "Time" and "Effort" values to each attack.

As regards "Time estimation", it is noteworthy that the probability for an attack to succeed in a given period of time mainly depends on the target user profile. Tools are available for statistical estimation of user profiles based on the analysis of the users behaviour and of their interactions within the system. These tools have been developed for intrusion detection, for instance IDES (Lunt *et al.*, 1990) and more recently NIDES (Jagannathan *et al.*, 1993) which can be used in Unix environments. Then, we can use the statistical profiles provided by such tools in order to estimate the success rates relative to the attacks identified in the privilege graph, without relying only on security experts judgement. Moreover, these statistical profiles can be continuously updated in real-time which allows to observe dynamically the variation of estimated values of the success rates of the attacks.

For "Effort" estimation, it is possible to define a measurement scale allowing to order the different attacks according to the difficulty of implementing them. We propose a 4-level rating scale as an example, where Level 1 (the lowest level) corresponds to the least difficult attacks (well known attacks frequently used by intruders in reported detected intrusions), and Level 4 includes sophisticated attacks that require the intruder to be physically close to the target system. Note that, since our main objective is to define a quantification scale to observe security evolution, we do not need to have too many levels for ranking attacks. Moreover, having a reduced number of levels will allow easier interpretation of the security quantification results.

3.3 Intrusion process model

In the previous sections, we have mentioned that security may be affected if at least one path leading to the specified target exists, i.e., if at least one attack scenario can succeed. Nevertheless, we didn't clearly explain yet how these scenarios can be identified from the privilege graph. This is the aim of the following paragraph.

Assumptions

Let us consider the privilege graph given in Figure 2. Each arc in the graph refers to a specific attack. In order to derive all the scenarios of attacks that might be made by the user *"insider"* in order to reach the user A, we need to make assumptions about the behavior of this potential intruder. As real data characterizing intruders profiles are not available, we have derived a minimum set of assumptions which are based on common sense:

A1) A priori, the intruders do not know the whole topology of the privilege graph. They only know the attacks that can be directly applied in a single step.

A2) The intruders have a good memory. They remember all the sets of privileges they already acquired during the intrusion process.

A3) The intruders are sensible. They will not attempt an attack which would give them privileges they already have.

The last two assumptions are clearly realistic. The first assumption is justified by the fact that any one who wants to build the whole privilege graph needs all the set of privileges described in the graph. If the intruder already has these privileges, then he does not need to build the graph! Finally, we should note that the intrusion process is stopped when the target is reached.

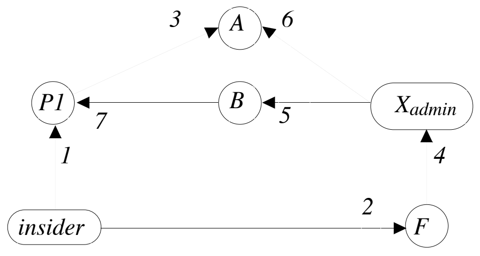


Figure 2 Example of a privilege graph.

Let us examine the scenarios of attacks that can be followed by the user "insider" to obtain user A's set of privileges. At the beginning, he knows that attacks 1 and 2 are possible. Suppose that he chooses attack 1; at the next step he has the choices between attacks 3 and 2; indeed based on assumption 2, the intruder remembers that he has the possibility to try attack 2 that he did not apply at the previous step. If at this step he successfully applies attack 3 then the target is reached and the intrusion process stops. However, if he tries attack 2 after 1, then he will have the same set of privileges as if he tried attack 2 followed by 1. At this step, he has the choice between attacks 4 and 3, etc.

