Compositional Approach to Quantify the Vulnerability of Computer Systems

HOSSEIN HOMAEI AND HAMID REZA SHAHRIARI*

Department of Computer Engineering and Information Technology, Amirkabir University of Technology, Tehran, Iran

*Corresponding author: shahriari@aut.ac.ir

Although analyzing complex systems could be a complicated process, current approaches to quantify system security or vulnerability usually consider the whole system as a single component. In this paper, we propose a new compositional method to evaluate the vulnerability measure of complex systems. By the word composition we mean that the vulnerability measure of a complex system can be computed using pre-calculated vulnerability measures of its components. We define compatible systems to demonstrate which components could combine. Moreover, choice, sequential, parallel and synchronized parallel composition methods are defined and the measurement of the vulnerability in each case is presented. Our method uses a state machine to model the system. The model considers unauthorized states and attacker capabilities. Furthermore, both the probability of attack and delay time to reach the target state are used to quantify vulnerability. The proposed approach would be useful to analyze complex systems which may have complicated models. This approach reduces the state space and complexity of computation. On the other hand, if a component is replaced by another one, the vulnerability measures of other components do not change. Thus, these quantities are reused in new computation. Therefore, the calculation of the vulnerability measure for a new system is simplified.

Keywords: compositional evaluation; security quantification; vulnerability analysis

Received 20 February 2011; revised 28 April 2011
Handling editor: Mostafa Bassiouni

1. INTRODUCTION

To compare the security of two same systems, or assess the security improvement of a system, we need security evaluation methods. If we have an accurate estimation of system security, we can evaluate the effectiveness of security countermeasures on the system and compare the system’s security.

Although some methods have been proposed to quantify vulnerability, these methods have some drawbacks. For example, some methods such as [1, 2] just model one kind of attack or attacker, and cannot model all kinds of attacks such as internal attacks. Another drawback to some existing models, for example [3], is that they are based on some assumptions such as exponential distribution for transition probability which has not been proved yet [4].

The vulnerability quantity of a system is not an absolute value but rather depends on the threatening adversary. For example, a system might be secure against an amateur attacker, but vulnerable against a skilled one. On the other hand, it seems that the vulnerability measure depends on the attack duration. The more time it takes to complete an attack, the more the system is secure. Accordingly, any vulnerability measurement approach should consider the attack duration.

Another important issue is that quantifying the vulnerability of large and complicated systems could be demanding. For example, if we use a state machine to model a complex system, it will be very large and complicated; especially if we want to model low level system details. Thus, it could be difficult to model the system and analyze it.

In this paper, we propose a general model to quantify the vulnerability of computer systems. We believe that the model could surmount above-mentioned drawbacks. Providing a composable approach to analyze system vulnerabilities and considering attacker capabilities are two major contributions of the model.

The method can compute the vulnerability quantity of systems using the vulnerability measures of constitutive
components. In other words, to quantify the vulnerability of a large system, we can compute the vulnerability measure of its components and then compose these quantities to measure the vulnerability of the whole system.

On the other hand, it can model computer systems against diverse attacks and attackers considering pertinent unauthorized states and appropriate attacker capabilities. Moreover, the model uses the delay time parameter in the evaluation process and thus it can discriminate between two systems having the same probability to reach unauthorized states. Furthermore, there is no assumption about transition probability such as exponential distribution.

The rest of this paper is organized as follows. Section 2 surveys related work on vulnerability quantification. Section 3 describes a formal model of computer systems. The vulnerability measurement process is presented in Section 4. Section 5 explains the compositional approach to quantify security based on the model that is represented in Section 3. Section 6 specifies a practical case study to clarify the vulnerability measurement process. Section 7 discusses some special issues of the paper and compares existing compositional evaluation methods with the method presented in this paper. Finally, Section 8 concludes this paper and discusses our future work.

2. RELATED WORK

Littlewood et al. [5] have discussed similarities between reliability and security. They have suggested using reliability and dependability measurement methods [6, 7] to predict the amount of security. The major intention of [5] is to invite discussion about the feasibility of this approach. Thus, some open questions that have been identified need to be answered.

Jonsson and Olsson [8] believe that the attacking process can be split into three phases. They have indicated that the periods of time between breaches in the standard attack phase are exponentially distributed. Thus, they have suggested using traditional reliability methods to model security. However, the results of this experiment have been achieved in a special environment and could not be extended to all situations.

Kaâniche et al. [4] have represented empirical analyses based on the data collected from honeypot platforms deployed on the Internet. They found out that the mixture of a Pareto and an exponential distribution is the best fit for the observed times between attacks. Thus, the traditional assumption considered in hardware reliability evaluation does not seem to be satisfactory for security evaluation.

Madan et al. [9] have considered security as a Quality of Service attribute. They have used stochastic modeling techniques to quantify security attributes of intrusion-tolerant software systems. They have suggested the Mean time (or effort) to a security failure metric to quantify security. The method could be used as a framework and needs more development.

