Computers and Mathematics with Applications 62 (2011) 4308–4321

Contents lists available at SciVerse ScienceDirect

Computers and Mathematics with Applications

journal homepage: www.elsevier.com/locate/camwa

A hybrid ranking approach to estimate vulnerability for dynamic attacks

Feng Zhao [a,b,c,*], Heqing Huang [a], Hai Jin [a,b,c], Qin Zhang [a,b,c]

[a] School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan 430074, China

[b] Services Computing Technology and System Lab, Wuhan 430074, China

[c] Cluster and Grid Computing Lab, Wuhan 430074, China

ARTICLE INFO

Article history:
Received 23 April 2010
Accepted 15 September 2011

Keywords:
Security evaluation
Hybrid ranking
Attack graph
CVSS
Dynamic scenarios

ABSTRACT

To enhance security in dynamic networks, it is important to evaluate the vulnerabilities and offer economic and practical patching strategy since vulnerability is the major driving force for attacks. In this paper, a hybrid ranking approach is presented to estimate vulnerabilities under the dynamic scenarios, which is a combination of low-level rating for vulnerability instances and high-level evaluation for the security level of the network system. Moreover, a novel quantitative model, an adapted attack graph, is also proposed to escaping isolated scoring, which takes the dynamic and logic relations among exploits into account, and significantly benefits to vulnerability analysis. To validate applicability and performance of our approach, a hybrid ranking case is implemented as experimental platform. The ranking results show that our approach differentiates the influential levels among vulnerabilities under dynamic attacking scenarios and economically enhances the security of network system.

1. Introduction

Estimating the influences of vulnerabilities under network scenarios has become more and more complicated, primarily, owing to the complex multi-steps and multi-hosts exploitations deployed by the system attackers. Scoring systems that separately consider vulnerabilities are currently deemed insufficient. Despite the dominance of scanner-like *Nessus*, a method to identify vulnerabilities using predefined rules, it was observed that network administrators could only separately score the scanned vulnerability through *CVSS* (common vulnerability scoring system) [1], given security metrics provided by *NVD* (national vulnerability database) [2,3]; however, they cannot fully deliver optimal security measures for dynamic network system. Thus, it is essential to propose an overall security rate reflecting the secure level of dynamic network at a particular scenario to accurately estimate the global influential levels of exploits.

To evaluate the security rate of a dynamic system, an appropriate model that considers various aspects and factors, and which likewise mimics the attacker's choice of exploits, should be set up. Researchers have conducted remarkable work on generating and ranking attack paths to analyze and model such multi-step and multi-agent attacks in order to provide high-level security metric for risky measurement. In this study, we propose an attack graph model that mimics the attacker's attacking strategy, and then chose a mathematical method to quantify the security level of a network scenario after assigning weight to each edge (vulnerability instance) in the attack graph. However, rather than focusing on optimizing attack graph generating algorithms or designing attack graph semantics, a different work from previous works mentioned in [4], we assumed that the attack graph to describe the attacking scenario, could be acquired from existing attack graph generation tools like *TVA*, the topological vulnerability analysis system [5]. The main goal of this study is to prioritize the

* Corresponding author at: School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan 430074, China.
E-mail addresses: zhaof@hust.edu.cn (F. Zhao), hjin@hust.edu.cn (H. Jin).

0898-1221/$ – see front matter © 2011 Elsevier Ltd. All rights reserved.
doi:10.1016/j.camwa.2011.09.031

vulnerability instances by modeling the elimination of their influences on the estimated network scenarios. And the main contribution of our work is to render a hybrid security metric for quantifying and ranking the security level of particular network system under particular time and situation. Combining the graph generation and analysis tools [6,7] with security metrics, it is effective to weigh the pros and cons of decisions on choosing patching orders of these vulnerabilities, and take more economic and efficient hardening strategy for given network scenarios.

The rest of this paper is organized as follows: Section 2 introduces the basic concepts and hybrid ranking approach; Section 3 describes our implementation of *CVSS* for the low-level rating; and thereafter our method for high-level evaluating via modeling the selected attacking strategy is given in Section 4. In Section 5, a round estimation sample of an enterprise network is rendered and thus concrete case study is performed and its results are analyzed. In Section 6, related works are introduced and compared with our work. Finally, conclusion and future work is provided in Section 7.

2. Hybrid ranking approach

2.1. Basic concepts

To describe the framework and make our study more clearly, in this section, we first introduce some basic concepts and give all notations used in this paper.

Definition 1. Adapted attack graph (A-AG) is an adjustable attack graph, composed by two types of nodes and two types of arcs, which is a valuable tool to network defenders, illustrating paths an attacker can use to gain access to a targeted network.

There are two types of nodes in A-AG: state node and vulnerability instance node. State node represents an attacker's access privilege on host in a given network scenario. Outgoing arcs from state nodes point to the vulnerability instances that can be exploited by an attacker. State node in A-AG shares almost the same meaning as that in MP-AG (multi-prerequisite attack graph) [8], but the difference lies in the outgoing arcs from state node. In A-AG, outgoing arc from state node is a precondition arc, which emphasizes the leading precondition for triggering the next vulnerability instance. However, in MP-AG all arcs are meaningless. Source state node represents a state from which an attacker can trigger several vulnerability instances. In A-AG, the source state node is connected to several vulnerability instances through the corresponding precondition arcs. Destination state node represents a state that an attacker can arrive at after the chosen vulnerability instance is issued successfully. In A-AG, the vulnerability instance leading to a destination state node is connected by the transferring arc. Vulnerability instance node represents the vulnerability instance that can be triggered from a certain source state node if given enough preconditions. Each outgoing transferring arc from vulnerability instance node points to a destination state node that the attacker can reach after successfully exploiting the related vulnerability instance. Vulnerability instance node in A-AG has the same meaning as the vulnerability instance node of the MP-AG model [8].

An example of A-AG is shown in Fig. 1, which is inspired from the structure of MP-AG, providing an appropriate description of the attacking process and an illustration of the whole network attacking scenarios. As shown in Fig. 1, *state node* is depicted as a circle and the *vulnerability instance node* is depicted as a triangle. To indicate attacking path, two types of arcs are stretched out to connect and show the consecutive order of the nodes, which are *precondition arc*, depicted by white arrowhead, and also a *transferring arc*, depicted by black arrowhead.

Definition 2. Vulnerability instance *(Vul_Inst)* is a series of actions deployed by an attacker when performing an atomic attack to issue certain vulnerability. In attack graph, a *Vul_Inst* is described through its corresponding vulnerability instance node combined with its precondition arc.

Definition 3. Attacking path specifies a path in an attack scenario that leads to an escalation of privilege sets from network hosts.

Attacking path shows how an attacker can level up his or her access to a network system. A-AG uses the consecutive order of state nodes, precondition arcs, and *Vul_Inst* nodes to describe the dynamic attacking paths of network scenarios.

Definition 4. Crucial rate (δ) is assigned to *Vul_Inst* in order to numerically describe its crucial level.

Crucial rate emphasizes both vulnerability's impact and its frequency of being exploited. We use vulnerability impact rate (*VIR*) to measure the harmful possibility to a host if the vulnerability is utilized by an attacker. Vulnerability exploitability rate (*VER*) ponders the difficulty of exploiting a vulnerability, which depends on the host's configurations and the complexity of deploying the vulnerability by an attacker. The higher the *VIR*, the more harm the vulnerability brings to the host; the higher the *VER*, the more exploitable is the vulnerability, and the exploitation is more frequent. Therefore, we use both the *VIR* and *VER* of vulnerability to describe the crucial level of single vulnerability. However, every *Vul_Inst* is a different case under a specific attacking scenario. Crucial rate should reflect the difference between vulnerability instances, even when they are corresponding to the same vulnerability. The stakeholder can assign weights to *VIR* and *VER*, to differentiate vulnerability instances from different source states and related destination states.