Intrusion process state graph

Figure 3 plots the complete state graph characterizing the intrusion process derived from Figure 2. To improve the clarity of the figure, X_{admin} and *insider* are respectively referred to as *X* and *I*. States transitions occur when new privileges are obtained. The arcs between states are labelled by the names of the attacks corresponding to new privileges acquisition. When the process moves from one state to another one, the set of privileges indicated in the new state includes all the privileges obtained in the already visited states. Blacked states indicate that the target has been reached. It can be noticed that depending on the path followed by the intruder, there are some states in the graph from which some possible transitions have been discarded. This is due to assumption A3. For instance, from any state *BFIP1X* (no matter the scenario leading to it) only attacks 3 and 6 are considered; attack 7 is discarded as it corresponds to the acquisition of P1 privileges which have already been obtained by the intruder. Note that figure 3 can be simplified by grouping duplicated states. For instance state *FIP*$_1$ can be reached by applying the scenarios 1-2 or 2-1.

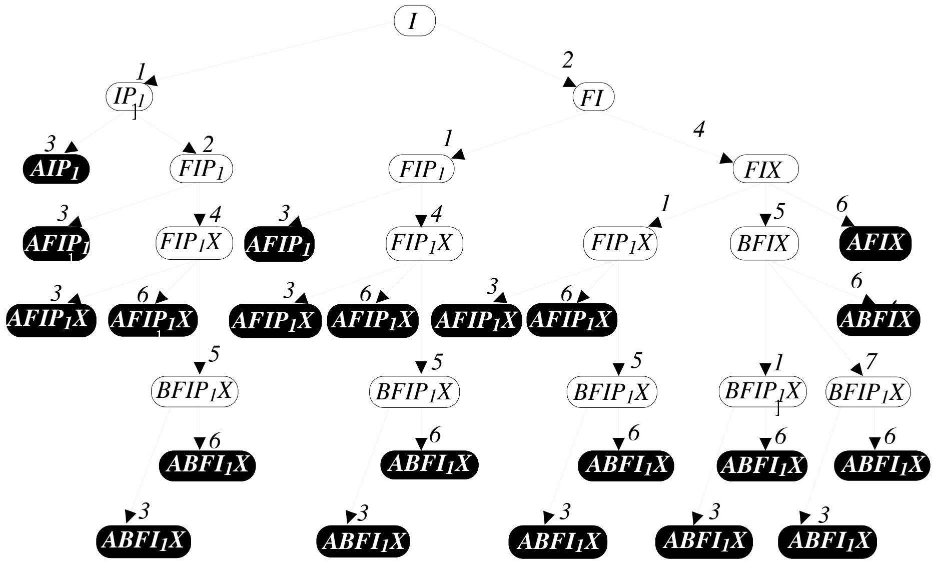


Figure 3 Intrusion process state graph summarising the scenarios of potential attacks.

3.4 Mathematical model for security quantification

The last step in our approach concerns the definition of a mathematical model that allows us to *evaluate the mean effort and the mean time to reach the target* (i.e., that takes into account the identified paths leading to the target). The choice of a mathematical model should undoubtedly be based on plausible assumptions in order to obtain significant results. Any model is an approximation of reality, with some unavoidable trade-offs between plausibility and usability. Our investigations led us to choose a Markovian model which satisfies the following requirements: 1) the results obtained are consistent with the intuitive properties regarding security evolution, and 2) the model is mathematically tractable. In the following we briefly outline some results which support this statement considering the “Time” variable. The same justifications also apply to the “Effort” variable.

The Markov model is based on the assumption that the probability to succeed in a given attack before time t is described by an exponential distribution given by: $P(t) = 1- \exp(-\lambda t)$. Practical considerations derived from the use of this distribution are the following:

- The potential intruder will eventually succeed in reaching the target, if a path leading to the target exists, provided that he spends enough time.
- The rate to succeed in a given attack is constant with respect to time.
- The mean time 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. These rates can be estimated as discussed in Section 3.2

Various probabilistic measures can be derived from Markov models, among these, the MTTF (Mean Time To Failure) which characterizes in our context the mean time for a potential intruder to reach the specified target. This metric allows easy physical interpretation of the results: the higher the MTTF the better the security. The analytical expressions that can be derived from the Markov model are detailed in (Dacier *et al.*, 1995). Moreover, it is proved that the intuitive properties presented in Section 3 are also satisfied.