Sallhammar et al. [10] have proposed a new approach to integrate security and dependability evaluation. The approach uses a stochastic game. The state machine used in this method is unable to model system details. Thus, quantifying ‘accumulated failure intensity’ would be difficult.

Manadhat and Wing [11] have represented a metric to determine whether one version of a software is relatively more secure than another version of it with respect to the system’s attack surface. This method could not compare different systems which have different resources.

Singh et al. [12] have divided the security quantification methods into penetration testing and analysis of formal models of the system. They have proposed a new approach to quantify security by performing repeated penetration testing of a detailed system model. This approach will utilize the accuracy of a formal model and the fastness of penetration testing. But the security metric, number of attack paths, is not accurate; because the time and probability of attack paths are not considered.

Pamula et al. [13] have presented a quantitative metric to assess the amount of network security based on attack graph analysis. The metric measures the security strength of a network in terms of the strength of the weakest adversary who can successfully penetrate the network. But unfortunately the capabilities of the weakest adversaries are not always comparable for all systems.

Frigault et al. [14, 15] have used attack graphs as a special Bayesian Network to measure the security of a system. However, they have not proposed a new metric to quantify the system security. Instead, they have used the Common Vulnerability Scoring System [16] to measure the exploit probability of a vulnerability based on dependency relations between vulnerabilities.

Mehta et al. [17] have represented two algorithms to rank the states of the attack graph. The first algorithm is similar to the PageRank algorithm used by Google. The second algorithm ranks individual states based on the reachability probability of an attacker in a random simulation. They have used the total rank of all error states as a system security measurement. But the method does not consider overlaps between paths.

Pham et al. [18] have constructed a graph based on open ports of network hosts and also the access rights between hosts. They have used a metric called attackability metric to compute the assurance level of the system. The metric uses just the number of access points on each host to measure the assurance level. Thus, it cannot be used as a security or vulnerability metric.

Other methods that use attack graph analysis for security quantification could be divided into two categories: reliability methods and minimum cost hardening methods. Reliability methods such as the method proposed in [19] use the probability of a successful attack as a security metric. On the other hand, minimum cost hardening methods, for instance [20, 21], use the cost of obviation of attack paths as a security metric. The reliability methods do not consider the cost of the attack path. The main challenge of minimum cost hardening methods,
on the other hand, is that they use just the cost of vulnerability obviation to quantify security; nevertheless, we could increase the security of the system by other methods like changing the configuration.

Wang et al. [2] have proposed a two-step framework to measure various aspects of network security. In the first step, the individual security components are evaluated. The second stage focuses on the composition of individual measures. It could be used as a general framework to quantify security.

Shahriari [22] has proposed a new method for vulnerability analysis using a state machine. In this method, the vulnerability of a state, $S$, is defined as the probability of reaching an unauthorized state, starting from $S$. The system vulnerability is defined as the vulnerability of the initial state. Thus, the vulnerability of any system that does not have any absorbing state will be equal to 1. On the other hand, it seems that if it takes different periods of times to reach unauthorized states in two different systems, the vulnerability measures of these systems should be different. However, this method would not differ between the vulnerability measures of these systems because it does not consider the delay time to reach unauthorized states.

### 3. Modeling Process

#### 3.1. System model

In this subsection, we model computer systems. The model consists of a set of states connected to each other by some transitions. Each transition is related to an event which causes the system to travel between states. Events might be triggered by the system, another system or an attacker. There is a specific state variable.

An intruder can trigger some events that lead the system to an unauthorized state where at least one of the security policies is violated. The following definition specifies a formal description of the system.

**Definition 3.1.** A computer system, say CS, is a 7-tuple $CS = (Q, q_0, E, \Delta, \rho, \delta, Q_{UA})$, where $Q$ is the finite set of all states, $q_0 \in Q$ is the initial state, $E$ is the finite set of all possible events, $\Delta \subseteq Q \times E \times Q$ is the transition relation, $\rho : \Delta \rightarrow (0, 1]$ is the transition probability function, $\delta : E \rightarrow [0, \infty)$ is the delay function, and $Q_{UA} \subseteq Q$ is the set of all unauthorized states.

Each part of the model is described as follows:

(i) $Q$ is the finite set of all system states. Each state is specified by system variables which can be changed when a system action occurs. These variables are called state variables. For example, user privilege is a security state variable. $Q$ should represent all possible states in the system. State variables can be chosen by the evaluator according to the modeling purpose.

(ii) $q_0 \in Q$ is the initial state in which the system starts working. If there is more than one initial state, we can use a virtual state as the initial state and connect it to the real initial states by virtual transitions.

(iii) $E$ is the set of events. The set is partitioned into three finite subsets $E^{in}$, $E^{int}$ and $E^{out}$, which show input, internal and output events, respectively.