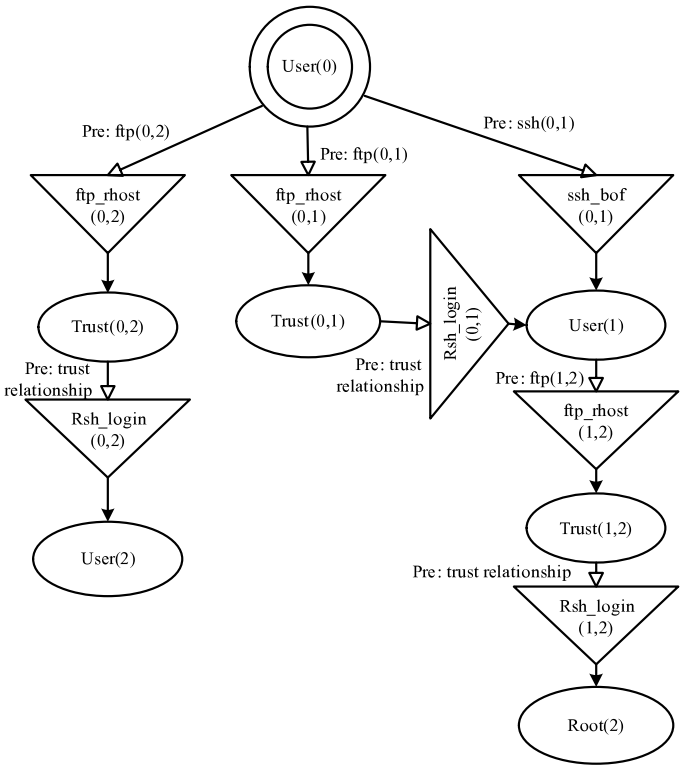


Fig. 1. An example of an adapted attack graph.

Definition 5. *Vul_Inst* elimination is an action that the administrator performs to fix the vulnerability in a certain host by patching or reconfiguring the system.

Definition 6. Dwindled attack graph is a degraded attack graph that generated after a time of *Vul_Inst* elimination has been completed and a new corresponding network scenario has been formed.

Definition 7. Original attack graph is a complete attack graph that generated without performing any elimination.

With *Vul_Inst* elimination, the whole network scenario would be adapted. In estimation process, *Vul_Inst* elimination acts as removing the *Vul_Inst* and its successive nodes and arcs of the original attack graph. However, if the successive attacking paths are available from other state nodes, they should be remained, since it indicates that the *Vul_Inst* remained can be triggered from other attacking paths. An example of *Vul_Inst* elimination is shown in Fig. 2, where the elimination of *Vul_Insts*, *Ftp_rhost* (1, 2) has no influence on its successive vulnerability instance node *Local_setuid_buffer_overflow* (2, 2), because this vulnerability instance node is available from other attacking paths.

Original attack graph illustrates the original attacking status and secure level of network scenario. When estimating vulnerabilities, *Vul_Inst* elimination is pondered by analyzing the different security rates between original AG and dwindled AG. With this comparison, whether or not such a patching is applicable can be figured out. Thus, this study focus on the quantification for the security rates related to the corresponding dwindled attack scenarios so as to provide a detailed rank of all the picked vulnerability instances for making better hardening strategy under the whole network scenario. In estimating approach, security rate of each dwindled attack graph is used to weigh *Vul_Inst* elimination, since a higher security rate means more enhancement of security after the elimination. With that rank, it is easy to compare the influence of vulnerability instances under network scenarios, and thus, prioritize them for network administrator.

2.2. Framework of hybrid ranking approach

Security issues can be considered as a dynamic process of preventing harm to the confidentiality, availability, and integrity of information and services [1,6], but in a static and operational way. Therefore, it is possible to estimate the influence of vulnerability by simulating the dynamic process of the attacker's multi-step attacks under dynamic network scenarios. In this section, we propose a hybrid ranking approach to estimate vulnerability, combining low-level rating via *CVSS* and high-level evaluation for the security rate, which provides numerical estimation of the vulnerability influence on the global network scenario by ranking of the dynamically adapted dwindled attack graph. Before hybrid ranking, if

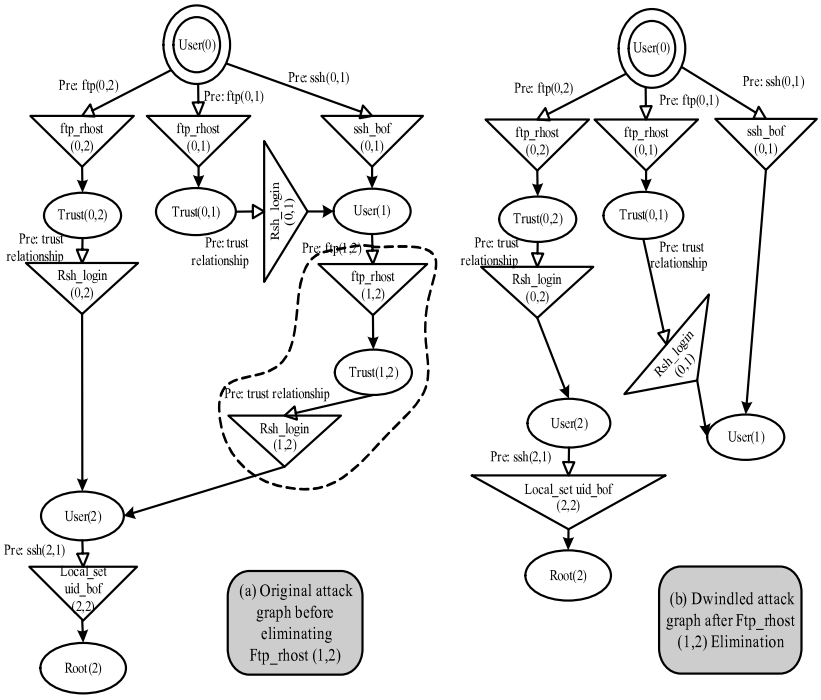


Fig. 2. An example of *Vul_Inst* elimination.

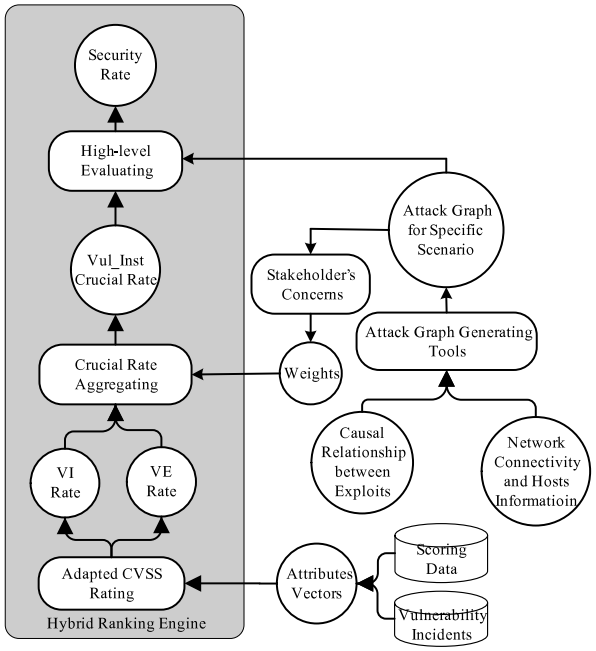


Fig. 3. Framework of hybrid ranking.

parts of the scanned vulnerability instances were evaluated to obtain an influential rank for patching, the eliminations should be performed on these picked vulnerability instances. After eliminations, the original attack graph is converted to the corresponding dwindled attack graph. Thus, the security level of the global network system can be checked via the dwindled attack graphs. The higher the security level of the converted network scenario after elimination, the more influential the vulnerability instances to the network before elimination. Fig. 3 illustrates the framework of proposed hybrid ranking approach.

