A vulnerability life cycle based security modeling and evaluation approach

Géraldine Vache, Mohamed Kaâniche, Vincent Nicomette
CNRS; LAAS; 7, Avenue du Colonel Roche, F-31077, Toulouse, France
Université de Toulouse, UPS, INSA, INP, ISAE ; LAAS ; F-31077, Toulouse, France
Email : {gvache, nicomett, kaaniche}@laas.fr

Abstract— The objective of this work is the evaluation of information systems security using quantitative measures. These measures aim at forecasting risks and providing information to monitor the security level of the system in operation. In our approach, we take into account some environmental factors that have a significant impact on the security of the system. We have identified three such factors that are related to the vulnerability exploitation process: the vulnerability life cycle, the behavior of the attackers and the behavior of the system administrator. We have studied the interdependencies between these factors and how the evolution of these factors could impact the system security. From this study, we have defined quantitative security measures taking into account these environmental factors and we have developed a model based on Stochastic Activity Networks (SANs), describing how the vulnerability exploitation process could lead to system to be compromised. We have distinguished two scenarios according to whether the vulnerability is discovered by a malicious user or not. By analysing a vulnerability database, we have characterised the probability of occurrence of several events of the vulnerability life cycle. This characterization helped us to quantify the measures by processing the SAN model.

Keywords: *Aircraft mission reliability, stochastic assessment, dependabity modeling, maintenance, mission planning*

1. Introduction

Securing an information system is a crucial and tricky issue: since 2006, more than 7000 vulnerabilities have been published every year, according to the data recorded in the OSVDB database [1]. In this context, evaluating information system security appears to be necessary in order to analyse and prevent risks.

First approaches for the security evaluation appeared in the 80's with the development of security evaluation criteria such as the TCSEC [2], the ITSEC [3] and more recently the Common Criteria [4]. These criteria have given rise to the ISO 27000 standards [5, 6]. They define security levels, guidelines and processes to support the assessment, during the design, of the level of protection provided by an information system to cope with vulnerabilities and security related risks. The security levels defined in these criteria are considered as qualitative, in spite of the not well- defined boundary between quantitative and qualitative assessment in security. Indeed, the ISO 27000 standards define security levels, according to the functionalities implemented in the system and the level of rigour and formalisation of the development processes, that are mostly considered as qualitative measures. Moreover, these security evaluation criteria are not well suited for the evaluation of security risks considering a dynamic environment: the evaluation processes take too much time to be run regularly during the operational life of the system.

Our approach aims at producing quantitative security measures to assess the level of risk faced by an operational system considering an evolving environment. For this purpose, we first identify external factors that have an important impact on the system vulnerability exploitation process: the vulnerability life cycle and two environmental factors that are 1) the attacker population behavior and 2) the system administrator behavior. To be able to provide measures and quantify them, we study the evolution of these factors, define measures considering the impact of these factors and model them and their interactions with the system. Then, we quantify the probability of occurrence of vulnerability life cycle events and process the stochastic models we developed to evaluate the consequences of environmental factors on the system security.

This paper is structured as follows: Section 2 describes existing work related to quantitative security evaluation. Then, Section 3 details the three factors considered in our study and their impact on the system. Sections 4 and 5 are respectively dedicated to the definition of measures and to the description of the complete model of the system that enables to evaluate these measures. Section 6 addresses the estimation of the probability of occurrence of vulnerability life cycle events based on data in a vulnerability database. Section 7 details the quantitative measures and the results derived from the processing of our model.

Section 8 discusses the practical usefulness of the model from the administrator view point and lists some of the current limitations. Finally, Section 9 concludes this paper and presents our perspectives.

2. Related Work

To cope with the limitations of qualitative approaches, alternative approaches have been proposed to make quantitative security assessment feasible during the operational life of the system. In [7], the authors highlight the need of evaluation techniques for security and discuss related work of existing methodologies. In 1993, [8] argued that security can be evaluated in terms of effort, without proposing a measure and a practical model for assessing security. During the same year, [9, 10] proposed the privilege graph model. Based on the identification and analysis of known vulnerabilities of the system, the privilege graph highlights the different paths of vulnerability exploitation an attacker may use to reach a security target. The privilege graph is a state-based model where arcs model vulnerability exploitation and places model privileges owned by the attacker. A weight is assigned to each arc to quantify the effort needed to exploit the vulnerability. These weights are used to evaluate a quantitative measure corresponding to the "Mean Effort To security Failure" (METF), which is aimed at characterizing the capacity of the system to resist to attacks [11].

The attack graph formalism described in [12] is based on similar concepts: each state in the graph represents the privileges owned by the attacker as well as the attacker's knowledge and the system environment state. Several studies addressed the general and optimization of attack graphs [12, 13, 14], and their use for quantifying security [15, 16, 17]. The attack tree is another formalism used for example in [18], to assess security risks based on the evaluation of the exploitability of system vulnerabilities and the analysis of their dependencies.

Another measure called "Time To Compromise", was presented in [19] and is based on three different processes corresponding to three attack situations: 1) the attacker knows at least one vulnerability giving the wanted privileges and there is at least one known exploit; 2) there is at least one known vulnerability giving the privileges the attacker wants and the attacker does not know any successful exploit for the vulnerability; 3) the attacker is continuously looking for new vulnerabilities and new exploits. The "Time To Compromise" measure results from the modeling of these three attack processes and depends on probabilities of process occurrence and the time needed by the attacker to be successful for each process.

The quantitative approaches discussed above provide security measures for systems in operation and consider an important factor of the environment: the attacker. However, the attacker is not the only environmental factor that may impact the system security. In [20], three complementary metrics, taking into account several environmental factors, are presented: 1) a base metric that is focused on the needed access rights to exploit the vulnerability and on the impact on confidentiality, integrity and availability; 2) a time metric that is focused on exploit and patch existence; 3) an environment metric that is focused on computer system neighbourhood having the same vulnerability. It also takes into account the assessment of damage on system environment. Mathematical equations are provided to compute quantitative values for the proposed metrics. However it is not explained how the parameters involved in these equations can be estimated.

These quantitative security metrics take into account the system environment but do not consider the impact of the vulnerability life cycle. The modeling approach and the results presented in this paper aim at addressing these issues. For example, we consider that the likelihood of an attack against a system exploiting a vulnerability is not constant in time: the likelihood that an attacker exploits a new vulnerability for which a patch does not exist yet may be higher than the likelihood that an attacker tries to exploit an old patched vulnerability, under the condition that the attacker has sufficient knowledge or an easy way to do it. Thus, the vulnerability impact on the system depend on the environment evolution, as presented in the next section.

The discussion of related work of this section is not meant to be exhaustive. We have focused on related studies that are close to the topic addressed in this paper in order to position our main contributions in this area. It is noteworthy that besides the probabilistic models discussed above, other stochastic approaches have been proposed in security-related studies (e.g., to analyze security based on bayesian networks [21, 22], fuzzy logic [23], or game theory [24], etc. or to model viruses and worms propagation and their impact [25, 26]).

3. External Factors

Our purpose is twofold: 1) produce quantitative measures taking into account three relevant environmental factors that affect system security (the vulnerability life cycle, the attackers behaviors and the administrators behaviors) and 2) study how a change in the environment may have an impact on the evolution of the likelihood for a system to be secure or compromised. Of course, various system environments such as military systems and banking systems may be very different from one another. In our study, we mainly focus on mass-market information systems. In this section, we identify important external factors and we study how these factors interact with the information system and with one another

3.1. The vulnerability lifecycle

We define the vulnerability life cycle as the set of events that could occur during the life of the vulnerability. In [27], the authors take into account the events corresponding to the vulnerability discovery, the vulnerability disclosure, the patch release and the exploit availability, but also the birth and the death of the vulnerability. This approach is also followed by [28] considering that the exploit availability and the resulting attacks happen always after the vulnerability disclosure. In [29], the author does not take into account the vulnerability birth and death but adds the patch application as an event of the vulnerability life cycle. In our approach, we aim at characterizing quantitatively the vulnerability life cycle events. The patch application is not included in the vulnerability life cycle. It is taken into account through the characterization of the administrator behavior (cf. Section 3.3). Thus, we focus on the main events of the vulnerability life cycle that are considered by existing approaches, and define them as follows:

- **the discovery of the vulnerability**: once this event has occurred, the discoverer knows about the vulnerability existence and can use this knowledge for a malicious or non malicious purpose.
- **the disclosure of the vulnerability**: this is the first time the information about the vulnerability is freely available on an important source. This vulnerability has generally been studied by expert for risk evaluation
- **the release of the vulnerability patch**: once this event has occurred, it is possible to protect the system by removing the vulnerability
- **the availability of the exploit**: this event enables the attacker population to simply exploit the vulnerability. The exploit may be elaborated by the attackers or may result from the reuse of the proof of concept disclosed at the same time as the vulnerability disclosure.

Clearly, the exploit availability has a high impact on the attackers behavior. We can distinguish two different scenarios.