(a) An input event occurs in the environment and affects the system. The system has no control on the generation of input events. The probability and delay time of the input event differ in various environments. But all possible transitions are known during system modeling because in this paper we model just deterministic systems.

Input events are partitioned into two sets: $E^{inn}$ and $E^{in}$, which represent normal and attacker input events, respectively. An attacker input event occurs when an attacker enforces the system to do an action which the system does not expect to receive normally. Other input events are called normal input events.

(b) An internal event is produced by the system and has no effect on the environment. Internal events are hidden from the external view.

(c) An output event is generated by the system and has an effect on the environment. System services are specified by output events.

The union of internal and output events are called local events and are denoted by $E^{loc}$ because they are produced by the system locally.

(iv) $\Delta \subseteq Q \times E \times Q$ is the transition relation. The relation is not input-enabled because in practice not all states can receive all input events. In contrast, some specific input events can occur in some states. However, we can provide an input-enabled property considering that each input event that has no effect on the system will not change the state.

(v) $\rho : \Delta \rightarrow (0, 1]$ is the transition probability function which satisfies the following properties:

1. $\rho(q_0, e, r) > 0$ iff $(q_0, e, r) \in \Delta$;
2. $\forall q, r \in Q \forall e \in E^{in} [(q, e, r) \in \Delta \rightarrow \rho(q, e, r) = 1]$;
3. $\forall q \in Q \sum_{e \in E^{inn}} \sum_{r \in E^{loc}} \rho(q, e, r) = 1$.

In contrast to the Markov model, it is not necessary that the exit probability be equal to 1 for all states because the system generates just the local events and we cannot predict input event probabilities. In other words, the transition probability function does not represent the probability of the occurrence of input events; instead it shows the transition probability. As we mentioned before, the model is deterministic. Thus, the transition probability of any input event is equal to 1.

(vi) $\delta : E \rightarrow [0, \infty)$ is the delay function. It takes $\delta(e)$ for the system to carry out event $e$. We see that $\delta(e)$ is
A computer system might be secure against an attacker but the final goal of this kind of modeling is not only to determine the state graph or the scenario graph. To specify security policies in addition to the system properties, an unauthorized state is the state in which at least one of the security policies is violated. For example, if an attacker gains root access to the database server, the security policy is violated and thus the system will reach an unauthorized state.

The model introduced in Definition 3.1 is a formal model that specifies security policies in addition to the system properties. The final goal of this kind of modeling is not only to determine a specific system but also to provide a general framework to model various kinds of systems. For example, if we consider just attacker input events, the model is well reduced to the attack graph or the scenario graph.

A path from \( q_1 \) to \( q_2 \) is a sequence of transitions from state \( q_1 \) to state \( q_2 \) and denoted by \( \text{path}_{q_1q_2} \). If \( \text{path}_{q_1q_2} \) contains a set of normal input events, \( E^m \), we call it a conditional path and denote it by \( (\text{path}_{q_1q_2}, E^m) \).

There could be either no path or many paths between two states in the model. Obviously, if there is a path from the initial state to an unauthorized state, the system have a vulnerability which can be exploited by attackers. But note that different attackers can reach the unauthorized state making different efforts. Thus, we need to model the attacker and then measure the vulnerability of the system against a particular attacker. In the following subsection, we define the attacker model and then in the subsequent subsection the attacker model is combined with the system model.

### 3.2. Attacker model

A computer system might be secure against an attacker but vulnerable against another one because different kinds of attackers have various capabilities and skills and they can do diverse actions. Moreover, an intruder who has more capabilities is able to accomplish an attack faster. For example, Blaze et al. [23] have compared the needed time to find a 56-bit key of a cryptographic algorithm by different types of attackers. A single hacker needs more time to attack a system than an intelligence agency does. Thus, we need an attacker model to evaluate the security of a computer system. The strongest attacker is modeled if a specific intruder is not identified.

**Definition 3.2.** An intruder, say \( I \), is a triple \( I = (E, SL, ED) \), where \( E \) is the set of all possible attacker events, \( SL : E \rightarrow (0, 1] \) is the skill level function and \( ED : E \rightarrow (0, \infty] \) is the event delay function.

A higher skill level of the intruder means a greater probability of having exploit codes and successful attacks. Thus, the attacker skill level represents the probability of executing attacker actions. On the other hand, the required time to accomplish an attack step is represented by the event delay function.

### 3.3. The model of an under-attack system

In this subsection, we specify the composition of attacker and system models. The combination of these two models is called the under-attack system and defined as follows.