The hybrid ranking method generates a rank to illustrate the different influential level of vulnerability instances under checked network scenario. It is helpful to render better hardening strategies for vulnerability instances in the estimated network. The executing process of hybrid ranking for vulnerability instance is as follows:

- Use *CVSS* rating module to rate *VIR* and *VER* of scanned vulnerabilities in a given network scenario via their attributes' metrics, which can be set from the attributes vectors retrieved from *NVD* and former published vulnerability incidents scoring data.
- Generate the original attack graph, describing the network attacking scenario, through the connectivity information and the causal relationships of scanned exploits using attack graph generation tool, such as TVA-AG presented in [5].
- Assign vector $[\alpha, (1-\alpha)]$ to *VIR* and *VER* for all vulnerability instances, which appear in the original attack graph with stakeholders specific concerns on vulnerability impact (*VI*) and vulnerability exploitability (*VE*) of each *Vul_Inst* in particular attacking path, namely the source state and destination state of each *Vul_Inst*. The vector $[\alpha, 1-\alpha]$ is defined as the stakeholder's weight vector. For experiment, default assignment of this vector is $[1/2, 1/2]$.
- Compute the crucial rate δ of each *Vul_Inst* via *VIR* and *VER*, and generate the assigned weight vector for each scanned *Vul_Inst*.
- Execute a single elimination for vulnerability instances submitted by the security requirement each time. Then, a dwindled attack graph corresponding to the *Vul_Inst* elimination can be adapted from the original attack graph.
- Evaluate the security rate of each dwindled attack graph corresponding to its adapted attacking scenario and compare the risk level of these dwindled attacking scenarios by the security rate.
- Prioritize the influential level of vulnerability instances through the rendered rank of the security rate for each dwindled attack graph after performing the related *Vul_Inst* elimination.

3. Low-level rating via *CVSS*

This section reviews the 14 scoring attributes of *CVSS* and describes adaption approach including the process of obtaining VI and VE rates. Then, a method for further computation of crucial rate from *VI* and *VE* is detailed. Finally, the main criterion for high-level evaluation is also introduced.

3.1. A brief of CVSS

Common Vulnerability Scoring System (*CVSS*) is an open framework that addresses the issues to obtain unified scores for specific vulnerabilities in a common way. The score of *CVSS* can reflect the cumulative experience of the *CVSS*-SIG as well as extensive testing of real-world vulnerability incidents in end-user environments [1,6]. All the *CVSS* attributes vectors address the objective characteristic of vulnerabilities and the subjective factors of attackers are not taken into account. *CVSS*2.0 is an adaptable and standardized scoring method, which provides information about vulnerabilities on an operational level and leaves stakeholders to add the information specific for their own explanations and needs. *CVSS*2.0 offers relatively accurate evaluation metrics for security status evaluation, rather than categorizing attacks on a general basis or providing a general model for evaluating security rate. In this study, hybrid ranking method adapts *CVSS*2.0 for low-level rating for *Vul_Inst*, and the output of the low-level rating is taken as the evaluation metrics for high-level evaluating.

To make low-level rating more clear, we introduce three types of metrics and the attributes of*CVSS*2.0 as follows, which are the base metric (Bs), the temporal metric (Tp) and the environmental metric (Evr).

Base metric evaluates the original and immutable attributes of vulnerabilities in terms of vulnerability exploitability (*VE*) and vulnerability impact (*VI*). Exploitability includes access vector (Bs_AR), access complexity (Bs_AC) and authentication (Bs_AU). Impact includes confidentiality impact (Bs_C), integrity impact (Bs_I), and availability impact (Bs_A). These two aspects are what were emphasized in our low-level rating.

Temporal metric evaluates the dynamic aspects of vulnerability in terms of three attributes: Exploitability tools and techniques (Tp_E), remediation level (Tp_RL), and report confidence (Tp_RC). Tp_E is related to the VE aspect and other two is related to VI aspect.

Environmental metric evaluates three aspects of vulnerabilities that are dependent on the environment, which we involve them both in the *VE* and *VI* rating of our low-level rating methods: Collateral damage potential (Evr_CDP), target distribution (Evr_TD), and security requirements (Evr_SR).

3.2. Rating VI and VE by extending CVSS

Except for twelve attributes related to *VIR* or *VER*, two attributes, collateral damage potential (Evr_CDP) and target distribution (Evr_TD), emphasize both the *VIR* and *VER*. Both *VI* and *VE* are addressed in the basic metrics of *CVSS* as the fundamental assessing criteria for the objective aspects of vulnerabilities. However, *CVSS* focus on the overall ranking, which is too complicated and general for low-level ranking. In this study, we adopt a novel computation mode by extending *CVSS* metrics, which pay more attention on *VI* and *VE*.

The followings are the computational formulas related to the *VIR*:

$$AdjustedImpact = 10.41[1 - (1 - \text{Bs\_C} * \text{Evr\_CR}) * (1 - \text{Bs\_I} * \text{Evr\_IR}) * (1 - \text{Bs\_A} * \text{Evr\_AR})] \quad (1)$$

$$VIadapted = [(0.6 * AdjustedImpact) - 1.5] * f(\text{Impact}) * \text{Tp\_RL} * \text{Tp\_RC} \quad (2)$$

$$VIR = [VIadapted + (10 - VIadapted) * \text{Evr\_CDP}] * \text{Evr\_TD}. \quad (3)$$

The following formulas are related to the VER:

$$VEadapted = (0.4 * 20 * \text{Bs\_AR} * \text{Bs\_Ac} * \text{Bs\_AU}) * f(\text{Impact}) * \text{Tp\_E} \quad (4)$$

$$VER = [VEadapted + (10 - VEadapted) * \text{Evr\_CDP}] * \text{Evr\_TD}. \quad (5)$$

In hybrid ranking approach, *VIR* and *VER* are initial outputs of the low-level rating engine, as shown in Fig. 3. Both rates of vulnerability, combined with the stakeholder's specific concern for vulnerability instances, are further aggregated into crucial rate δ of *Vul_Inst*. For this aggregation, the stakeholder only has to assign the weight vector to a special emphasis on the concerned *Vul_Inst*. Our improvement provides a more convenient way for stakeholder to define rating vectors and the input of the high-level evaluation can be easily obtained. Furthermore, the improvement can correspond to the *CVSS* overall score well, which has been demonstrated through our tests on different sample vulnerability instances. We will detail the accuracy of our rating in the analysis of the ranked cases.

3.3. Rating crucial rate

Crucial rate, the final output of the low-level rating, is also taken as the input of the high-level evaluation. *Crucial rate* is generated from *VI* and *VE*, both of which are objective factors addressed by *CVSS*2.0. As mentioned above, *VI* is related to the impact level to a host, given that a particular vulnerability is issued successfully; *VE* is related to the frequency of attacker's triggering certain vulnerability. The more exploitable the vulnerability, the more frequently it may be triggered, and thus, the higher crucial rate. The crucial rate reflects not only the impact level and the frequency rate of vulnerability in a statistical way via the uploaded *CVSS* scores, but also the stakeholder's specific concerns for vulnerability instances under a certain network scenario. Therefore, weights can be assigned to *VIR* and *VER*, respectively, for a particular emphasis. So, the crucial rate of *Vul_Inst* can be numerically defined through *VIR*, *VER*, and the weight vector:

$$\delta = \alpha VIR + (1 - \alpha)VER, \quad \alpha \in [0, 1], \quad (6)$$

where $[\alpha, 1 - \alpha]$ is the weight vector addressed by stakeholders for their specific considerations of each *Vul_Inst* between the consecutive source state and destination state along the attacking path. In this paper, default setting to weight vector is [0.5, 0.5], if no particular vector assigned. Base on formula (6), the security stakeholder can conveniently define the weight for each *Vul_Inst* with one assignment, and need not to care about the *CVSS* attribute metrics. Furthermore, crucial rate acts as the input metrics of the high-level evaluation, which is defined numerically.