In the first scenario, the vulnerability is discovered by a non-malicious person, who informs the developer of the vulnerable component that the vulnerability exists and this action leads to the disclosure of the vulnerability. The disclosure of the vulnerability enables administrators to be careful but also informs the attackers population that the vulnerability exists. From that moment, an attacker may develop the exploit enabling all the attackers population to perform attacks.

In the second scenario, the vulnerability is discovered by a malicious person, who may inform other malicious persons of the existence of the vulnerability or create himself an exploit. The use of this exploit (i.e. attacks performed thanks to that exploit) leads to the vulnerability disclosure. In both scenarios, we assume that the vulnerability patch may be disclosed at the same time as the vulnerability disclosure or later. From this section, the non malicious (resp. malicious) discovery scenario will be abbreviated by the notation NM-S (resp. M-S).

3.2. Attackers behavior

The attackers population is an important environmental factor. However, it is difficult to characterize because this population is not homogeneous. In [30, 31], the authors describe a two dimensional classification considering the attackers motivations and skills. In [32], the authors studied two categories of attackers: the script kiddies and the black hats. The first ones need an exploit to be able to perform attacks. The second ones are experts, who elaborate most of the exploits. Moreover, the authors note that the black hats represent a small proportion of the attackers population. According to that, our approach is focused on the biggest part of the population that are the script kiddies.

3.3. Administrator Behavior

The third external factor that is investigated in our approach is the administrator's awareness about information system security. This is a key parameter in our approach. Indeed, whether the administrator is aware about security risks or not, may have serious consequences on the system: if the administrator does not regularly check for patch releases and does not install them as soon as they appear, the system may stay vulnerable a long time, despite of the patch release. It is noteworthy that the impact of the administrator's behavior on the security of the system depends on the vulnerability life cycle: even if the administrator is very cautious, we make the pessimistic assumption that he cannot protect his system as long as the vulnerability patch does not exist.

In the following, we will consider the two possible administrators behaviors: lax or rigorous.

4. System States and Measures

4.1. System States

In this section, we present the different states of the system, considering one vulnerability and the external factors described in Section 3. Once the vulnerability is in the system, the system becomes *vulnerable*. When an exploit for this vulnerability is available, the system becomes *exposed*. There is not much difference for the system itself between these two states: the state transition results from an environment change.

As soon as the exploit exists and is available for the attackers population, the system may be *attacked successfully* and *compromised*. If an attacker performs a successful attack, the system moves to the *compromised* state. The system stays in this state until the administrator *patches* the system, provided that the patch is available. Thus, the system becomes *patched*. However, it does not mean that, in this state, there is no risk anymore: an attacker may have obtained new privileges thanks to the exploitation of the vulnerability that could allow him to still access the system even after the vulnerability patch, unless the system is carefully checked and cleaned. For instance, an attacker that has successfully exploited the vulnerability may have installed a backdoor or a keylogger in the system. The backdoor allows him to obtain remote access to the computer (even if the vulnerability is patched), the keylogger allows him to capture all the keystrokes hit by the users of the system (which may reveal confidential data, such as passwords for instance). We consider that, in these two states (*compromised* and *patched*), the system is in danger. So, as the system is still *in danger*, it is necessary for the administrator to *clean* and *repair* the system to bring it in a *secure* state. For instance, the administrator may re-install all or part of the system software from a safe backup.

He may also compute checksums of all the binary files and compare them to previously backed-up checksums in order to detect modified binaries and re-install them. Of course, if the patch is available, the administrator may have patched the system before the exploit availability or before an attacker has enough time to perform a successful attack. In this case, the system state changes directly from *vulnerable* or *exposed* to *secure*. All these system states, as well as the impact of the external factors, are pictured in Figure 1.

Different measures could be defined based on these states and events, as detailed in the next section. As it is described in the previous paragraph, the difference between the *vulnerable* and the *exposed* states is only caused by an environment change. To make the definition of the measures more understandable, we aggregate these two states into a single state called *vulnerable* or *exposed*.

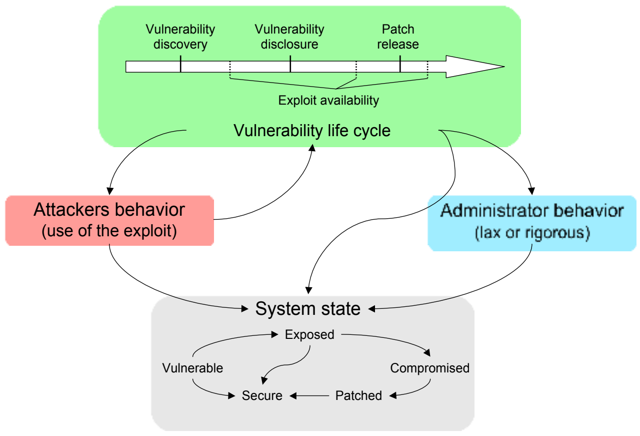

Figure 1. Summary of external factors and system states

4.2. Measures definition

In our context, we define four quantitative measures that are illustrated by Figure 2. This figure identifies, for each of the four measures, the system states that are considered (colored states): we measure the probability for the system to be in one of the colored states. The four probability measures mentioned in this figure are defined in the following.

PPC(t) quantifies the probability that the system is in the *compromised* state at instant t. It means that the system has been compromised due to a successful attack and has not been patched during the interval *[0, t]*.

PC(t) quantifies the probability for the system to be at time t in the *compromised*, *patched* or *secure* states given that the system has been compromised and repaired. Indeed, this measure quantifies the probability of occurrence of a successful attack during the interval [0, t]. It can be used to evaluate the maximum time during which the probability of having the system compromised by vulnerability exploitation does not exceed an acceptable threshold.

Besides the previous measures, the level of risk faced by the system can be assessed through the evaluation of *PCNR(t)*, which quantifies the probability, at instant t, that the system has been compromised due to a successful attack but the damage caused by attacker intrusion has not been repaired yet. This situation corresponds to the case where the system is in the *compromised* or *patched* states.

The last measure considered in Figure 2 is *PS(t)* which quantifies the probability that the system is *secure*, taking into account the impact of the considered vulnerability. This measure takes into account the two possible scenarios during the considered interval *[0, t]:*

1) a patch is applied before a successful attack occurs, and 2) an attack occurs followed by the application of a patch and repair actions.

To evaluate *PPC(t), PC(t), PCNR(t)* and *PS(t)*, it is necessary to model the factors we have presented in Section 3 and their impact on the states of the system. The next section presents the proposed model.

5. Modeling

In this section, we present a modeling approach aiming at describing the system state evolution taking into account the environmental factors. The model can be used to evaluate quantitative measures characterizing the probabilities associated with the different states of the system, presented in Section 4.

5.1. Choice of the modeling formalism

The modeling is based on Stochastic Activity Networks (SAN) [33] as this formalism can be easily used to describe the evolution of the system state and to express event occurrence conditions considering different types of stochastic distributions. SAN are composed of four modeling elements:

- *places*: they contain one or more tokens and model the system and environment states;
- *activities*: they model events that have an effect on the system or its environment; they can follow probabilistic or deterministic laws;
- *input gates*: they contain activity firing conditions; it is possible to define predicates specifying the conditions to be satisfied for the firing of the activity, according e.g., to the marking of some places;
- *output gates*: they can be used to specify the consequences of an activity firing on the marking of the SAN places.

In the next section, we describe our SAN modeling for one single vulnerability.

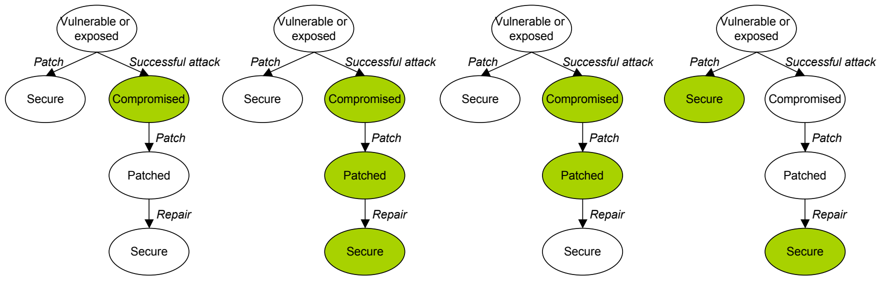

Figure 2. Measures

5.2. SAN models description

We created two models, considering the two scenarios described in Section 3 depending on the malicious or non-malicious origin of the vulnerability discovery (see Figs 3 and 4). These models are composed of two main parts: the vulnerability life cycle is modeled at the top; the remainder of the model describes the different states of the system including the administrator's and attackers behaviors. The dashed lines indicate the places used in the precondition contained in the input gates of the models.

In this section, we describe the model more in detail, beginning with the vulnerability life cycle. A preliminary version of this model is presented in [7].