**Definition 3.3.** The composition of computer system \( CS = (Q_{cs}, q_{cs}, E_{cs}, \Delta_{cs}, \rho_{cs}, \delta_{cs}, Q_{UA_{cs}}) \) and intruder \( I = (E_{att}, SL, ED) \) is called the under-attack system and denoted by \( CS \leftarrow I \). The under-attack system is 7-tuple \( CS \leftarrow I = (Q, q', E, \Delta, \rho, \delta, Q_{UA}) \) where

- \( Q = Q_{cs}; \)
- \( q' = q_{cs}; \)
- \( E = E_{cs}; \)
- \( \Delta = \Delta_{cs}; \)
- \( \forall q, r \in Q \forall e \in E \{ \rho(q, e, r) = SL(e), \forall q, r \in Q \forall e \in E \{ \rho(q, e, r) = \rho_{cs}(q, e, r) \}; \)
- \( \forall e \in E \{ \delta(e) = ED(e), \forall e \in E \{ \delta(e) = \delta_{cs}(e) \} \)
- \( Q_{UA} = Q_{UA_{cs}}. \)

An under-attack system represents real quantities for the probability and delay of local and attacker events, but we do not have any information about the delay and probability of normal input events. Thus, we use default quantities, 1 for the probability and zero for the delay time, for normal input events in the modeling phase.

### 4. Vulnerability Measurement

In this section, we propose a metric to quantify the vulnerability. However first, we define some required parameters.

**Definition 4.1.** The cost of transition \( (q, e, r) \) is denoted by \( \text{cost}(q, e, r) \) and defined as the event delay per transition probability ratio. In other words, \( \text{cost}(q, e, r) = \delta(e)/\rho(q, e, r) \).

In other words, the cost of each attack step is related to the delay time and the reverse of the transition probability. For instance, if the delay time of event \( e \) is equal to 1 h and the probability of transition \( (q, e, r) \) is equal to 0.5, the cost of transition \( (q, e, r) \) will be equal to 2. As the probability of transition is equal to 0.5, in every two attempts at doing the transition \( (q, e, r) \) one of them is guaranteed statistically. Thus, the delay time of transition \( (q, e, r) \) approaches to 2 instead of 1 h.

When we talk about the cost of normal input transitions, it suffices to know the delay time and the probability of event occurrence. Because the transition probability is equal to 1 and we can specify the cost of these transitions using the probability...
and delay of events. The cost of the normal input event, $e$, is denoted by \( \text{DPR}(e) \) and defined as \( \text{DPR}(e) = \delta(e)/P(e) \), where \( P(e) \) is the probability of occurring event \( e \).

Furthermore, we define the cost of path as follows.

**Definition 4.2.** The cost of conditional path (\( \text{Path}_{q_{i}q_{j}}(E^\text{innor}_{\text{path}_{q_{i}q_{j}}}) \)) in the under-attack system \( CS \leftrightarrow I \) is denoted by \( \text{cost}(\text{Path}_{q_{i}q_{j}}(E^\text{innor}_{\text{path}_{q_{i}q_{j}}})) \) and calculated by the following equation:

\[
\text{cost}(\text{Path}_{q_{i}q_{j}}(E^\text{innor}_{\text{path}_{q_{i}q_{j}}})) = \sum_{\forall(q,e,r)\in\text{Path}_{q_{i}q_{j}}} \text{cost}(q, e, r). \quad (1)
\]

If the parameters of normal input events \( E^\text{innor}_{\text{path}_{q_{i}q_{j}}} = \{e_{1}^\text{inn}, e_{2}^\text{inn}, \ldots, e_{n}^\text{inn}\} \) are available, the cost of \( \text{Path}_{q_{i}q_{j}} \) will be calculated as follows:

\[
\text{cost}(\text{Path}_{q_{i}q_{j}}) = \text{cost}(\text{Path}_{q_{i}q_{j}}(E^\text{innor}_{\text{path}_{q_{i}q_{j}}})) + \sum_{i=1}^{k} \text{DPR}(e_{i}^\text{inn}). \quad (2)
\]

The most interesting path from an intruder point of view is the path that starts from the initial state and terminates at an unauthorized state and takes the least cost, because the attacker wants to reach unauthorized states using minimum effort.

The shortest conditional path from the initial state, \( q'_{i} \), to the state \( q \) under the condition of occurrence of the set of normal input events, \( E^\text{innor}_{\text{pre}} \), denoted by \( \text{Path}_{q'_{i}q}(E^\text{innor}_{\text{pre}}) \), is the minimum cost path within all possible conditional paths like \( \text{Path}_{q'_{i}q}(E^\text{innor}_{\text{pre}}) \). In other words

\[
\text{cost}(\text{Path}_{q'_{i}q}(E^\text{innor}_{\text{pre}})) = \text{Min}_{\forall(q,e,r)\in\text{Path}_{q'_{i}q}(E^\text{innor}_{\text{pre}})} \times \{\text{cost}(\text{Path}_{q'_{i}q}(E^\text{innor}_{\text{pre}}))\}. \quad (3)
\]

Using this concept, we can define the vulnerability measure of an unauthorized state as follows.