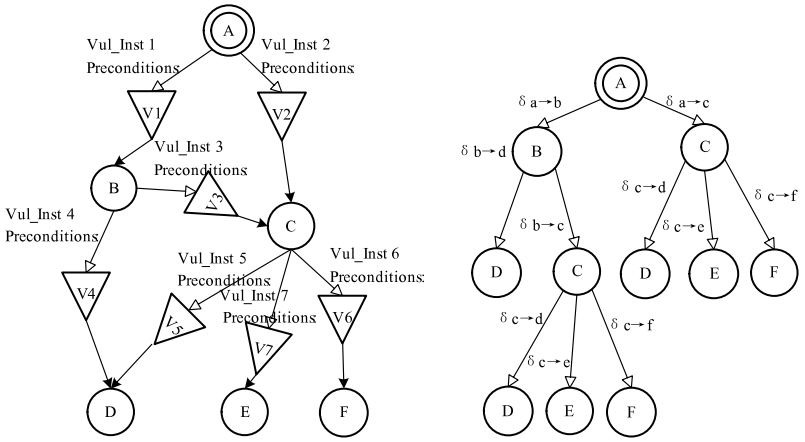
There is a primary assumption that the more crucial a *Vul_Inst*, the less expenditure is required for its exploitation. Here, the expenditure is the probability for successfully cracking vulnerability in given time. Thus, there can be an invert ratio between the crucial rate and the attacker's expenditure given in each exploit. To achieve hybrid ranking, for low-level rating, the objective factors is measured using the Formulas (1)–(6) to get crucial rate; for high-level evaluation, the mean expenditure given by an attacker for triggering certain *Vul_Inst* is considered as the main criteria for describing the attackers' subject factors. So, in next section we evaluate the subjective factors by mimicking a common attacking strategy of attackers, the one-step-forward strategy and a Markov stochastic model is chosen to describe the state-transition rate related to both the mean expenditure and the vulnerability instances' crucial rates. In this way, a numerical evaluation of the security level of related network scenario can be rendered.

4. High-level numerical evaluation

In high-level evaluation, discovering the attackers' strategy in selecting *Vul_Inst* is a highly difficult and critical task. Provided that there were available preconditions, the attackers' choice of vulnerability instances should not be a random process. In low-level rating, we take into account the objective factors addressed by *CVSS*2.0 in order to define vulnerability instance's crucial rate, which is taken as the bridge for low-level rating and high-level evaluating. In this section, detailed high-level modeling objectives and evaluation methodology is further expounded based on A-AG mentioned above.

4.1. Stimulating attacking path via OSFS

To simulate attacking path, lots of works have addressed on the attack strategy model, such as shortest path strategy (*SPS*) and one-step-forward-strategy (*OSFS*) [9]. To implement *SPS*, the attacker should grasp the global system network, deploying the most desirable attacks in their shortest path. Bu it is very difficult or even unnecessary for the attacker to obtain this comprehensive information prior launching attacks. Even assuming some attackers could identify the shortest attacking paths and deliberately deploy consecutive attacks, the number of vulnerability instances is quite small, since there are few shortest attack paths or a probability of a single path to be identified. With these limited vulnerability instances along



(a) A-AG model. (b) Stretching-out tree.

Fig. 4. An example of A-AG and stretching-out tree.

this few shortest attack-paths, it is difficult to rank all the vulnerability instances useful for security analysis. Therefore, we pick up *OSFS* as a compensation for *SPS*, in terms of these above shortcomings when modeling attack strategies via *SPS*.

With this strategy, the attacker plunders target privileges and assets in each host step by step. At each stage of the attacking process, only the available vulnerability instances in front of the source state node are considered. So in this study, several assumptions regarding *OSFS* are made to define subjective factors of the attackers. Based on different objectives to describe the subjective factors related to the attacking strategy, different assumptions can be made.

First, the attacker chooses one *Vul_Inst* among exploitable vulnerabilities that are one step ahead without considering the expenditure of further attacks. Therefore, the higher the crucial rate of a *Vul_Inst*, the less expenditure is needed for exploiting, and the more willingness of the attacker to choose the related *Vul_Inst*. Hence, the conditional probability transition between two consecutive states addressed in the evaluation model should be able to reflect this attacking strategy. If an elementary attack for the chosen *Vul_Inst* succeeded, the attacker should be expected to move forward to destination state, that is, the attacker would prefer exploiting vulnerability instances that are one step ahead. Attackers are not expected to return to previously available vulnerability instances that were not set off, who only focus ahead with no memory or consideration of the past. So, when using A-AG as high-level evaluation, backward arcs are not addressed and a depth-first-traverse is performed to extend attacking paths. Furthermore, it is always to ignore the situations, an attacker gives up halfway, since traditional attack graph does not taken halfway behaviors into when modeling objectives. A-AG considers every state as target that is similar in the real world wherein every state is meaningful for an attacker, or at least can be taken as a stepping-stone for further attacks. So, merit is assigned to every state in A-AG model, whether or not it is considered a terminal state.

It deems that an attacker would finally reach a state, given a path leading to the state exists and enough expenditure is rendered. So, a depth-first-traversing manner is preferred to describe the OSFS since complete attacking paths can be investigated. Under this manner, the attack graph stretches out like a tree to reflect the modeling objective, and all vulnerability instances and attacking paths are fully illustrated. An example of stretching-out tree, as shown in right of Fig. 4, is formed by the depth-first traversing of the attack graph model. Every leaf node represents the end of one possible attacking path. Each edge in the stretching-out tree has been assigned a crucial rate δ to its corresponding Vul_Inst. Taking node C as an example, three vulnerability instances with crucial rates of $\delta_{c\to d}$, $\delta_{c\to e}$, $\delta_{c\to f}$ can be triggered from this node. After selecting and successfully infiltrating one of these vulnerability instances, the attacker can gain more privileges of certain hosts. A privileges escalation will be produced as attacker's step into a deeper destination node among *D*, *E* and *F*.

With A-AG introduced in Section 2.1 and the more illustrative stretching-out tree, it is possible to stimulate the attacking strategy OSFS graphically and define the conditional probability of state-transition. Each state node represents sets of privileges in different hosts that the attacker has already captured. Transitions between source and destination states appear when the attacker chooses and succeeds in exploiting corresponding Vul_Inst, permitting to continue plundering other privileges and indicating that state nodes are continuously seized. The conditional probability of the state-transition can be numerically defined through the evaluated objectives and chosen Markov stochastic model. In that case, the process for attacker's privileges escalation can be quantitatively measured through high-level evaluation.

4.2. Numerical definition of security rate

In order to numerically evaluate the security rate of network scenario related to each dwindled attack graph, a Markov stochastic model has been chosen in high-level evaluation [9]. The implementation of this model has been addressed in a previous work for computer security level estimation [10]. Considering it is a common stochastic model for security

evaluation, we combine it with vulnerability instances rating via *CVSS* and the modeling of A-AG so as to provide a further attempt on network security assessment.

In this paper, exponential distribution assumption is employed to quantify the mean expenditure rendered by the attacker in each exploit. In the exponential distribution, $p_V(\varepsilon)$ is the probability of triggering V, the *Vul_Inst*, for state transition between source state and destination state, while ε is the amount of expenditure given when exploiting *Vul_Inst* and forwarding into the destination state node. Thus we have:

$$p_V(\varepsilon) = p(Ex < \varepsilon) = 1 - e^{-\delta\varepsilon}, \tag{7}$$

where *Ex* is the equivalent expenditure from an attacker after successfully exploiting the desired *Vul_Inst* and thus advancing into the destination state, δ represents the crucial rate corresponding to the given *Vul_Inst*.

The reason employing exponential distribution function is following: Firstly, assuming an atomic attack of certain *Vul_Inst* fails, the attacker can still maintain its former state, which corresponds to the "memory less property" of the exponential distribution. Secondly, given available attacking paths to target state nodes, and provided enough expenditure ε, that means $\varepsilon \to \infty$ and $p(\varepsilon)|_{\varepsilon\to\infty} = 1$. With these prerequisites, certain target state node can be conquered. Furthermore, with consideration of specific attacking scenario, crucial rate δ is assigned to each *Vul_Inst*, which can reflect the crucial level of certain *Vul_Inst* separately. As the expenditure is described via exponential distribution, the mean expenditure to succeed in a given atomic attack is $1/\delta$, corresponding to the assumption that the higher crucial rate of a certain vulnerability instance, the less mean expenditure is required for its exploitation. Also, we can define $\varepsilon = 0$, $p(\varepsilon)|_{\varepsilon=0} = 0$, that means attackers cannot trigger vulnerability instant without expenditure provided.