5.2.1. Vulnerability life cycle modeling

Three activities (discovery, disclosure, patch) model the three events corresponding to the vulnerability discovery, the vulnerability disclosure and the patch release. States between these events are modeled by a set of four places *Ve*, V0, Vd and Vp defined as follows: (i) Ve (meaning 'existence') models the system state in which the vulnerability exists but has not yet been discovered; (ii) V0 models the system state in which the vulnerability has been discovered but has not been disclosed yet; (iii) Vd (meaning 'disclosed') models the system state in which the vulnerability has been discovered and disclosed but there is no patch available yet; (iv) Vp (meaning 'patched') models the system state in which the vulnerability has been discovered, disclosed and there is a patch available. The exploit availability is modeled by an activity with different conditions reflecting the mutual impact between exploit availability and vulnerability disclosure, with respect to the two scenarios described in Section 2. So, the activity exploit depicted in Fig. 3 models exploit availability after vulnerability disclosure (according to the NM-S) and the activity exploit in Fig. 4 models exploit availability before vulnerability disclosure (according to the M-S). The preconditions, post-conditions and the parameters characterizing this activity are different, according to the scenario. Indeed, the disclosure of the vulnerability increases the likelihood that attacker population creates an exploit as more attackers know about the vulnerability. The existence or the non-existence of the exploit are modeled by two places named, respectively, E (meaning 'exploit') and NE (meaning 'no exploit').

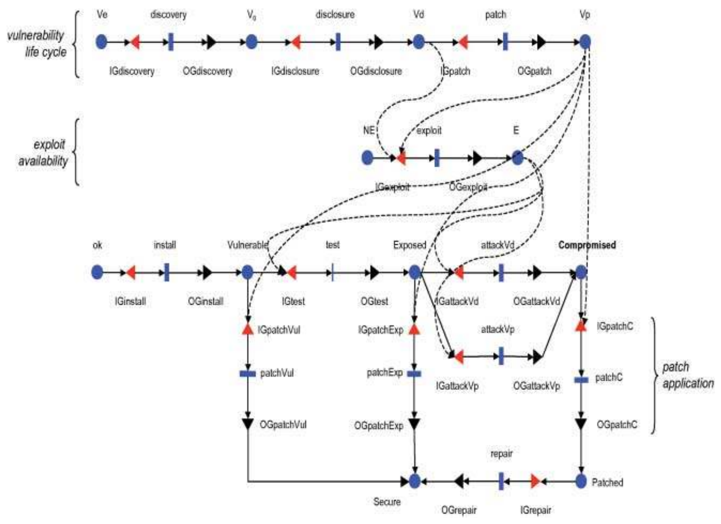

Figure 3. Vulnerability exploitation process model: NM-S

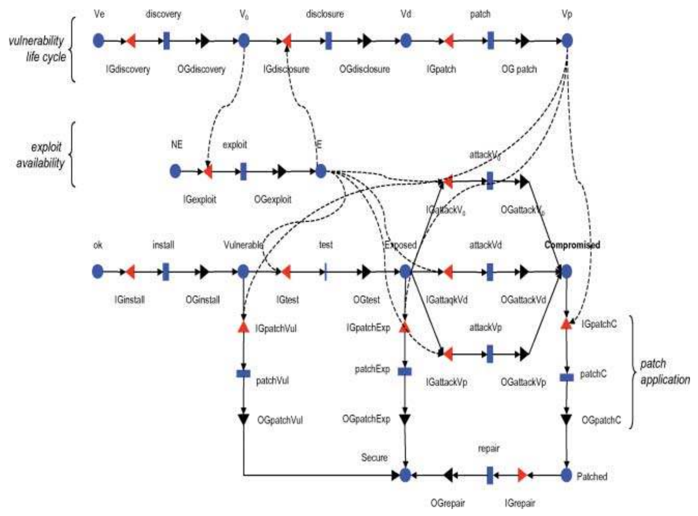

Figure 4. Vulnerability exploitation process model: M-S

5.2.2. Administrator and attackers behavior and system states modeling

The administrator's behavior is modeled through the system states themselves. Initially, the system is in state ok. The activity install models the installation of the component that can be affected by the considered vulnerability. Thus, the system moves to the state vulnerable. It becomes exposed as soon as an exploit exists (state modeled by the place E). This event is modeled by the instantaneous activity test. Conditions for the firing of this activity are defined in the input gate: the existence of the exploit and the vulnerable state of the system are necessary conditions or the system to become exposed (modeled by the place Exposed). The use of the exploit by an attacker on the system may be successful and this action is modeled by three activities attackV0, attackVd and attackVp corresponding to an attack event during the different phases of the vulnerability life cycle. It is noteworthy that the activity attackV0 does not exist if we consider the non-malicious scenario (see Fig. 4). As a result of such attack, the system becomes compromised. It will remain in this state until the vulnerability patch application by the administrator, provided that the patch is available. This action is modeled by the activity patchC: it means that the vulnerability has been patched and cannot be exploited again. However, the system is not secure yet as the damage caused by the intrusion has not been completely fixed. This transient state is modeled by the place Patched. From this state, the administrator has to clean the system, that brings it in the state Secure. It is noteworthy that the vulnerability patch application may occur as soon as the patch is available, possibly before a vulnerability exploitation. It may occur in two other different situations: (i) the system is only vulnerable and there is no exploit available yet; (ii) the system is in the state *exposed* but has not been the target of an attack. In both cases, the system becomes secure.

As described in previous sections, the approach considers the script kiddies attackers. However, only a few changes in the input gates related to the modeling of attack activities are necessary to take into account another attacker category and reflect the fact that the black hat attackers do not need an exploit to attack the system.

5.3. SAN models description

The state graphs generated from the SAN models plotted in Figs 3 and 4 and described in this section are presented in Figs 5 and 6 for the NMS and MS scenarios, respectively. These graphs summarize the possible evolutions of the system state that result from the evolution of the vulnerability life cycle, the exploit availability process and the administrator and attackers behaviors.

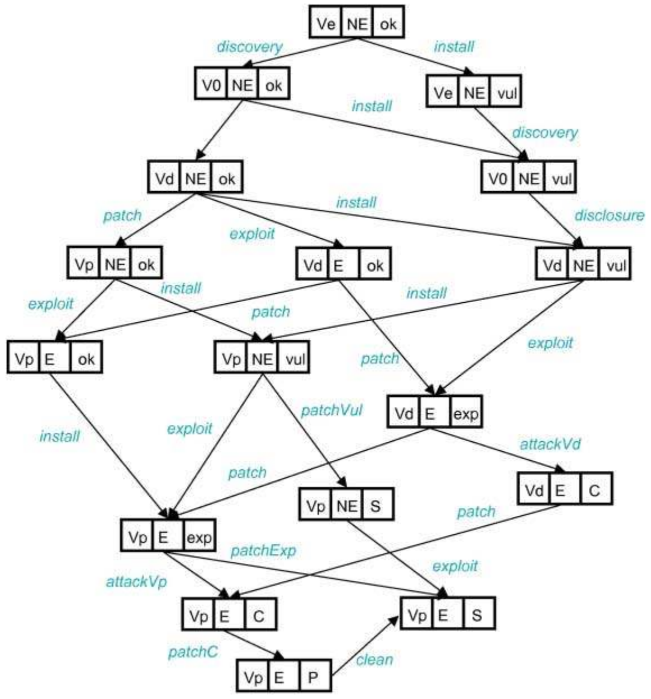

Figure 5. State graph: NM-S

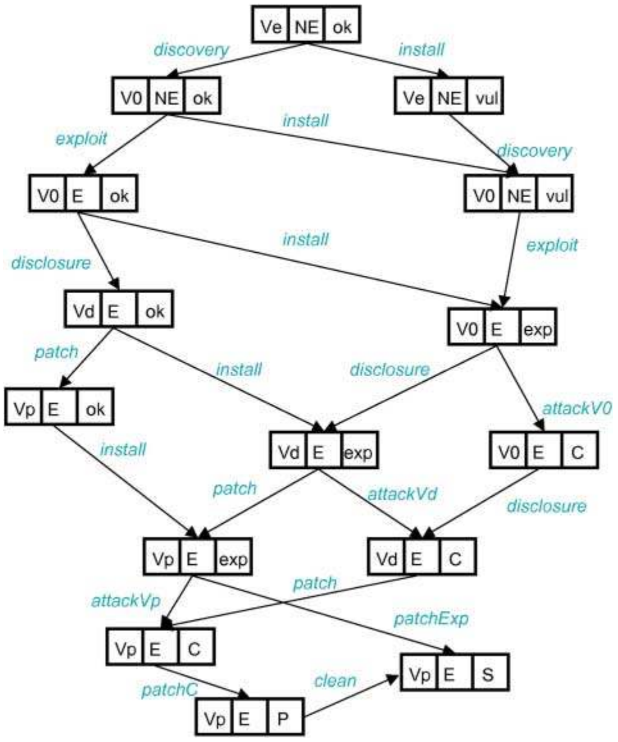

Figure 6. State graph: M-S

The states are labeled (x, y, z) where x denotes one of the vulnerability life cycle states Ve, V0, Vd, Vp; y denotes whether an exploit is available (E) or not (NE); and z denotes the system states (ok, Vulnerable, Exposed, Compromised, Patched, and Secure), represented by the labels (ok, vul, Exp, C, P, and S), respectively. It is noteworthy that only timed activities of the SAN models appear in the state graphs. The firing of instantaneous transitions leads to vanishing states that are automatically eliminated and merged with the following stable states.