4 EVALUATION TOOLS

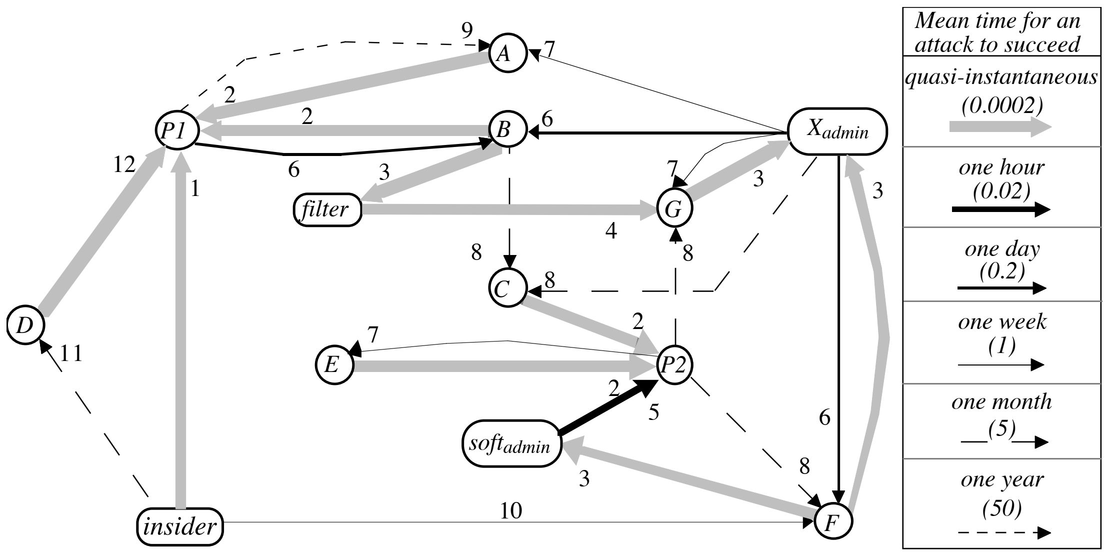
In order to show the feasibility of our approach, we have developed a prototype integrating a set of tools allowing the implementation of the different steps of the approach in order to evaluate the operational security of a Unix system:

1) **Building the privilege graph:** in a first step, we model the system by building its privilege graph. We have developed a tool, named *ASA* for Automatic Security Advisor, to construct automatically the privilege graph related to a Unix system. So far, ASA is using many procedures included in the COPS package (Farmer and Spafford, 1990).
2) **Definition of the security policy:** in this step, the security targets (*what must be protected*) and the potential attackers (*against whom*) are specified. This phase is facilitated by the use of a graphical language called Mirò (Heydon, 1992).
3) **Evaluation phase:** the privilege graph is transformed into a Stochastic Petri Net from which the Markov chain describing the potential intrusion state graph is automatically derived. Different initial markings are generated. The reachability graph corresponding to each marking is derived and some security measures are evaluated. The tool used in this step is SURF2 (Metge *et al.*, 1994) .
4) **Presentation of the results:** the results are processed according to some selection criteria in order to be efficiently displayed on the prototype graphical user interface.

5 EXAMPLE

In order to illustrate the kind of results that can be obtained with our approach, let us consider the example already presented in Section 2. In Figure 4, the thickness of the arcs characterizes the difficulty of the attacks: the thicker the arc, the easier the attack. In this figure a unit failure rate corresponds to one success of the corresponding attack per week in average; 0.2 corresponds to one success per month, etc. To characterize very easy attacks we have attributed to them a very high transition rate value: 5000.

The users identified in the privilege graph were grouped into different sets according to the roles assigned to them in the system (see Figure 5). For instance users A and E belong to the same group “Professor”, additionally A belongs to “Proj1” and E also belongs to “Proj2”. Each role defines a set of users. We consider that each of these sets defines a specific team of possible attackers.