When quantifying the security level, the main criterion for evaluation is deemed as the mean expenditure (*MEx*) rendered when issuing vulnerability instances by an attacker in order to conquer the target states. As mentioned above, every *Vul_Inst* node in the attacking scenario is assigned with a crucial rate δ, which has an invert ratio to the mean expenditure. Therefore, the *MEx* of the whole network can be derived from the sum of the mean expenditure $1/\delta$, which is given by the attacker when conquering successive destination state nodes in the attacking paths, also see in Fig. 4. This summing-up complements our evaluation objective, to measure every state node in the given attacking scenario but not merely the single terminal state or single shortest attacking path.

Supposing $\delta_{s\to d}$ is the specific crucial rate of the *Vul_Inst* between the source state s and destination state d, we can have a detailed definition of $\delta_{s\to d}$ in our high-level quantification: $\delta_{s\to d} = \alpha_{s\to d}VIR + (1 - \alpha_{s\to d})VER$, $\alpha_{s\to d} \in [0, 1]$. With this definition of each *Vul_Inst*'s crucial rate, we can combine low-level rating and high-level evaluating together for the quantification of *MExC* for each dwindled attack graph and use this to rank the influential level of each corresponding *Vul_Inst*. The mean expenditure rendered in state node s is denoted as *MEx_s*, which is related to all the vulnerability instances that state s can reach within one step, just like all the outgoing edges from father node C to son nodes D, E and F illustrated in the tree of example in Fig. 4. As the inverse ratio between crucial rate δ and the mean expenditure, *MEx_s* is calculated by the inverse of the summing-up for all the vulnerability instances' crucial rates related to state s, thus *MEx_s* is given by:

$$MEx\_s = \frac{1}{\sum\limits_{d\in dest(s)} \delta_{s\to d}}, \tag{8}$$

where the set of destination state nodes directly related to the current source node s by different vulnerability instances in the attack graph is defined as $dest(s)$.

With stretching-out tree, the one-step-forward strategy for state transition can be illustrated. To define the conditional probability transition from source state s to destination state d, we use $\delta_{s\to d} \times MEx\_s$. A detailed explanation can be provided through the stretching-out tree. The *MEx_s* of a father node s is determined by all direct related son nodes, thus the larger crucial rate assigned to the edge related to a particular son node, the less mean expenditure required and the larger possibility of the particular *Vul_Inst* on the edge being chosen and triggered by the attacker:

$$\delta_{s\to d} \times MEx\_s = \frac{\delta_{s\to d}}{\sum\limits_{d\in dest(s)} \delta_{s\to d}}. \tag{9}$$

As Markov stochastic model is taken as the mathematical model, the quantification of the network security level can be derived by the following iteration formula (10), node s is taken as the father node in the stretching out tree, then denoted *MEx_s* as the total mean expenditure required from node s to all its leaf nodes. Therefore we have *MEx_s*:

$$MEx\_s = \sum_{i\in dest(s)} \frac{\delta_{s\to i}}{\sum\limits_{i\in dest(s)} \delta_{s\to i}} \times MEx\_i + \frac{1}{\sum\limits_{i\in out(s)} \delta_{s\to i}}, \tag{10}$$

where $\frac{1}{\sum_{i\in out(s)} \delta_{s\to i}}$ represents the mean expenditure rendered at the current state node s, $out(s)$ is the set of all destination nodes connecting with source state node s. The iteration part addresses the summing-up of the mean expenditure given at each destination state node that is one step ahead. With this numerical evaluation of the mean expenditure, we can describe

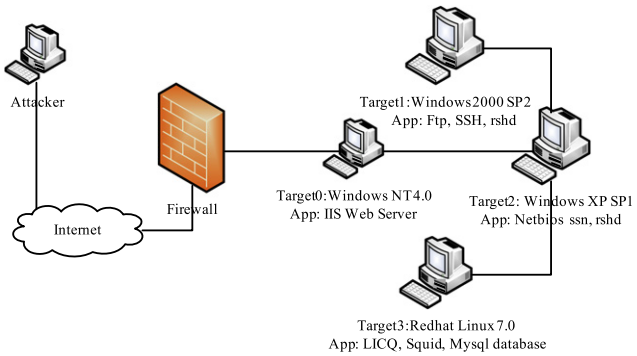


Fig. 5. Network topology of ranking example.

Table 1
Targets description of ranking example.

Host	Services	Vulnerability	OS
T_0	1. IIS_Web_Service	IIS buffer-overflow	Window NT 4.0
T_1	1. ftp 2. ssh 3. rshd	ftp-rhost overwrite ssh buffer-overflow rsh-login	Windows 2000 SP2
T_2	1. Netbios_ssn 2. rshd	Netbios_ssn_nullsession	Windows XP SP1
T_3	1. LICQ 2. Squid_Proxy 3. Mysql_DB	LICQ-remote-to-user Squid-port-scan Local-setuid-buffer-overflow	Red Hat Linux 7.0

the whole security rate of the network scenario via *MEx_ini*, where *ini* refers to the initial state node from which the attacker can initiate its consecutive attacks. The higher the *MEx_ini*, the more expenditure for attackers to plunder the states and the better the whole security status is.

With quantifying security rate, the risk level of the dwindled network scenario can be evaluated. As illustrated by each *MEx_ini* for a dwindled attack graph corresponding to the dwindled attacking scenario. The higher a *MEx_ini*, the more secure a status after performing certain *Vul_Inst* elimination and the more influential the *Vul_Inst*. Using this evaluating methodology, we can perform the hybrid ranking procedure to estimate the influential level of each *Vul_Inst* on the whole network through the related *MEx_ini* required by network. Our hybrid ranking approach is an efficient complement for the static scoring of vulnerability. In an estimation procedure, not only is the crucial rate of each *Vul_Inst* calculated, but also the security rate is quantified to indicate the influential level of each *Vul_Inst* in the estimated network system. Moreover, the order for patching these vulnerability instances can be rendered according to the ranking of related security rate of dwindled attack scenarios.

5. Experimental results

To validate proposed hybrid ranking approach for the estimation of influential level of vulnerability instances, a hybrid ranking engine (HRE) has been developed to test several sample networks. In this section, a round and concrete case study is depicted to show the performance of our hybrid ranking approach. Comparisons between our hybrid ranking and *CVSS* scoring are also rendered and analyzed. Experimental results demonstrate that our approach can be a complementary work to the *CVSS* scoring in measuring the influential level of vulnerability instance.

The topology of ranking example network is shown in Fig. 5, which is the same as the network topology used in [11]. There are four target hosts in internal network and a hacker host is in the external network separated by a firewall. The internal target hosts' information is show in Table 1, where *T* means target host.

The firewall in this sample network permits external hosts to connect to IIS web service running on T_0, and connection to all other services are blocked. The internal hosts are permitted to connect mutually within the internal network. The connectivity among each host is given in Table 2, where S_i represents the source host i and thus T_i represents the target host i which contains several services that may be opened to a certain source host. The number 1, 2 and 3 represent the open services which can be referred to the numbers assigned for each T_i in Table 1; -1 represents no warranted connection allowed according to the firewall strategy; 0 means a self-connection.

Table 3 shows eight example vulnerabilities and their basic vectors retrieved from *NVD* and the assembled temporal and environmental vectors from the samples. Fig. 6 is an original attack graph generated according to the above sample network in Fig. 5 with no *Vul_Inst* elimination performed. Nine attacking paths can be investigated in the original attack

Table 2
Connectivity of ranking example.