The evaluation for the quantitative measures presented in Section 4 is based on the processing of the state graphs once the distributions associated to the different state transitions are specified. When all the transitions follow exponential distributions, the state graph corresponds to a Markov chain that can easily be processed using analytical techniques. Monte-Carlo simulation techniques are more appropriate when other types of distributions are considered for some state transitions. Both analytical and Monte-Carlo simulation techniques are supported by the Möbius tool implementing the SAN formalism developed by the University of Illinois at Urbana-Champaign [37].

6. Vulnerability Life Cycle Events Characterization

To quantify the measures defined in Section 4, it is important to set realistic parameters to the activities of the models. This section addresses the characterization of such parameters based on real data.

In [38], the authors study the impact of vulnerability disclosure and patch availability on the attack process. Using a data set of 308 vulnerabilities, they quantify an economical model to predict the evolution of the number of expected attacks per host and per day. In [39], the analyses are centered on the disclosure date: patch release and exploit availability events are studied, taking the disclosure date as the time origin. The study included 14326 vulnerabilities collected from several databases. This work could have been useful for us but, unfortunately, it does not consider the vulnerability 530 discovery. Thus, such data does not allow us (i) to quantify the disclosure event considering the vulnerability discovery and (ii) to characterize the exploit availability considering the vulnerability discovery.

In [40], the authors analyze 140 vulnerabilities of the OpenBSD operating system to study the vulnerability report rate. They extend the analysis described in [25] that studies the vulnerability life cycle and conclude that the rate of vulnerability discovery for an operating system can be considered as constant.

To the best of our knowledge, there are only a few studies about the characterization of the events we consider in our approach and existing work cannot be reused. Thus, the next section presents the work aiming at quantifying the probability of occurrence of the vulnerability life cycle events based on real data.

6.1. Existing vulnerability databases and statistical reports

Several organizations collect and study vulnerabilities. Some of them regularly produce reports giving information and trends about vulnerability evolution: for example, Symantec Corporation edits a survey each year, focusing on vulnerability trends and presenting analyses of quantitative data recorded by their products like their antivirus solution [41]. Other reports exist like the X Force trends and risk report [42] that classifies, e.g. the operating systems considering how many vulnerabilities were disclosed [43].

Data are also available in several databases that record each new vulnerability and characterize it by several attributes. The National Vulnerability Database (NVD) [44], managed by the National Institute of Standards and Technology of the United States and associated with the Common Vulnerabilities and Exposures (CVE), records vulnerabilities since 1999 and provides an evaluation of each vulnerability based on the CVSS metrics [23]. The Security Focus vulnerability database [45] is managed by Symantec Corporation and contains around 35 000 vulnerabilities recorded since October 1998. The OSVDB was created by the Black Hat Conference community and contains more than 52 000 vulnerabilities [1] recorded since December 1998. Secunia, a private company that provides services in security defense and vulnerability analysis, maintains also a vulnerability database since 2002 [46]. The database also indicates the severity based on the CVSS metrics. The characteristics of each vulnerability database are summarized in Table 1, indicating the vulnerability life cycle events for which the corresponding date is available.

TABLE 1. Vulnerability databases comparison.

Database	NVD	Security Focus	OSVD	Secunia
Discovery date	No	No	Yes	No
Disvlosure date	Yes	Yes	Yes	Yes
Patcg date	No	No	Yes	No
Exploit date	No	No	Yes	No

6.2. Events characterization

To estimate the parameters characterizing the occurrence of the vulnerability life cycle events described in our models, it is necessary to obtain a sufficient and as complete as possible vulnerability data set. The most complete data set is, of course, the union of the data provided by every vulnerability database. Unfortunately, the vulnerability databases are very heterogeneous. Even if the CVE reference seems to be a useful and unique vulnerability reference, it is not indicated in each database. Thus it is not easy to merge and to correlate the information reported in different vulnerability databases. Thus, before analyzing data, we have to choose the most relevant vulnerability database for our study. As our goal is to characterize the vulnerability life cycle events, our interest is focused primarily on timed parameters. The OSVDB database matches our requirements as it is the only one that records that kind of information for all the events. This database provides a large set of data. We analyzed 52 000 vulnerabilities extracted from the database and recorded since December 1998. For each vulnerability, the OSVDB identifier, the vulnerability categories, and the time corresponding to the discovery date, the disclosure date, the patch release date and the exploit date are recorded if they are available. Unfortunately, this is not the case for each vulnerability. The next section presents the first step of our analysis. This work is an extension of the work described in [8].

6.3. Preliminary analysis of the data set

Before analyzing the data set to fit the intervals between two of these events with probabilistic distribution, this subsection provides a preliminary analysis of the data set, based on the information summarized in Table 2.

The number in cell(i,j) indicates the number of events for which the occurrence date of the corresponding events i and j is available. The percentage in cell(i,j) indicates the proportion among the set of vulnerabilities with information available about the occurrence of event i for which information is also available about the occurrence of event j. Let us take as a simple example the set of vulnerabilities for which both discovery and disclosure dates are available. This set of 3926 vulnerabilities represents only a small proportion of the total number of vulnerabilities: 7.8%. But it also represents 99.12% of the set of vulnerabilities for which the discovery date is available. This small number of vulnerabilities may be explained by the fact that the vulnerability discovery is not an event that is officially published.

Considering the set of vulnerabilities for which both vulnerability disclosure and patch release dates are available, it counts 871 vulnerabilities and represents only 1.71% of the set of vulnerabilities for which the vulnerability disclosure date is available. This set represents 75.67% of vulnerabilities for which the patch release date is available.

Two reasons could explain the small number of vulnerabilities for which the patch release date is available:

(i) Only a small proportion of the vulnerabilities that are 630 disclosed have an available patch. This explanation seems possible because we consider that the vulnerability may be disclosed by another source than the producer of the vulnerable component, as in [47].

(ii) The information reported in the database is incomplete and the fact that the patch release date is not recorded does not mean that it does not exist.

It is impossible for us to validate or invalidate one of these two explanations. However, studies by Jumratjaroenvanit and Teng-amnuay [47] encourage us to consider the fact that vulnerabilities could be disclosed and not be patched.

Finally, let us examine the set of vulnerabilities for which both exploit availability and disclosure dates are available: it represents only 34.34% of the studied vulnerability set. This highlights the fact that the exploit availability, as the patch release, is not a systematic event in the vulnerability life cycle. It is important to take this new information into account for the parameterization of our models. In the next section, we present the probability distributions characterizing the occurrence of the vulnerability life cycle events that we estimated based on the data set presented in Table 2.

6.4. Data analysis and event characterization

To characterize the probability of occurrence of an event from the vulnerability state *i* to the state *j*, we select the vulnerabilities for which the dates *ti* and *tj* are available and evaluate the duration *tj–ti*. Thus, a new data set composed of the evaluated durations between state *i* and state *j* is obtained. We need to classify these data to be able to analyze them and find the more appropriate probability distribution. Organizing the data in bins to estimate the corresponding empirical distribution enables to focus on the general trend and then to minimize the impact of very little variations. However, it is also important not to choose a too small number of bins, which could mask important information. We determine the number of bins thanks to the Sturges formula [48]. We make the choice that all the bins contain the same number of data in each bin [8].

Once the data are processed, we use the EasyFit tool [49] to fit the empirical distribution obtained with several probability distributions. The Kolmogorov–Smirnov statistical test is used to assess the quality of fit of the considered probability distributions.

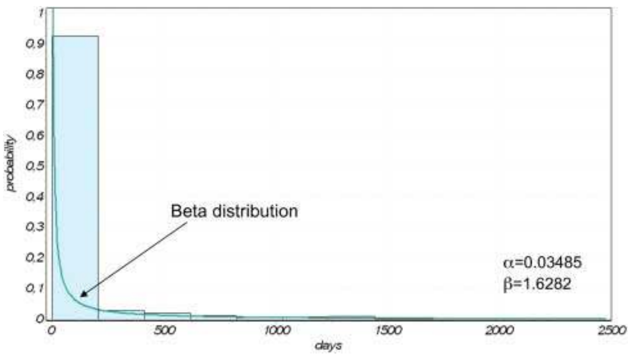

FIGURE 7. Time intervals between discovery and disclosure dates.

We processed our data to characterize vulnerability disclosure, patch release and exploit availability events. As the data set used in our study provides the discovery date of the vulnerability but does not provide the date of release of the vulnerable component, it is not possible for us to characterize the vulnerability discovery event.