1) Y has no password; 2) X is in Y's ".rhost" ; 3) Y is a subset of X; 4) X can attack Y via Email; 5) Y uses a program owned by X (one time per hour); 6) Y uses a program owned by X (one time per day); 7) Y uses a program owned by X (one time per week); 8) Y uses a program owned by X (one time per month); 9) Y uses a program owned by X (one time per year); 10) X can guess Y's password in a week; 11) X can guess Y's password in a month; 12) X can modify a setuid program owned by Y.

Figure 4 Example.

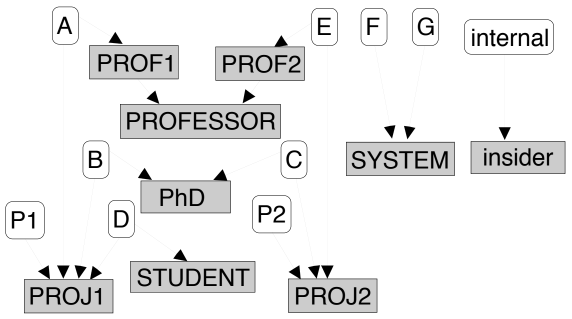


Figure 5 Grouping the users according to their roles in the system.

Let us assume that we want to protect the privileges owned by A, E and F. These three nodes are the target nodes, according to our definitions. The MTTF values calculated for each attacker team and for each of these targets are given in Figure 6 in an increasing order.

In this example, we make the hypothesis that the people having the same role in the system trust each other. Therefore, target A, for example, is not supposed to be threatened by any member of Proj1 and the corresponding MTTF is not displayed in this figure. The same is true for other roles and targets.

The lessons learned from Figure 6 are summarized hereafter.

- Considering the three targets together, the most threatening groups are *PhD* (B, C), *PROJ1* (A, B, D, P1) and *SYSTEM* (F, G). A careful analysis of the graph (Dacier, 1994) shows the following: B is a member of both groups PROJ1 and PhD; there is an easy path between B and G; G is a member of SYSTEM which has high privileges. By transitivity, the two other groups (PROJ1 and PhD) can obtain these high privileges too.

Target A: MTTF		Target E: MTTF		Target F: MTTF	
PhD:	0.980984	**PhD**:	1.000200	**PhD**:	0.191057
SYSTEM:	0.983763	**SYSTEM**:	1.020092	**PROJ1**:	0.199734
insider:	1.143936	**PROJ1**:	1.200590	**insider**:	0.305684
STUDENT:	1.176475	**insider**:	1.314162	**PROFESSOR**:	0.364364
PROJ2:	2.983863	**STUDENT**:	1.400790	**STUDENT**:	0.399934
PROF2:	2.983863	**PROF1**:	1.400790	**PROF1**:	0.399934
				PROJ2:	2.095333
				PROF2:	2.095333

Figure 6 MTTF (in # weeks) evaluated for each team of attackers.

- The low value of the MTTF from insider to F (0.305) indicates that the evaluated threat mainly results from path *insider-P1-B-Filter-G-*X_{admin}*-F* rather than from *insider-F*. However, the presence of this path can be detected since the calculated value is lower than the MTTF obtained when only the path *insider-F* is considered (equal to 1).
- The existence of multiple paths between an intruder and a target have a major impact on security. For instance, the group PROF2 which contains user E only, can easily obtain user F privileges (MTTF≈ 2 weeks) while the length of all the paths between E and F is more than 4 weeks (because all these paths contain either *P2-F* or *P2-G*).

It must be noted that the graph can also be used as a predictive tool. This means that various alternatives to enhance security can be tested by evaluating the impact of possible modifications on the security of the system and then selecting the most efficient ones. The results obtained would help in convincing the users to adopt the proposed modifications. For instance, the suppression of the facility which introduced the path between B and G, which has been identified as harmful to the security, would increase the security. Note that, in (Dacier, 1994), various modifications have been proposed and their consequences on security analyzed.