Host	Attacker	T_0	T_1	T_2	T_3
Attacker	0	1	−1	−1	−1
S_0	−1	0	1, 2, 3	1, 2	1, 2, 3
S_1	−1	1	0	1, 2	1, 2, 3
S_2	−1	1	1, 2, 3	0	1, 2, 3
S_3	−1	1	1, 2, 3	1, 2	0

Table 3
Vulnerability attributes vectors.

Vulnerability CVE IDs	Basic vectors	Environmental vectors	Temporal vectors
CVE-2001-1030	AV:N/AC:L/Au:N/C:P/I:P/A:P	E:P/RL:O/RC:C	CDP:H/TD:H/CR:H/IR:M/AR:H
CVE-1999-1455	AV:N/AC:L/Au:N/C:P/I:P/A:P	E:H/RL:O/RC:C	CDP:H/TD:H/CR:M/IR:M/AR:M
CVE-1999-0180	AV:N/AC:L/Au:N/C:P/I:P/A:P	E:P/RL:O/RC:C	CDP:H/TD:H/CR:H/IR:M/AR:M
CVE-2003-0661	AV:N/AC:L/Au:N/C:P/I:N/A:N	E:F/RL:O/RC:C	CDP:H/TD:H/CR:H/IR:M/AR:M
CVE-2002-0364	AV:A/AC:L/Au:N/C:P/I:P/A:P	E:F/RL:O/RC:C	CDP:H/TD:H/CR:M/IR:H/AR:M
CVE-2006-3368	AV:N/AC:L/Au:N/C:P/I:N/A:N	E:H/RL:O/RC:C	CDP:H/TD:H/CR:H/IR:H/AR:M
CVE-2008-1396	AV:N/AC:M/Au:N/C:P/I:N/A:N	E:F/RL:U/RC:C	CDP:H/TD:H/CR:H/IR:M/AR:M
CVE-2001-0439	AV:A/AC:L/Au:N/C:P/I:P/A:P	E:H/RL:O/RC:C	CDP:H/TD:H/CR:M/IR:H/AR:M

Table 4
Results of hybrid ranking.

Removed *Vul_Inst* (Ranking in order)	$S \rightarrow D$	*MEx_ini*	CVE IDs
1. IIS_buffer_overflow (0, 0)	0 → 1	$+\infty$	CVE-2002-0364
2. Squid_port_scan (0, 3)	1 → 4	1.042701	CVE-2001-1030
3. Squid_port_scan (1, 3)	3 → 10	1.004924	CVE-2001-1030
4. Squid_port_scan (1, 3)	9 → 10	0.993117	CVE-2001-1030
5. SSH_buffer_overflow (0, 1)	1 → 3	0.941156	CVE-1999-1455
6. SSH_buffer_overflow (0, 2)	2 → 8	0.927974	CVE-1999-1455
7. Netbios-ssn-null session (0, 2)	1 → 2	0.924948	CVE-2003-0661
8. Netbios- ssn-null session (1, 2)	3 → 2	0.906139	CVE-2003-0661
9. ftp_rhost (0, 1)	1 → 6	0.895265	CVE-2008-1396
10. ftp_rhost (2, 1)	2 → 7	0.886722	CVE-2008-1396
11. LICQ_remote_to_user (1, 3)	10 → 11	0.870946	CVE-2001-0439
12. Squid_port_scan(2, 3)	2 → 5	0.836305	CVE-2001-1030
13. LICQ_remote_to_user (0, 3)	4 → 11	0.823693	CVE-2001-0439
14. Rsh_login (1, 1)	6 → 9	0.805711	CVE-1999-0180
15. Rsh_login (1, 1)	7 → 9	0.706708	CVE-1999-0180
16. Local-setuid buffer-overflow (3, 3)	11 → 12	0.677666	CVE-2006-3368
17. Squid_port_scan(1, 3)	8 → 10	0.653999	CVE-2001-1030
18. LICQ_remote_to_user (2, 3)	5 → 11	0.609754	CVE-2001-0439

graph by the *OSFS* depth-first-traversing. Moreover, it includes 13 state nodes, 18 preconditions arcs and 18 vulnerability instances related to the precondition arcs. However, only 15 *Vul_Inst* nodes appear, since it can be possible the case that several different vulnerability instances share the same *Vul_Inst* node with distinctions of state nodes and preconditions for triggering different attacks. Both the *Squid_port_scan* (1, 3) and the *Rsh_login* (1, 1) instance fall into this case.

After ranking 18 vulnerability instances of the sample network in Fig. 5 via HRE, the results are gained in Tables 4 and 5. In Table 4, the number in front of each *Vul_Inst* elimination indicates its rank after estimating the security rates of the related dwindled attack graph by HRE. In the column $S \rightarrow D$, S stands for source state node and D is the destination state node for the attacker to capture after succeeding in issuing certain *Vul_Inst*.

Comparing the ranking results of Tables 4 and 5 according to the adaptation of the given network system between original attack graph and all dwindled attack graphs, the following conclusion are get from experimental results:

Firstly, the experimental results indicate the efficient on our implementation of *CVSS* for rating the *VI* and *VE* respectively, and aggregating them into the *Vul_Inst*'s crucial rate δ is applicable.

Through the order of the 18 crucial rates of vulnerability instances in Table 4, there is a good corresponding to the ranking order of the 8 vulnerabilities comparing with overall *CVSS* scores. With the properly rated crucial rate, each *Vul_Inst* can be checked. Moreover, more accurate information of each *Vul_Inst*'s influential level using the high-level evaluation is grasped by attacking crucial rates to each *Vul_Inst* node in the attack graph model and performing high-level evaluation.

Secondly, our hybrid ranking approach is much more appropriate to dynamic network scenarios.

After checking the two tables via CVE IDs, we can find that the *CVSS* ranking of the vulnerability *IIS_buffer_overflow* is at 6th, shown in Table 5. In contrast, through our hybrid ranking, it comes to 1st as shown in Table 4, which corresponds well to the estimated attacking scenario. In original attack graph of Fig. 6, *IIS_buffer_overflow* (0, 0) is the first *Vul_Inst* to be triggered and it is the very critical for *Vul_Inst* to stand out that no attacker can ignore or bypass, while deploying attacks. Without

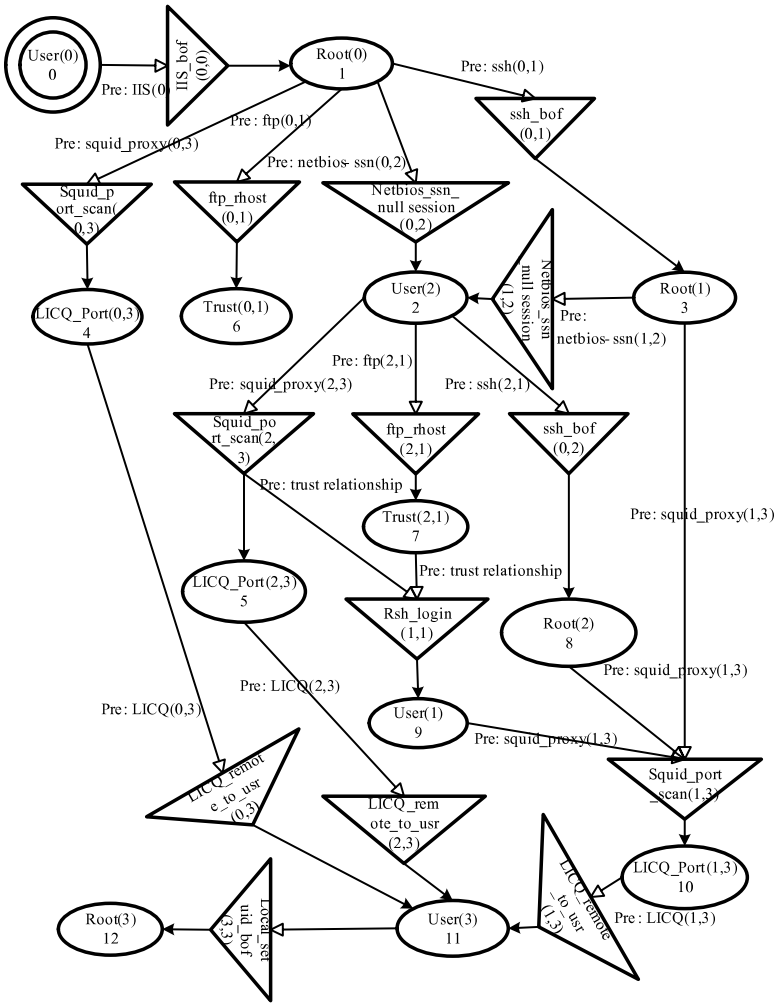


Fig. 6. The original attack graph for ranking example.