6.4.1. Vulnerability disclosure event characterization

The vulnerability disclosure event may occur after the non- malicious discovery of the vulnerability or after the use of the exploit, according to the two scenarios detailed earlier in this paper.

First, we study the disclosure event in the context of the non-malicious discovery scenario (NM-S). There are 3926 vulnerabilities in the OSVDB database with discovery and disclosure dates. Seven hundred and eight of them have been discovered and disclosed at the same time, and 3218 vulnerabilities have been disclosed 1 day or more after the discovery. The histogram depicted in Fig. 7 represents the empirical distribution of the time between the discovery and the disclosure of the vulnerability. The first bin has a value equal to 92.51% and the second one to 2.75%. This sharp decrease of the probability can be described by a Beta distribution. This was confirmed by the Kolmogorov–Smirnov test applied to the data.

The parameters and the P -values of the Kolmogorov–Smirnov test are summarized in the recapitulative Table 3: *t0* represents the discovery date; td the disclosure date; *tp* the patch release date and *te* the exploit date. The parameters α and β are the shape parameters of the Beta distribution, whose density function is

$$f(x) = \frac{x^{\alpha-1}(1-x)^{\beta-1}}{B(\alpha,\beta)}.$$

Let us consider the second scenario in which the vulnerability is discovered by a malicious person (M-S). We need to take into account the disclosure dates td and the exploit dates *te* such as $td - te$ has a positive value. There are 222 vulnerabilities for which the exploit date predates the disclosure date. The time interval between these two events varies from 1 to 2151 days. The probability distribution fitting with the data set is a Beta distribution once again. The empirical distribution and the associated Beta probability distribution are depicted in Fig. 8.

6.4.2. Patch release event characterization

The patch release event is studied considering the same principle that we presented for the vulnerability disclosure event. We studied 871 vulnerabilities. For 712 of them, the date of the patch release is the same as the date of vulnerability disclosure. The time intervals are between 0 and 759 days. It appears that the Beta distribution fits with our data and satisfies the Kolmogorov–Smirnov test. The estimated parameters of the Beta distribution are given in Table 3.

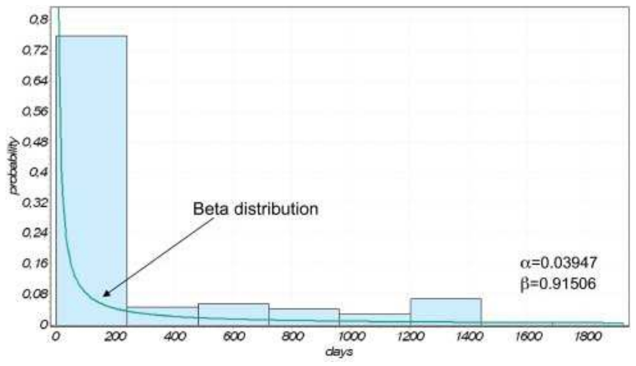

FIGURE 8. Time intervals between exploit availability and disclosure dates.

TABLE 3. Summary of parameters of the Beta probability distribution.

Time Interval	Event	α	β	p-value
td−t0	Vulnerability disclosure (NM-S)	0.03485	1.6282	0.38
tp−td	Patch release	0.00352	0.62362	0.41
te−td	Exploit availability (NM-S)	0.00090	1.8666	0.35
te−t0	Exploit availability (M-S)	0.02916	1.5813	0.38
td−te	Vulnerability disclosure (M-S)	0.03947	0.91506	0.34

6.4.3. Exploit availability event characterization

To characterize the occurrence of the exploit availability event, we need to compare the dates of exploit availability to the dates of vulnerability discovery (in the M-S) and vulnerability disclosure (in the NM-S).

First, we consider the data set of 2131 vulnerabilities for which exploit availability and discovery dates are available: 389 of them have the same discovery and exploit availability dates.

For these ones, we can make the assumption that the discovery is malicious. This set of 2131 vulnerabilities fits with a Beta distribution and this fitting satisfies the Kolmogorov–Smirnov test. Then, we focus on the comparison between vulnerability disclosure and exploit availability dates. It is based on 17 857 vulnerabilities. In this data set, we notice that: (i) for 222 vulnerabilities, the exploit appears before the vulnerability disclosure (these vulnerabilities are used for the characterization of vulnerability disclosure event in the M-S); (ii) for 17 077 vulnerabilities, the exploit and the vulnerability are disclosed the same day; (iii) for 558 vulnerabilities, the exploit appears after the vulnerability disclosure. In this section, we analyze the vulnerabilities of the two last cases that are very likely to correspond to the NM-S. It is important to notice the large amount of vulnerabilities that are disclosed and exploited in the same day. This may highlight the important impact of the disclosure event. Considering the two last vulnerability sets of, respectively, 17077 and 558 vulnerabilities, the analysis of the distribution of the time interval between vulnerability discovery and exploit availability have shown that the Beta distribution provides a good fit confirmed by the Kolmogorov–Smirnov test.

6.4.4. Discussion

The previous results show that the Beta distribution provides a good fit to characterize the occurrence of the vulnerability life cycle events. These results are based on the data stored in the vulnerability database at the date of 25 December 2008.

It is important to monitor the validity of this distribution and re-estimate the parameters based on recently collected data. This is important as the validity of the results derived from the models relies on the representativeness of the assumptions and of the values assigned to the parameters associated to the events described in the models. As an example, Table 4 reports the average values of the time intervals associated to life cycle events considering three periods: (i) before 2001, (ii) between 2001 and 2006 and (iii) after 2006. The number of vulnerabilities for each period is also indicated. In this case, comparing the average values computed for the global data set with those obtained for each period, we obtain generally the same order of magnitude. The more significant differences are observed for the cases where the number of vulnerabilities is low. In an operational real-life context, a more thorough analysis of the possible time evolution of the estimated parameters need to be done at a regular basis.

7. Model Pprocessing and Quantitative Evaluation of the measures

The next step of our approach consists in running simulations of the defined models to obtain quantitative values of the measures presented in Section 4.2. We use the Möbius tool that integrates a set of solvers allowing the processing of SAN models using analytical and Monte-Carlo simulation [37]. Different types of distributions are supported by the tool. In this section, we first present how we parameterize the models activities. Section 4.2 discusses the parameterization of the vulnerability life cycle activities based on the results described in Section 6. Section 7.1.2 focuses on the activities related to the attackers behavior and Section 7.1.3 is dedicated to the administrator behavior activity parameterization. The results of the model processing are presented in Section 7.2.

In this section, we assume that the vulnerability is already in the system. Thus, we do not consider the installation process modeled by the activity install and we assume that the system is initially in the vulnerable state.

TABLE 4. Life cycle events parameters: time evolution.

Event		***td-t0***	***tp-td***	***te-td ≥0***	***te-td < 0***	***te-t0***
Global	*Mean(h)*	1474	142	33	-3098	1300
	Nb Vul	3925	871	17857	222	2108
Before 2001	*Mean(h)*	1135	0	73	-150	1561
	Nb Vul	113	1	747	16	50
2001-2005	*Mean(h)*	1280	1284	59	-3431	1202
	Nb Vul	2676	17	6916	138	1536
After 2005	*Mean(h)*	1958	120	13	-827	1561
	Nb Vul	1137	853	10194	68	522

7.1.Parameters description

7.1.1. Vulnerability life cycle

To parameterize the activities modeling the life cycle of the vulnerability, we use Beta distribution with the results of the characterization described in Section 6.

In the NM-S, the preliminary analysis has shown that the exploit availability and the patch release may not occur. According to the database, only 2% of the vulnerabilities that are disclosed have a patch disclosed as well. This value of 2% for the patch existence seems to be very small. Thus, we have decided to perform sensitivity analyses on the model considering not only this value provided from the database analysis, but also other possible values for the probability of patch existence (5, 10, 50 and 100%). When the patch exists, its disclosure is modeled with a Beta distribution.

The analysis of the vulnerability database shows that only 34.5% of the vulnerabilities have an associated available exploit. In the malicious discovery scenario, we make the assumption that the patch release occurs inevitably. This choice is justified by the fact that a vulnerability discovered by malicious people, and so exploited before the vulnerability disclosure, represents a very serious threat. This is the reason why we assume that the patch will be disclosed with a probability equal to 1 in this case.

7.1.2. Vulnerability exploitation: attack process

The attack process is different according to the vulnerability discovery scenario that is considered. The vulnerability may be exploited before the vulnerability disclosure only in the M-S and is modeled by the activity attackV0. The other two attack activities, attackVd and attackVp, are present in the two models. These three activities are described by probabilistic exponential distributions, based on the work presented in [6, 50]. As summarized in Table 5, we assume different rates, defined empirically, according to the considered phase of the 820 vulnerability life cycle and higher attack rates, for the M-S as the vulnerability represents, in this case, a higher threat for the system. It is assumed that the attack rate when the vulnerability has been disclosed is higher than at any other moment of the life cycle, as all the attackers may exploit it successfully. A sensitivity analysis considering different attack rates values is presented in Section 7.2.1.