**Definition 4.3.** The conditional vulnerability measure of an unauthorized state \( q \in Q_{\text{UA}} \) under the condition of occurrence of the set of normal input events \( E^\text{innor}_{\text{pre}} \) in the under-attack system \( CS \leftrightarrow I \) is denoted by \( \nu(q|E^\text{innor}_{\text{pre}}) \) and defined as the reverse of the shortest path cost. In other words,

\[
\nu(q|E^\text{innor}_{\text{pre}}) = \frac{1}{\text{cost}(\text{Path}_{q'_{i}q}(E^\text{innor}_{\text{pre}}))}. \quad (4)
\]

If \( E^\text{innor}_{\text{pre}} \) is null, \( \nu(q|\emptyset) \) is called unconditional vulnerability measure of state \( q \).

The less it takes cost to reach an unauthorized state, the more the state is vulnerable. Thus, the definition matches the concept of vulnerability practically.

Any possible path might be chosen by intruders to compromise the system. We cannot predict which path will be chosen by the attacker. But we know that the attacker tries to choose the convenient path which takes less cost. Thus, to measure the vulnerability of the system, we use the shortest path concept. In other words, we find an upper bound for the quantity of vulnerability. The interest in using the shortest path arises in the attack concept. The hard intruder of today becomes the most probable intruder of tomorrow. Thus, the system will be exploited by a stronger attacker in the future [24].

After quantifying the vulnerability of each state, we can evaluate the vulnerability of the whole system. As we want to find an upper bound of system vulnerability, we use the following definition.

**Definition 4.4.** The conditional vulnerability measure of the under-attack system \( CS \leftrightarrow I \) under the condition of occurrence of a set of normal input events \( E^\text{innor}_{\text{pre}} \) is denoted by \( \nu(CS \leftrightarrow I|E^\text{innor}_{\text{pre}}) \) and calculated by the following equation:

\[
\nu(CS \leftrightarrow I|E^\text{innor}_{\text{pre}}) = \max_{\forall q \in Q_{\text{UA}}} \{\nu(q|E^\text{innor}_{\text{pre}})\}. \quad (5)
\]

To determine the unconditional vulnerability measure, the actual parameters of normal input events should be available. Thus, the unconditional vulnerability could be measured by the following equation:

\[
\nu(CS \leftrightarrow I) = \max_{1 \leq i \leq n} \left\{ \frac{1}{\nu(CS \leftrightarrow I|E^\text{innor}_{\text{pre}_{i}})} + \sum_{\forall q \in E^\text{innor}_{\text{pre}_{i}}} \text{DPR}(e) \right\}, \quad (6)
\]

where \( n \) is the number of conditional vulnerability measures and \( \nu(CS \leftrightarrow I|E^\text{innor}_{\text{pre}_{i}}) \) is the \( i \)th conditional vulnerability measure for the system.

## 5. Compositional Evaluation

In previous sections, we proposed a model of a computer system and a method to quantify its vulnerability. But it seems difficult to measure the vulnerability of complex systems because the state space of complex systems are very large and complicated.

In this section, we use the system model introduced in Definition 3.1. However, some compositional methods to measure the vulnerability of complex systems are applied.

### 5.1. Compatible Systems

If we want to quantify vulnerability in a compositional manner, we should illustrate which systems can combine. Thus, we define compatible unauthorized states and also compatible systems.

**Definition 5.1.** Suppose that two systems \( CS_{1} \) and \( CS_{2} \) have been modeled as \( CS_{1} = (Q_{1}, q'_{1}, E_{1}, \Delta_{1}, \rho_{1}, \delta_{1}, Q_{\text{UA}_{1}}) \) and \( CS_{2} = (Q_{2}, q'_{2}, E_{2}, \Delta_{2}, \rho_{2}, \delta_{2}, Q_{\text{UA}_{2}}) \). Each unauthorized
state $q \in Q_{UA_1}$ is compatible with system CS$_2$, if and only if $(q \notin Q_2)$ or $(q \in Q_2 \rightarrow q \in Q_{UA_1})$.

In other words, an unauthorized state should be an unauthorized state in both systems and security policies should be compatible.

**Definition 5.2.** A finite set of under-attack systems \{($CS_i \rightarrow I$) : $i \in I$\} is compatible if and only if each unauthorized state in each system is compatible with other systems and, for all $i, j \in I (i \neq j)$, the following equations are satisfied:

1. $E_i^{out} \cap E_j^{out} = \emptyset$;
2. $E_i^{int} \cap E_j^{int} = \emptyset$.

It is obvious that the compatibility is defined just for systems that are under the same attacker model because compositional analysis is possible if all components are analyzed against the same intruder. Compatible systems do not have the same outputs because each system has its own tasks and services. Furthermore, each system has its own internal events because an external observer cannot view these events.

### 5.2. Compositional operators

In this subsection, we define some operators to combine components and construct composite systems. We can determine four operators as follows:

(i) Choice: just one of the components is chosen and executed. The choice operator is denoted by ‘+’.
(ii) Sequential: components are executed in order sequentially. We use a dot sign to demonstrate this kind of composition.
(iii) Parallel: each component is executed independently. $C_1||C_2$ means that components $C_1$ and $C_2$ are executed in a parallel manner.
(iv) Synchronous parallel: components are executed simultaneously until they need to execute a common event. One component might stop working until the common event happens.