Table 5
Overall *CVSS* score and crucial rate.

CVE IDs	*CVSS result*	*VIR*	*VER*	$\delta_{s\to d}$	*Vul_Inst* Elimination
1.CVE-2001-1030	8.4	6.664	7.116	6.900 6.895 6.860 6.873 6.881	2. Squid_port_scan (0, 3) 3. Squid_port_scan (1, 3) 4. Squid_port_scan (1, 3) 12. Squid_port_scan(2, 3) 17. Squid_port_scan (1, 3)
2.CVE-1999-1455	8.3	6.210	7.351	6.678 6.667	5. SSH_buffer_overflow (0, 1) 6. SSH_buffer_overflow (0, 2)
3.CVE-1999-0180	8.2	6.446	7.116	6.781 6.781	14. Rsh_login (1, 1) 15. Rsh_login (1, 1)
4.CVE-2003-0661	7.8	5.559	7.234	6.484 6.480	7. Netbios_ ssn_null session (0, 2) 8. Netbios_ ssn_null session (1, 2)
5.CVE-2002-0364	7.8	6.446	6.519	6.471 6.471 6.469	13. LICQ_remote_to_user (0, 3) 18. LICQ_remote_to_user (2, 3) 11. LICQ_remote_to_user (1, 3)
6.CVE-2006-3368	7.7	6.446	6.443	6.445	1. IIS_buffer_overflow (0,0)
7.CVE-2008-1396	7.6	5.559	7.351	6.133	16. Local_setuid_buffer_overflow (3, 3)
8.CVE-2001-0439	7.5	5.643	6.919	6.217 6.205	10. ftp_rhost (2, 1) 9. ftp_rhost (0, 1)

issuing this *Vul_Inst* successfully, no deeper state nodes can be touched. Formulas (8) defines the *MEx_s*, through which we found $dest(s) = \emptyset$, which indicates that after this instance node is eliminated, a single initial source node is left and all its further state nodes turned unavailable for the attacker. In this case, our Markov stochastic model defines $MEx\_s \to \infty$, meaning $MEx\_ini \to \infty$, which can reflect the influential level of this initial and critical *Vul_Inst*.

Thirdly, our hybrid ranking approach can reflect the influential level of vulnerability instances more precisely, as the estimation is processed under the dynamic attack graph scenarios.

Two vulnerabilities *netbios_ssn_nullsession* and *LICQ-remote-to-user*, share the same overall *CVSS* score of 7.8, and given a coordinate rank 4th in Table 5. These sorts of coordinate rankings can happen frequently in *CVSS*, confusing on patching order and hindering decision on taking hardening strategies. Checking ranking results, 2 vulnerability instances of *netbios_ssn_nullsession* and 3 instances of the vulnerability *LICQ-remote-to-user* have been ranked. It is obvious that all the instances of *netbios_ssn_nullsession* are ranked prior to the instances of *LICQ-remote-to-user*. It indicates that based on the view of the whole security level, the elimination of each *netbios_ssn_nullsession* can provide more secure enhancement than the elimination of *LICQ-remote-to-user* instances.

The *MEx_ini* of the dwindled network scenarios is also checked. When comparing between the original attack graph and each related dwindle attack graph, all the three instances of *LICQ-remote-to-user* have only 2 state nodes ahead, state 11 and state 12 one *Vul_Inst* and one path in Fig. 6 for further attacking. While turning to the two *Vul_Inst* nodes of *netbios_ssn_nullsession*, both of them have 3 related paths including nine successive vulnerability instances and 8 state nodes. As the measuring objectives emphasize all the correlated states composing attack graph, it should take every available state node as a meaningful one no matter whether it is an intermediate state node or a terminal one. Based from the measuring objective, the more ahead state nodes shrink, the more improvement to the network security can be provided. Our ranking results correspond well to this measuring objective. Therefore, the ranking reflects more accurate prioritizations of these 2 sets of vulnerability instances, and it can obtain a better order for handling these nearly undistinguishable vulnerability instances.

Fourthly, the hybrid ranking approach can precisely distinguish different instances of the same vulnerability with consideration of the given global attack scenario.

Even the same vulnerability can be ranked differently via evaluating each after patching distinctive vulnerability instances in different paths. From all the 18 vulnerability instances, 5 are related to the vulnerability *squid-port-scan*, so we take this set of instances as an analysis example. 5 vulnerability instances can be triggered from state nodes: state 1, state 3, state 9, state 2 and state 8. The highest *MEx* of this set is provided by the elimination of *squid_port_scan* (0, 3) from state 1 and the lowest is from state 8. State 1 is the direct state node that an attacker can trigger *squid_port_scan* (0, 3). If eliminated, a path becomes unavailable including 2 successive state nodes, state 4 and state 11. On the contrary, the state 8's successive state 10 is still available from both the state 9 and state 3, after the elimination of vulnerability instances triggered from state 8. Moreover, to conquer state node 1, only one atomic attack should be issued from the state node 0. However, conquering state node 8 requires at least 3 previous atomic attacks being successfully exploited.

Finally, the different vulnerability instances for the same *Vul_Inst* node triggered from different state nodes can be prioritized through our hybrid system.

Two such cases appear in A-AG. One is *rsh_login* (1, 1) available from state node 6 and 7; the other is *LICQ_Port* (1, 3) reachable from state nodes 3, 9 and 8. As state 6 and state 7 share the successive vulnerability instance node, *rsh_login* (1, 1), all their successive paths bears no difference. In that case, the antecedent atomic attacks may contribute to their differentiation. The original attack graph illustrates that there is only one intermediate state node 1 from state node 0 to 6. However, for state 7, at least state 1 and state 2 or even state 3 should be obtained as preceding state nodes, which means less pre-expenditure needed for reaching state 6 than for reaching state 7. It also indicates that more security improvement can be gained to fix the *Vul_Inst* in state 6 than to eliminate the coordinate instance from state 7. The *MEx_ini* different rates of for these 2 *Vul_Inst* eliminations can reflect the difficulty of deploying attacks in the attacking scenario corresponding to the influential level of *Vul_Inst*.

6. Related work

Much previous work on operational vulnerability evaluation based on *CVSS* has been rendered. Aussibal and Gallon [12] describes a new distributed intrusion detection system based on *CVSS* framework. It employs the *CVSS* in a static scoring, through a local *CVSS* database. On the other hand, adaptations have been made on the *CVSS* for specific needs. Houmbet al. [13] estimates the impact and frequency rate of risky vulnerability via adapting the *CVSS*, and later Houmb has used this method in a target of estimation (ToE) risk level assessment work [14]. These works prove that the *CVSS* can be properly adapted and thus implemented based on stakeholders' specific demands. Our low-level rating module is also based on an appropriate adaptation of the *CVSS*.