7.1.3. Patch application and system repair

The patch may be applied by the administrator considering three different circumstances: the system is *vulnerable*, *exposed* or *compromised* (modeled, respectively, by the activities patchVul, patchExp, patchC). As there is no previous work in literature and no data available to provide such information, we assume that these activities can be described by normal distributions (cf. Table 5), as this distribution seems intuitively to well describe the time of reaction needed by the administrator[1]. The higher the threat faced by the system (considering the states *vulnerable, exposed and compromised*), the shorter the meantime between the patch release and the patch application. The patch application prevents other attacks from being successful on the system but it is not sufficient to make the system secure. The repair of the system corresponds to the cleaning and recovery task that is necessary to secure the system. In the model processing, we assume a one day duration to repair the system. In our study, we analyze two different administrator behaviors: lax and rigorous[2]. When there is no exploit and no disclosure of the vulnerability, i.e. in a context with no real threat, the lax administrator updates the system once a month when the rigorous one updates it every day.

Table 5. Parameters for the modeling of attackers and administrator related activities (NM-S and M-S scenarios)

Activity	**Distrib.**	**Parameter**	**Value**
Attack Vd (NM-S)	Exp.	Rate	0.5/day
Attack Vp (NM-S)	Exp.	Rate	0.1/day
Attack V0 (NM-S)	Exp.	Rate	1/day
Attack Vd (M-S)	Exp.	Rate	5/day
Attack Vp (M-S)	Exp.	Rate	1/day
patchVul	Normal	Mean, Variance	1/30 days 0.5 days^{-2}
patchExp	Normal	Mean, Variance	0.5/15 days 0.5 days^{-2}
patchC	Normal	Mean, Variance	0.1/3 days 0.5 days^{-2}
repair			3 days 0.5 days^{-2}

[1] The choice of the normal distribution is done empirically and any other type of distribution could be considered in our model and processed by the Mobius tool.

[2] We study two extreme administrator behaviors in order to highlight the impact of this external factor. An administrator may, of course, has an intermediate behavior

7.2. Results

7.2.1. Non-malicious discovery scenario (NM-S)

This section presents the results obtained from the processing of NM-S model. We focus on the states of the system which are necessary to quantify the measures. The first part of this section is dedicated to the study of the probability for the system to be in the vulnerable, exposed and patched states before evaluating the measures presented in Section 4.2. The two different administrator behaviors (rigorous and lax) are modeled by the three activities patchVul, patchExp and patchC.

Figure 9 depicts the evolution of the probability for the system to be in the *vulnerable* state. It highlights the influence of the patch existence (through probability *p*) but also the difference between the two administrator behaviors. When the administrator is rigorous, the probability for the system to be in the *vulnerable* state decreases quickly because of the exploit availability (which makes the system move to the *exposed* state) but also because the administrator applies the patch as soon as it is disclosed. Both events occur around 1 day in average after the vulnerability disclosure. In the case of a lax administrator, the probability starts by decreasing because of the exploit availability (around day 1) that leads the system in the *exposed* state. Anyway, it takes a long time for the administrator to apply the patch, as illustrated by the slow decreasing curve (starting around day 30).

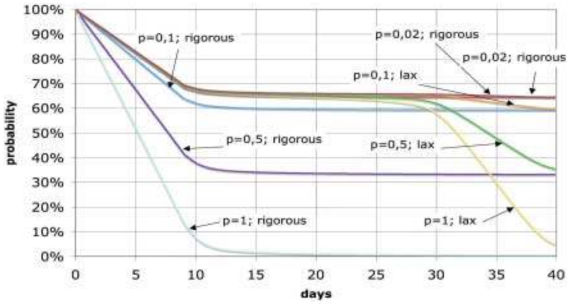

FIGURE 9. Evolution of the probability for the system to be in vulnerable state in NM-S.

The curves in Fig. 10 depict the evolution of the probability of having the system *exposed*. First, the curve exhibits an increasing trend because of the exploit availability event. This increase is, however, less important for the rigorous administrator who has already applied the patch before the occurrence of a successful attack. The decreasing phase may be caused by two events: the patch application or a successful attack. In the case considering a low probability of patch existence (2 and 10%), the event that has the highest impact is the attack process which has a higher occurrence rate than the patch application rate. On the contrary, for the other cases (50 and 100%), the decrease of the curve is mainly due to the patch application and not the vulnerability exploitation.

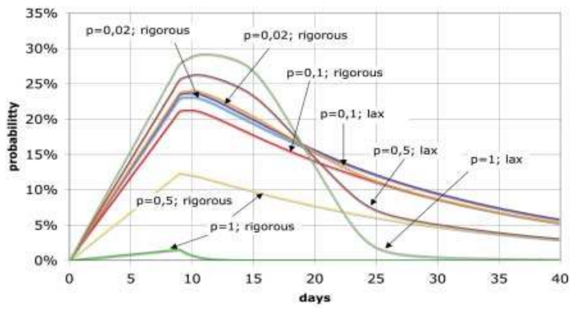

FIGURE 10. Evolution of the probability to be in exposed state in NM-S

The evolution of the measure *PCNR(t)* (cf. Fig. 11) is very similar to the *PPC(t)* evolution. Considering a low probability of patch existence (2 or 10%), the curves corresponding to the lax administrator and the rigorous one are indistinguishable and increase quickly to the value 34.5%, that is the maximum probability of exploit availability (cf. Section 6). When we consider a 100% probability of patch existence, it is noticeable that there is more difference between the values obtained, due to the difference between the two administrator behaviors: the lax administrator has a small *PCNR(t)* value but significantly higher than the rigorous one. Moreover, the difference between the curves is due to the fact that the probability to reach the patched state, given that the system has been compromised, is lower in the rigorous case compared with the lax case is directly related to the fact that the system in this case has a higher probability to be patched before a successful attack occurs.

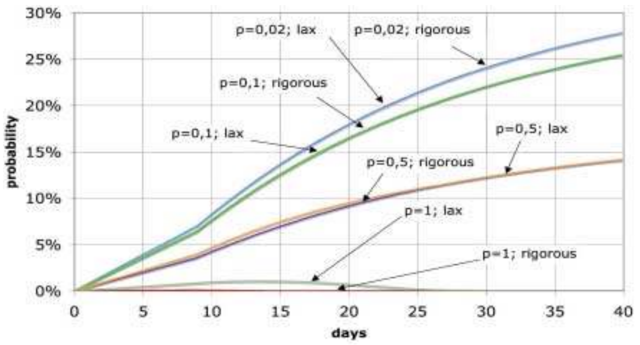

FIGURE 11. Evolution of *PCNR(t)* in NM-S.

The probability for the system to be in a patched state has a direct impact on the probability for the system to reach the secure state *PS(t)*, plotted in Fig. 12. As expected the probability of patch existence has a high influence on *PS(t)* which exceeds (for a rigorous administrator) 90%, at day 10 in the case of a 100% patch existence probability, compared with 9% when the probability of patch existence is 10%. Figure 12 also highlights the difference between the two administrator behaviors: with the same patch existence probability, it takes at least 25 days for the lax administrator to reach the same *PS(t)* value than the rigorous administrator, and certainly after the system has been compromised and repaired.

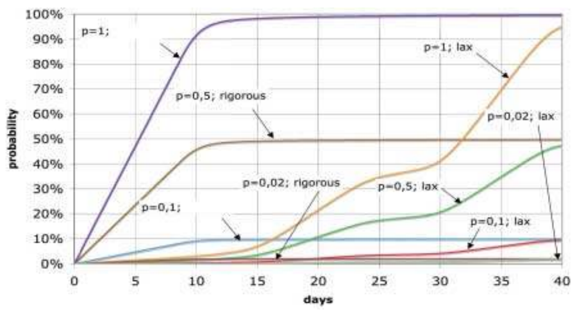

FIGURE 12. Evolution of *PS(t)* in NM-S.

An additional insight is obtained by varying the attack rates: *Rd* and *Rp*, that are associated to the rates of activities `attackVd` and `attackVp`. Figure 13 shows the evolution of the probability *PS(t)* assuming that the patch always exists. The probability evolution considering different attack rates are very similar when a rigorous administrator is considered, as the patch is applied as soon as it is disclosed. The case in which we consider a lax administrator shows a higher sensitivity to the attack rate.

If we consider a high attack rate, the system becomes secure sooner but this quick patch application is due to the fact that the system has been compromised and repaired before becoming secure.