Considering Definition 5.2, the common event might be an output of one component and an input event of another one or it might be a common input event for both components. Thus, we have two kinds of synchronous composition: output–input and input–input synchronization. Synchronization on the input event has no practical meaning; because if an input event is available, it can have an effect on the system. In other words, the system does not stop working until another component proceeds to the point at which it can receive the common input. Thus, just output–input synchronization is considered in this article. If $e$ is an output event of $C_1$ and an input event of $C_2$, the synchronous composition of components $C_1$ and $C_2$ on event $e$ is denoted by $C_1 \, \Box_e \, C_2$.

Note that in parallel composition, all components can be executed, but when we use the choice operator, just one of the components is selected and executed.

The formal definitions of these operators are omitted because of lack of space.

### 5.3. Service parameters

Each output event of the system is a service. Before talking about the compositional methods, we should define some parameters about the system services.

For any output event like $e \in E^{out}$, the set of all states where event $e$ occurs is called the exit set of $e$ and denoted by $Q_e$. To give a service like $e$, the system should reach one of the exit states $q \in Q_e$ and then execute the output event. There could be several paths to reach state $q$. Thus, we should determine which path is chosen by the system. Using the shortest path could be proportional to the vulnerability measure definition. The service path of $e$, denoted by $path_e^s$, is the shortest path from the initial state to any state $q^e$, where $q^e$ is the next state of executing $e$ in the state $q \in Q_e$.

Thus, the service time is defined as follows.

**Definition 5.3.** The service time of output event $e \in E^{out}$, denoted by $ST_e$, is defined as the cost of service path $path_e^s$.

Another parameter that we need in compositional analysis is the completion time.

**Definition 5.4.** The completion time of a system denoted by $ST$ is the cost of shortest path from the initial state to the final state where the system terminates. In other words $ST = cost(path_{\min}^{\text{Min}}_q)$.

If there are more than one final state in the system, the shortest path from the initial state to any final state will be calculated and the path which has the lowest cost will be used to quantify the completion time.

### 5.4. Proposed compositional method

In this subsection, we demonstrate four theorems to measure the vulnerability of complex systems. In each theorem, one of the compositional operators is studied.

**Theorem 5.1.** The vulnerability measure of the system $C = C_1 + C_2 + C_3 + \cdots + C_n$ against an intruder $I$ is computed by the following equation:

$$\nu(C \leftarrow I) = \max_{1 \leq i \leq n} \{\nu(C_i \leftarrow I)\}. \quad (7)$$

**Proof.** If components are combined by a choice operator, the composed system might behave like each of them. In the worst case, the component that has the highest vulnerability is chosen by the attacker. Thus, the vulnerability measure of the system will be equal to the vulnerability measure of that component...
because we use the shortest vulnerable path to quantify the vulnerability of the system.

**Theorem 5.2.** The vulnerability measure of composite system $C = C_1 \cdot C_2 \cdot C_3 \cdot \ldots \cdot C_n$ against an intruder $I$ is calculated by the following equation:

$$v(C \leftrightarrow I) = \text{MAX} \left\{ v(C_1 \leftrightarrow I), \text{MAX}_{2 \leq k \leq n} \frac{1}{\sum_{i=1}^{k-1} \text{ST}_{C_i} + 1/v(C_k \leftrightarrow I)} \right\}. \quad (8)$$

**Proof.** Each unauthorized state in the composite system should be an unauthorized state in one of its components. Assume that there are two components $C_1$ and $C_2$ that combine and construct $C = C_1 \cdot C_2$. If an unauthorized state of component $C_1$ causes the unauthorized state of $C$, the vulnerability measure of the composite system will be equal to the vulnerability measure of component $C_1$ because the shortest path starts from the initial state of $C_1$ and reaches the unauthorized state of the same component.

But the shortest path to reach the unauthorized state which is caused by an unauthorized state in $C_2$ starts from the initial state in $C_1$, reaches the final state of $C_1$ by the shortest path and continues to the unauthorized state in $C_2$ through the shortest vulnerable path. Thus, the cost of this path is the sum of the completion time of $C_1$ and the cost of the vulnerable path of $C_2$, which are equal to $\text{ST}_{C_1}$ and $1/v(C_2 \leftrightarrow I)$, respectively. Therefore, the vulnerability measure of the system is equal to the reverse of this cost.

If there is a shorter path than this path, it should be a shorter path for completion of $C_1$ or a shorter path to reach the unauthorized state in $C_2$, which are in direct contradiction to the definitions of completion time and vulnerability measure.