Apart from this, various works on modeling specific security status of systems can be found. Mohammadi and Gharehpetian [15], for example, presents a core vector machine-based algorithm for on-line voltage security assessment of systems. To classify the system security status, a CVM has been trained for each contingency. The proposed CVM-based security assessment algorithm has a very small training time and space in comparison with support vector machines and artificial neural networks-based algorithms. Al-Kuwaiti et al. present a systematic approach for determining common and complementary characteristics of five widely-used concepts, dependability, fault-tolerance, reliability, security, and

survivability, which consists of comparing definitions, attributes, and evaluation measures for each of the five concepts and developing corresponding relations [16]. Study [8] has created a multiple-prerequisite attack graph (MP-AG) that scales nearly linear as the size of a typical network increases. The proposed attack graph uses a depth first algorithm to create the predictive attack graph. We have adapted the MP-AG model to define our attack graph model and a stretching tree can be extended from the depth-first-traverse manner. Ammann et al. revisit the idea of attack graphs themselves and argue that they represent more information explicitly than is necessary for the analyst [17]. They propose a more compact and scalable representation, which relies on an explicit assumption of monotonicity, which states that the precondition of a given exploit is never invalidated by the successful application of another exploit. Sheyner [18] presents an attack graph to model exploitations of the security vulnerabilities in a system. An attack graph is considered as data structure, which succinctly represents all paths through a system, which all end in a single state the final target of an attacker. However, in our hybrid ranking, each state in our attack graph model is a goal with merit. We sum up all the expenditure of exploits to every destination state node instead of the terminal ones. That means all the assets and privileges in the evaluated network system can be defined as goals and have their value for attackers.

Many techniques for quantitative security analysis and attacking strategy modeling are presented. Khazan et al. present simulation of a network system for quantitative security evaluation based on discrete-event simulation by SimEvents [10]. In [10], the system in normal state is simulated, an attacker is modeled as a client by means of zombies attacks to the system, then the availability of system begins to decrease, finally the system cannot respond to the requests. Peng Liu [19] renders a game theoretic approach to infer attacker intent, objectives and strategies and thus the attacker objectives are practically modeled. In [20], authors deal with various issues related to quantifying the security attributes of an intrusion tolerant system, which facilitates the use of stochastic modeling techniques to capture the attacker behavior as well as the system's response to a security intrusion. However, we use a numerical definition of crucial rate for each *Vul_Inst* to quantify the security rate of whole network scenario, but not only the attacking paths to measure computer security level. The previous framework was proposed in [21]. In a generated graph, the most likely attack sequences are computed. The shortest path cannot be a rounded criterion for the complete evaluation of scanned vulnerabilities, since the number of vulnerabilities in the single shortest path is limited.

7. Conclusion

In this paper, a hybrid ranking approach is proposed to estimate the influential level of vulnerability instances in a dynamic way. An adapted *CVSS* rating has been implemented to rate for the *VE* and *VI* of each vulnerability respectively and aggregating them into the crucial rate of *Vul_Inst* for high-level evaluating. *MEx_ini*, the evaluated and quantified security rate of corresponding dwindled attacking scenario, greatly provides a precise measurement on the influential level of vulnerability instances under a view of the dynamic and global network scenario, which renders better patching priority. The main contribution of our work is to render a hybrid security metric for quantifying and ranking the security level of particular network system under particular time and situation. Combining the graph generation and analysis tools with security metrics, it is effective to weigh the pros and cons of decisions on choosing patching orders of these vulnerabilities, and take more economic and efficient hardening strategy for given network scenarios. The experimental results also demonstrate the applicability and accuracy of our hybrid ranking approach.

In future, we will focus on how to sharpen both the low-level rating and high-level evaluating in order for large-scale network estimation. We will emphasize two crucial parts: Firstly, how to properly involve the stakeholders' specific considerations or experts' experience into a more accurate aggregation of the crucial rate; Secondly, how to provide a flexible framework to let network administrators predefine the suitable attacking strategies for simulation and the precise models for quantification.

Acknowledgments

This work is supported by National Natural Science Foundation of China under Grant 60803114 and National 973 Basic Research Program of China under Grant 2007CB310900.

References

[1] Common Vulnerability Scoring System-SIG. Available at: http://www.first.org/cvss/ (accessed May 2008).
[2] MITRE, CVE—Common vulnerabilities and exposures. http://cve.mitre.org/ (last retrieved April 2007).
[3] National Vulnerabilities Database. http://nvd.nist.gov/.
[4] Jason Li, Levy Renato, Peng Liu, An intelligent cyber security analysis in enterprise networks, in: Proceedings of 2007 AAAI RIDIS Workshop, Arlington, Virginia, November 2007.
[5] S. Jajodia, S. Noel, B. O'Berry, Topological analysis of network attack vulnerability, in: V. Kumar, J. Srivastava, A. Lazarevic (Eds.), Managing Cyber Threats: Issues, Approaches and Challenges, Springer, 2005.
[6] P. Mell, K. Scarfone, S. Romanosky, A complete guide to the common vulnerability scoring system version 2.0, Published by Forum of Incident Response and Security Teams, FIRST, June 2007. http://www.first.org/cvss/cvss-guide.pdf.
[7] J. Dawkins, J. Hale, A systematic approach to multi-stage network attack analyses, in: Proceedings of the Second IEEE International Information Assurance Workshop, Charlotte, North Carolina, April 2004, pp. 48–56.
[8] K. Ingols, R. Lippmann, K. piwowarski, Practical attack graph generation for network defense, in: Proceedings of the 22nd Annual Computer Security Applications Conference, December, 2006.
[9] E.A. Feinberg, A. Shwartz, Handbook of Markov Decision Process, Kluwer Academic Publishers, Boston, 2001.

[10] G. Khazan, M.A. Azgomi, A distributed attack simulation for quantitative security evaluation using SimEvents, in: Proceedings of IEEE/ACS International Conference on 2009 Computer Systems and Applications, AICCSA 2009, Rabat, Morocco, May 2009, pp. 382–385.
[11] Nirnay Ghosh, S.K. Ghosh, An intelligent technique for generating minimal attack graph, in: Proceedings of International Conference on Automated Planning and Scheduling, Workshop on Intelligent Security, Thessaloniki, Greece, September 2009.
[12] Julien Aussibal, Laurent Gallon, A new distributed IDS based on CVSS framework, in: Proceedings of 2008 IEEE International Conference on Signal Image Technology and Internet Based Systems, Bali, December 2008, pp. 701–707.
[13] S. Houmb, V. Franqueira, E. Engum, Estimating impact and frequency of risks to safety and mission critical systems using CVSS, in: Proceedings of the ISSRE 2008 Conference: 1st Workshop on Dependable Software Modelering 2008, Bangalore–Mysore, India, November 2008.
[14] S. Houmb, N.L. Virginia, Franqueira estimating ToE risk level using CVSS, in: Proceedings of 2009 International Conference on Availability, Reliability and Security, Fukuoka, Japan, March 2009, pp. 718–725.
[15] M. Mohammadi, G.B. Gharehpetian, Application of core vector machines for on-line voltage security assessment using a decisiontree-based feature selection algorithm, IET Generation, Transmission & Distribution 3 (8) (2009) 701–712.
[16] M. Al-Kuwaiti, N. Kyriakopoulos, S. Hussein, A comparative analysis of network dependability, fault-tolerance, reliability, security, and survivability, IEEE Communications Surveys & Tutorials 11 (2) (2009) 106–124.
[17] P. Ammann, D. Wijesekera, S. Kaushik, Scalable, graph-based network vulnerability analyses, in: Proceedings of the 9th ACM Conference on Computer and Communications Security, CCS, Washington, USA, Novermber 2002, pp. 217–224.
[18] S. Jha, O. Sheyner, J.M. Wing, Two formal analyses of attack graphs, in: Proceedings of the 15th IEEE Computer Security Foundations Workshop, Nova Scotia, Canada, June 2002, pp. 49–63.
[19] Peng Liu, Wanyu Zang, Meng Yu, Incentive-based modeling and inference of attacker intent, objectives, and strategies, ACM Transactions on Information and System Security 8 (1) (2005) 78–118.
[20] B.B. Madan, K.G. Popstojanova, K. Vaidyanathan, K.S. Trivedi, A method for modeling and quantifying the security attributes of intrusion tolerant systems, Performance Evaluation 56 (3) (2004) 167–186.
[21] H.L. Vu, K.K. Khaw, T. Chen, Kuo. Fei-Ching, A new approach for network vulnerability analysis, in: Proceedings of 33rd IEEE Conference on Local Computer Networks, Montreal, Canada, October 2008, pp. 200–206.