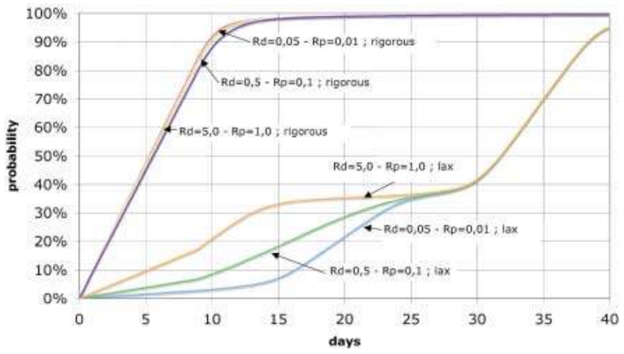

FIGURE 13. Evolution of *PS(t)*in NM-S considering different attack rates.

7.2.2. Malicious discovery scenario (M-S)

As has been done for the NM-S, we present the probability for the system to be in one of the states and deduce the values for the measures defined in Section 4. For each state or measure considered, we compare the trends for the two administrator behaviors and the two scenarios. First, Fig. 14 depicts the probability for the system to be in the vulnerable state considering the case when the patch always exists (p = 1). The two curves corresponding to the two administrator behaviors in M-S scenario are very similar and decrease more quickly than in the case of the NM-S.

This is due to the fact that the exploit is available before the patch. Thus, even if the administrator is a rigorous one, the patch cannot be applied and the system moves to the exposed state. Figure 15 depicts the evolution of the measure *PCNR(t)* in the M-S, compared with the NM-S.

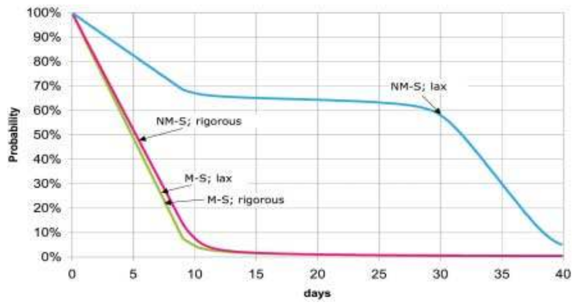

FIGURE 14. Evolution of the probability to be in *vulnerable* state in M-S compared with NM-S.

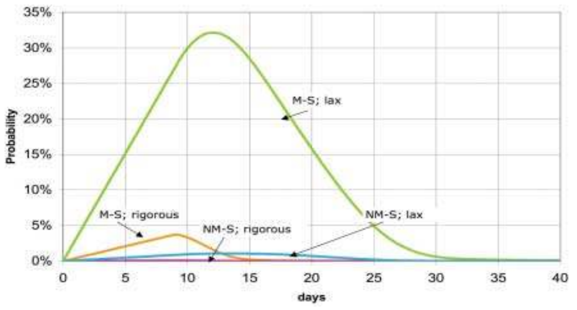

FIGURE 15. Evolution of *PCNR(t)* in M-S compared with NM-S

The difference between the two administrator behaviors is important: *PCNR(t)* has a maximum value of 32% with a lax administrator and 4% with a rigorous one. Figure 16 shows the evolution of the measure *PC(t)*. This measure enables to know if the system is or has been in danger considering the vulnerability. Considering the same administrator behavior, *PC(t)* reaches lower values in the case of the NM-S as the administrator can apply the patch before the exploit is available. The difference between the two administrator behaviors is also important as the probability of the system to have been compromised reaches 60.6% at day 10 in the case of a lax administrator and only 10.2% in the case of a rigorous administrator, considering the M-S. Finally, Fig. 17 depicts the evolution of *PS(t)*, which measures the probability for the system to be in a secure state, even if it has been compromised. The secure state is an absorbing state in the model, so, the curve on the graph increases and reaches asymptotically the value 1. It is interesting to note that the two curves considering rigorous administrators follow the same trend. In the case of the lax administrator, the patch is applied sooner in the M-S scenario because of the higher probability of successful attack.

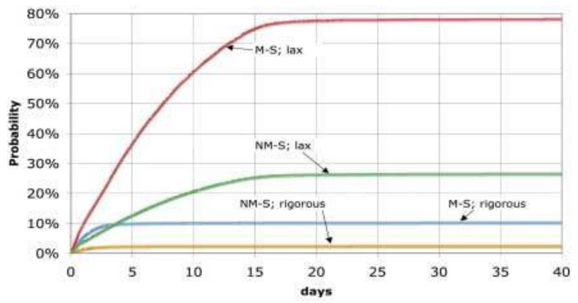

FIGURE 16. Evolution of *PC(t)* in M-S compared with NM-S

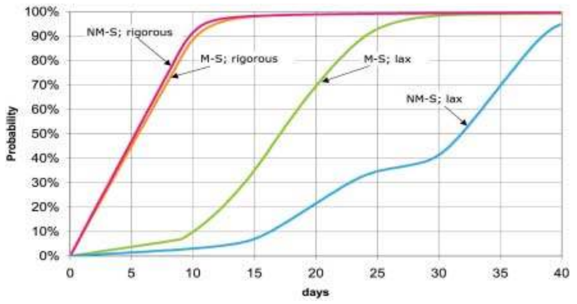

FIGURE 17. Evolution of *PS(t)* in M-S compared with NM-S

8. Discussion on Practical Applicability and Limits of the Model

The modeling approach presented in this paper is a first step towards an 'automated process' for quantitatively evaluating the security of information systems. As for now, the proposed models are useful and adapted to understand, represent and analyze the influence of some factors on the security level of information systems. The proposed set of four security measures enables us to have a global vision of the security risks for the system resulting from the combined effect of three different factors: the vulnerability life cycle events, the attacker behavior and the administrator behavior. They enable a system administrator to answer the following questions: (i) 'what is the probability for my system to be compromised by a successful attack?', (ii) 'what is the probability for my system to be secure considering a specific vulnerability?', (iii) 'how the time to patch a given vulnerability affects the probability of the system to be secure', etc. These measures should be assessed and updated at a regular basis using recent information reported in public vulnerability databases. By providing this global view of the security risks, these measures are aimed at enabling the security administrators to have a better awareness of the levels of risks induced by the considered factors and providing them useful hints to mitigate these risks by taking some strategic decisions such as: patching some particular software as soon as possible, de-installing some vulnerable component, etc.

The proposed models identify a set of parameters that need to be estimated based either on data already available from vulnerability databases or based on security experts judgments (when some of such data are missing). The problem of data availability is common to all model-based studies and usually it is sufficient to indicate the order of magnitude of some parameters to perform sensitivity analyses and derive some general trends about the computed measures. As indicated in the paper, the information recorded in the vulnerability databases are heterogeneous and sometimes incomplete and some research studies are needed to improve the content of such databases or to correlate the information contained in different databases in order to obtain a more complete dataset compared with information derived from a single database. On the other hand, we believe that our work also contributes to identify relevant information that it would be useful and interesting to record in the future in the vulnerability databases.

Finally, it is important to note that the models developed at this stage analyze the system at a macroscopic abstraction level adopting a black box approach. Further extensions are needed in order to perform analyses at a finer granularity taking into account, in addition to the considered factors, the system architecture and the interactions between the system components. The development of such a model will require the analysis of the vulnerabilities at the component level rather than globally leading to a more detailed description of the system states. Also, besides patch application, more details could be included to describe system administrators' behavior and the activities that can be used to manage and resolve software and system-related vulnerabilities.

Clearly, the complexity of the models increases with the level of detail included. Such a complexity can be mastered thanks to the significant progress achieved in the last decade in the area of SANs and more generally in the context of other state-based modeling formalisms.

9. Conclusion and Perspectives

This paper presented a modeling approach for quantitative security evaluation. Our objective is to elaborate an evaluation process that can provide measures quantifying the risks for the system to be compromised by a successful attack exploiting a vulnerability. Our study focuses on the characterization and the modeling of the vulnerability exploitation process and its impact on the state of the system. The first step of this approach is the identification and the characterization of the external factors that could have an impact on the vulnerability exploitation process. This study highlights three factors that have an impact on security: the vulnerability life cycle and two environmental factors, the attacker behavior and the administrator behavior with respect to the application of vulnerability patches. The dependencies between these factors led us to distinguish two scenarios based on whether the vulnerability is discovered by a malicious or a non-malicious user. Taking into account these factors, we identified the different states of the system and defined four

References

[1] Open Security Foundation, Open source vulnerability database. http://osvdb.org.

[2] U.S. Department of Defense. (1985) Trusted computer security evaluation criteria.

[3] European Communities.(1991) Information technology security evaluation criteria

[4] ISO/CEI 15408. (1996) Common criteria for information technology security evaluation.

[5] Nicol, D., Sanders, W. and Trivedi, K. (2004) Model- based evaluation: from dependability to security. IEEE Trans. Dependable Secure Comput., 1, 48–65.

[6] Ortalo, R., Deswarte, Y. and Kaâniche, M. (1999) Experimenting with quantitative evaluation tools for monitoring operational security. IEEE Trans. Softw. Eng., 25, 633–650.

[7] Vache, G. (2009) Environment Characterization and System Modeling Approach for the Quantitative Evaluation of Security. Proc. 28th Int. Conf. on Computer Safety, Reliability and Security, pp. 89–102.