We can extend the proof to more components and achieve Equation (2) using inductive reasoning. Assume that Equation (2) is true for $n$ components. We want to show that this equation is true for $n + 1$ components. If there are $n + 1$ components in the system, the vulnerability of the system $C = C_1 \cdot C_2 \cdot C_3 \cdot \ldots \cdot C_n \cdot C_{n+1}$ will be equal to the maximum of inverse cost of the shortest path to reach the unauthorized state in $C_1$, $C_2$, $C_n$, or $C_{n+1}$. This quantity for components $C_1$, $C_2$, $C_n$ and $C_{n+1}$ is calculated by Equation (2) considering the induction assumption. If an unauthorized state occurs in component $C_{n+1}$, the vulnerability of the system will be computed as follows:

$$v(C^{n+1} \leftrightarrow I) = \frac{1}{\sum_{i=1}^{n} \text{ST}_{C_i} + 1/v(C_{n+1} \leftrightarrow I)},$$

where $v(C^{n+1} \leftrightarrow I)$ is the vulnerability measure of system $C$ if the unauthorized state occurs in component $C_{n+1}$.

In other words, to reach the unauthorized state in $C_{n+1}$, all components from $C_1$ to $C_n$ should complete their work and then the vulnerable path of component $n + 1$ should be traversed. Thus, the vulnerability of system $C$ is calculated as follows:

$$v(C \leftrightarrow I) = \text{MAX} \left\{ v(C_1 \leftrightarrow I), \text{MAX}_{2 \leq k \leq n+1} \frac{1}{\sum_{i=1}^{k-1} \text{ST}_{C_i} + 1/v(C_k \leftrightarrow I)} \right\}. \quad (8)$$

We can simplify this equation and achieve the expected one.

$$v(C \leftrightarrow I) = \text{MAX} \left\{ v(C_1 \leftrightarrow I), \text{MAX}_{2 \leq k \leq n+1} \frac{1}{\sum_{i=1}^{k-1} \text{ST}_{C_i} + 1/v(C_k \leftrightarrow I)} \right\}. \quad (8)$$

Thus, Equation (8) is true for $n + 1$ components as it is true for $n$ components.

**Example 1.** Assume that two systems $C_1$ and $C_2$ are modeled as shown in Fig. 1. The unauthorized states are depicted by gray color. The label of each transition specifies the transition cost. State $S3$ is the final state of components $C_1$.

Sequential composition of these two components is shown in Fig. 2. There are two unauthorized paths in the composite system. The cost of the path which reaches the unauthorized state $(S4, S1')$ is $X + Z$ and the cost of the second path which reaches state $(S3, S3')$ is $X + Y + Z$.
On the other hand, the cost of reaching state S4 in component C1 and state S3’ in component C2 are equal to X + Z and S, respectively. The completion time of component C1 is equal to X + Y. Thus, the vulnerability measure of each unauthorized state in the composite system is calculated as follows:

\[ \nu(S4, S1') = \frac{1}{X + Z} = \nu(C1 \leftarrow I), \]

\[ \nu(S3, S3') = \frac{1}{X + Y + S} = \frac{1}{ST_{C1} + 1/\nu(C2 \leftarrow I)}. \]

The vulnerability measure of the composite system will be equal to the maximum of these two quantities. This example clarifies the use of Equation (2).

**Theorem 5.3.** The vulnerability measure of composite system \( C = C_1||C_2||C_3|| \cdots ||C_n \) against an intruder I is calculated by the following equation:

\[ \nu(C \leftarrow I) = \text{MAX}\{\nu(C_i \leftarrow I)\}. \tag{9} \]

**Proof.** In the parallel composition, the component that reaches its unauthorized state earlier specifies the vulnerability measure of the composite system. Therefore, we can use the same reasoning which is used to prove Theorem 5.1. But note that after occurrence of an unauthorized state in one component, other unauthorized states of other components can be reached if components are combined by the parallel composition operator.

However, accessing these unauthorized states is just possible by going through more edges in the state machine. Thus, reaching later states takes up more cost than reaching the first one. Therefore, these paths will not have an effect on the vulnerability measure of the composite system. \( \square \)

**Example 2.** Figure 3 depicts two simple systems C1 and C2. Each system has one unauthorized state. The cost of reaching S2 and S4’ are X and Z + R, respectively. The vulnerability measures of the components are equal to the reverse of these quantities.

Figure 4 shows the parallel composition of these components. Obviously, each state in the composite system will be an unauthorized state if the state variables of S2 or S4’ have occurred. Thus, the vulnerability measures of unauthorized states in the composite system are calculated as follows:

\[ \nu(S2, S1') = \frac{1}{X} = \nu(C1 \leftarrow I), \]

\[ \nu(S2, S2') = \frac{1}{X + Y}, \]

\[ \nu(S2, S3') = \frac{1}{X + Z}, \]

\[ \nu(S2, S4') = \frac{1}{X + Z + R}, \]

\[ \nu(S1, S4') = \frac{1}{Z + R} = \nu(C2 \leftarrow I). \]

But X + Y, X + Z and X + Z + R are greater than X. Thus, the vulnerability measure of (S2,S2’), (S2,S3’) or (S2,S4’) is less than the vulnerability measure of state (S2, S1’). Therefore, these states have no effect on the vulnerability measure of the composite system. In other words, the maximum quantity of \( \nu(C1 \leftarrow I) \) and \( \nu(C2 \leftarrow I) \) will be equal to \( \nu(C \leftarrow I) \).