6 POTENTIAL APPLICATIONS AND CONCLUSION

We are currently working on some enhancements of the prototype in order to allow efficient application of the approach to real life systems. Various potential applications of the proposed approach are summarized hereafter.

1) Monitoring of the operational security. Once the security policy defined, each phase of the evaluation process is completely automated. Therefore, it is possible to run an evaluation once a day to monitor system security evolution. It is thus possible to identify the causes of variations, such as the installation of new software or the registration of new users, and potentially to detect a malicious user, stealing, day after day, new privileges from other users.

2) Simulation and deduction. Security constraints are sometimes conflicting with user requirements: users ask for widely open systems, ease of use and flexibility while the security administrator tries to protect sensitive information. Our prototype can help the administrator to manage this kind of problems: he can evaluate the effectiveness of different possible modifications of the privilege graph and then get some pragmatic data that should help him to convince the users of the validity and usefulness of the changes he proposes to the system.

3) Enrichment of the intrusion detection tools. We have already mentioned in Section 3.2 how our model can benefit from statistical profiles generated by intrusion detection tools. From an other point of view, we think that intrusion detection tools could benefit from our results: they could be enriched with some functionalities of the privilege graph. For instance, if a potential intruder is detected, it is possible to evaluate the risk that he represents with respect to the security constraints and to raise an alarm adapted to the corresponding risk.

Acknowledgement

The work reported in this paper has been partially supported by the ESPRIT Basic Research Action PDCS-2 (Predictably Dependable Computing Systems, Action no. 6362).

7 REFERENCES

Dacier, M. (1994) *Towards Quantitative Evaluation of Computer Security,* Doctoral Thesis, Institut National Polytechnique de Toulouse, December 20, LAAS Rep. 94488 (in French).

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

Dacier, M., Deswarte, Y. and Kaâniche, M. (1995) *Models and Tools for Quantitative Assessment of Operational Security*, LAAS-CNRS, LAAS Report, N°95353.

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

Heydon, C. A. (1992) *Processing Visual Specifications of File System Security,* Ph.D. Thesis, School of Computer Science, CMU-CS-91-201, PA, USA.

Jagannathan, R., Lunt, T., Anderson, D., Dodd, C., Gilham, F., Jalali, C., Javitz, H., Neumann, P., Tamaru, A. and Valdes, A. (1993) *System Design Document: Next-Generation Intrusion Detection Expert System (NIDES)*, SRI, Contract N0039-92-C-0015.

Lunt, T. F.,Tamaru, A., Gilham, F., Jagannathan, R., Neumann, P. G. and Jalali, C. (1990) IDES: A Progress Report, in *the Sixth Annual Comp. Security Applications,* Tucson, USA.

Metge, S., Aguéra, M., Arlat, J., Bachmann, S., Bourdeau, C., Doucet, J.-E., Kanoun, K., Laprie, J.-C., Moreira de Souza, J., Powell, D. and Spiesser, P. (1994) SURF-2: A Program for Dependability Evaluation of Complex Hardware and Software Systems, in *23rd Int. Symp. on Fault-Tolerant Computing (FTCS-23),* Toulouse, France, 668-673.

8 BIOGRAPHY

Marc Dacier is currently working at the IBM Zurich Research Laboratory in the Information Technology Solutions Department. He prepared his PhD at LAAS-CNRS and then worked at Firstel as a security consultant. His research interests focus on penetration testing of computing systems and on security policies.

Yves Deswarte is a Research Director of INRIA, and member of the Fault Tolerance and Dependable Computing research group at LAAS-CNRS. He has been president of the AFCET Technical Committee on Computer Security and Dependability and is currently the chairman of the International Steering Committee of ESORICS, the European Symposium on Research in Computer Security.

Mohamed Kaâniche is a Chargé de Recherche of CNRS, and member of the Fault Tolerance and Dependable Computing research group at LAAS-CNRS. His research interests focus on dependability modeling of computing systems, and especially on software reliability evaluation and operational security assessment.