[8] Vache, G. (2009) Vulnerability Analysis for a Quantitative Security Evaluation. Proc. Int. Symp. on Empirical Software Engineering and Measurement, pp. 526–534. IEEE Computer Society.

[9] Jonsson, E. and Olovsson, T. (1997) A quantitative model of the security intrusion process based on attacker behavior. IEEE Trans. Softw. Eng., 23, 235–245.

[10] Dacier, M. (1994) Vers une évaluation quantitative de la sécurité informatique. PhD Thesis, Institut National Polytechnique, Toulouse.

[11] Dacier, M., Deswarte, Y. and Kaâniche, M. (1996) Models and Tools for Quantitative Assessment of Operational Security. Information Systems Security: Facing the Information Society of the 21st Century, pp. 177–186.

[12] Sheyner, O. (2004) Scenario graphs and attack graphs. PhD Thesis, Carnegie Mallon University, Pittsburgh, PA.

[13] Jha, S., Sheyner, O. and Wing, J. (2002) Two Formal Analyses of Attack Graphs. Proc. 15th IEEE Computer Security Foundations Workshop, pp. 49–63.

[14] Swiler, L., Phillips, C., Ellis, D. and Chakerian, S. (2001) Computer-Attack Graph Generation Tool. Proc. DARPA Information Survivability Conf. & Exposition II, 2001. DISCEX'01, pp. 307–321.

[15] Wang, L., Islam, T., Long, T., Singhal, A. and Jajodia, S. (2008) An Attack Graph-Based Probabilistic Security Metric. Proc. 22nd annual IFIP WG 11.3 Working Conf. on & Data and Applications Security, pp. 283–296. Springer, Berlin Heidelberg.

[16] Idika, N. and Bhargava, B. (2010) Extending Attack Graph-Based Security Metrics and Aggregating Their Applica- tion. IEEE Transactions on Dependable and Secure Com- puting, Vol. 99. ISSN: 1545-5971 (Pre-prints). http://doi.ieeecomputersociety.org/10.1109/TDSC.2010.61.

[17] Noel, S., Jajodia, S. and Singhal, A., (2010) Measuring Security Risk of Networks Using Attack Graphs. Int. J. Next-Gener. Comput., 1, 135–147.

[18] Mauw, S. and Oostdijk, M. (2005) Foundations of Attack Trees. Information Security and Cryptology-ICISC 2005, Lecture Notes in Computer Science, Vol. 3935, pp. 186–198.

[19] Jurgenson, A.and Willemson,J.,(2010) On Fasta nd Approximate Attack Tree Computations. Proc. 6th Int. Conf. on Information Security Practice and Experience, ISPEC

2010, Seoul, Korea, Lecture Notes in Computer Science, Vol. 6047, pp. 56–66.

[20] Schneier, B. (1999) Modeling security threats. Dr Dobb's Journal, December 1999.

[21] Kordy, B., Mauw, S., Radomorovic, S. and Schweitzer, P. (2011) Foundations of Attack-Defense Trees. Proc. Formal Aspects of Security and Trust (FAST 2010), Lecture Notes in Computer Science, Vol. 6561, pp. 80–95.

[22] Balzarotti, D., Monga, M. and Sicari, S. (2006) Assessing the Risk of Using Vulnerable Components. Quality of Protection, pp. 65–77. Springer, USA.

[23] Mell, P., Scarfone, K. and Romanosky, S. A complete guide to the Common Vulnerability Scoring System Version 2.0. http://www.first.org/cvss/cvss-guide.html.

[24] Arbaugh, W., Fithen, W. and McHugh, J. (2000) Windows of vulnerability: a case study analysis. Computer, 33, 52–59.

[25] Rescorla, E. (2005) Is finding security holes a good idea? IEEE Secur. Priv., 3, 14–19.

[26] Frei, S. (2009) Security Econometrics—The Dynamics of (In)Security. Eth zurich, PhD dissertation, ETH Zurich.

[27] Frigault, M. and Wang, L., (2008) Measuring Network Security Using Bayesian Network-Based Attack Graphs. Proc. 32nd Annual IEEE Int. Computer Software and Applications (COMPSAC'08), pp. 698–703.

[28] Xie, P., Li, J.H., Ou, X., Liu, P. and Levy, R., (2010) Using Bayesian Networks for Cyber Security Analysis. Proc 2010 IEEE/IFIP Int. Conf. on Dependable Systems and Networks (DSN-2010), Chicago, IL, USA, pp. 211–220.

[29] Lu, G.-M., Chen, Z.-H.,He, X.-Z.and Li,J.-P.,(2008) A Method of Security Evaluation based on Fuzzy Mathematics. Proc. Int. Conf. on Apperceiving Computing and Intelligence Analysis (ICACIA 2008), pp. 106–109.

[30] Zonouz, S.A., Khurana, H., Sanders, W.H. and Yardley, T.M. (2009) RRE: A Game-Theoretic Intrusion Response and Recovery Engine. Proc. 2009 IEEE/IFIP Int. Conf. on Dependable Systems and Networks (DSN-2009), Lisbon, Portugal, pp. 439–448.

[31] Gelenbe, E. (2007) Dealing with software viruses: a biological paradigm. Inf. Sec. Tech. Rep., 12, 242–250.

[32] Zou, C.C. and Towsley, D. (2007) Modeling and simulation study of the propagation and defense of Internet E-mail Worms. IEEE Trans. Dependable Secure Comput., 4, 105–118.

[33] CERIAS. (2005) The Development of Meaningful Hacker Taxonomy: A Two Dimensional Approach. Technical Report 2005-43. CERIAS. 1195

[34] Rogers, M.K.(2006) A two-dimensional circumplex approach to the development of a hacker taxonomy. Digital Invest., 3, 97–102.

[35] Alata, E., Nicomette, V., Kaaniche, M., Dacier, M. and Herrb, M. (2006) Lessons Learned from the Deployment of a High-Interaction Honeypot. EDCC'06: Proc. 6th European 1200 Dependable Computing Conf., pp. 39–46. IEEE Computer Society.

[36] Sanders, W.H. and Meyer, J.F. (2002) Stochastic Activity Networks: Formal Definitions and Concepts. Lectures on Formal Methods and Performance Analysis: First EEF/Euro Summer School on Trends in Computer Science, pp. 315–343.

[37] Deavours, D.D., Clark, G., Courtney, T., Daly, D., Derisavi, S., Doyle, J.M., Sanders, W.H. and Webster, P.G. (2002) The Möbius framework and its implementation. IEEE Trans. Softw. Eng., 28, 956–969.

[38] Arora, A., Krishnan, R.,Telang, R. and Yang, Y.(2004) Impact of Vulnerability Disclosure and Patch Availability—an Empirical Analysis. 3rd Workshop on the Economics of Information Security.

[39] Frei, S., May, M., Fiedler, U. and Plattner, B. (2006) Large-1215 Scale Vulnerability Analysis. LSAD'06: Proc. 2006 SIGCOMM Workshop on Large-Scale Attack Defense, pp. 131–138.

[40] Ozment, A., Schechter and Stuart, E.(2006) Milkor Wine: Does Software Security Improve with Age? USENIX-SS'06: Proc. 15th Conf. on USENIX Security Symp., Berkeley, CA, USA. USENIX Association.

[41] Symantec Enterprise Security. (2012) Symantec global internet security threat report—Volume 17—2011 trends.

[42] IBM. X-force trends reports. http://www935.ibm.com/ services/ us /iss/xforce/ trendreports/.

[43] IBM Global Technology Services. (2009) Ibm internet security systems x-force 2008 trend & risk report.

[44] National Institute of Standards and Technology. National vulnerability database. http://nvd.nist.gov.

[45] Security Focus. Security focus vulnerability database. http://www.securityfocus.com.

[46] Secunia. Secunia vulnerability database. http://secunia.com/.

[47] Jumratjaroenvanit, A. and Teng-amnuay, Y. (2008) Probability of Attack Based on System Vulnerability Life Cycle. ISECS'08: Proc. 2008 Int. Symp. on Electronic Commerce and Security, Washington, DC, USA, pp. 531–535. IEEE Computer Society.

[48] Sturges, H.A. (1926) The choice of a class interval. J. Am. Stat. Assoc., 21, 65–66.

[49] Mathwave. The easyfit tool. http://www.mathwave.com.

[50] Kuhl, M.E., Kistner, J., Cotantini, K. and Sudit, M. (2007) Cyber Attack Modeling and Simulation for Network Security Analysis. WSC'07: Proc. 39th Conf. on Winter Simulation, Piscataway, NJ, USA, pp. 1180–1188. IEEE Press.

[51] McQueen, M., Boyer, W., Flynn, M. and Beitel, G. (2006) Time-to-Compromise Model for Cyber Risk Reduction Estimation, Quality of Protection, pp. 49-64, Springer, USA

[52] ISO/IEC 27001. (2005) Requirements for information security management systems.

[53] ISO/IEC 27002 (2005) Code of practice for information security management.

.