To quantify the vulnerability of a synchronous parallel composition, we should determine the cost of each vulnerable path in the second component before and after the occurrence of the input event. The following definition satisfies this requirement.

**Definition 5.5.** The cost of vulnerable conditional path \( \nu_{\nu Q}(e) \) under the condition of occurrence of normal input event e in the under-attack system CS \( \leftarrow I \) is denoted by \( \text{cost}(\nu_{\nu Q}(e)) \) and defined as \( \text{cost}(\nu_{\nu Q}(e)) = \)

![FIGURE 3. Two sample systems used for parallel composition.](image-url)

![FIGURE 4. Parallel composition.](image-url)
(\text{cost}_{q'}, \text{cost}_q) = (\text{cost}(\text{path}_{q', q}), \text{cost}(\text{path}_{q', q})), \text{where } q' \text{ is the state in which the input event } e \text{ occurs.}

Obviously, there could be more than one path in the form of \((\text{Path}_{q', q}(\{e\}))\) because the states which lie between the initial and final state could be different.

**Theorem 5.4.** The vulnerability measure of the composite system \(C = C_1 \# e \cdot C_2\) against intruder \(I\) is calculated by the following equation:

\[
v(C \leftarrow I) = \max \left\{ v(C_1 \leftarrow I), v(C_2 \leftarrow I), \frac{1}{\min_{\forall \text{path}_{q', q}(\{e\})} \{\max\{\text{ST}_{C_1}^{e}, \text{cost}_{q'}^{C_1} + \text{cost}_{q}^{C_1}\}\}} \right\},
\]

where \(\text{cost}_{q'}^{C_1}\) and \(\text{cost}_{q}^{C_2}\) are the first and second part of \(\text{cost}(\text{path}_{q', q}(\{e\}))\) in component \(C_2\), respectively, and \(q\) is an unauthorized state in \(C_2\).

**Proof.** Any unauthorized state in \(C\) might be an unauthorized state in \(C_1\) or \(C_2\). The vulnerability measure of the first case is equal to \(v(C_1 \leftarrow I)\). The unauthorized state in \(C_2\) might happen in a path that does not contain the common event \(e\), or it might happen after the occurrence of \(e\). These paths are called independent and dependent path, respectively.

If the unauthorized state happens in an independent path, the vulnerability measure of the system will be equal to \(v(C_1 \leftarrow I)\). But in the second case we should calculate the cost of the shortest path from the initial state in \(C_1\) to the unauthorized state in \(C_2\).

The cost of the path from the initial state in \(C_2\) to the state in which the common event occurs is equal to the maximum of the service time of \(e\) in \(C_1\) and the cost of reaching the state in which an input event occurs in component \(C_2\) because each component can operate simultaneously with the other one. Thus, each component can execute events independently until the second component reaches the common event. In this situation, the second component should wait for \(e\). Therefore, if \(\text{ST}_{C_1}^{e}\) is greater than \(\text{cost}_{q'}^{C_2}\), the system should wait until \(e\) happens in \(C_1\).

After occurring \(e\), component \(C_2\) continues its work and reaches the unauthorized state. In this case, the cost of the vulnerable path of the system will be equal to the sum of \(\text{cost}_{q'}^{C_2}\) and the maximum of \(\text{cost}_{q'}^{C_2}\) and \(\text{ST}_{C_1}^{e}\).

But we should repeat this process for all possible vulnerable conditional paths in \(C_2\) and find the minimum quantity; because the shortest path determines the vulnerability measure of the system. Therefore, the vulnerability measure of the system is computed by Equation (10).

**Example 3.** Figure 5 shows a parallel synchronous composite system. The numbers depict transition costs. The service time of common event \(e\) is equal to 4. There are two vulnerable conditional paths to reach the unauthorized state 5 in component \(C_2\). The cost of vulnerable paths \([1, 2, 3, 4, 5]\) and \([1, 6, 7, 8, 5]\) without considering the common event are equal to 6 and 7, respectively. Thus, the cost of the first path is less than the second one.

However, to reach the unauthorized state in the composite system, the cost of the common event should be considered. The cost of reaching state 3 in component \(C_2\) is equal to 2. But the transition \(e\) cannot occur until component \(C_1\) produces the common event. Thus, the cost of reaching state 4 will be equal to 4 instead of 2. The cost of reaching state 8 in the second vulnerable path is equal to 5 because the common event is produced before component \(C_2\) reaches state 7. Therefore, the vulnerability measure of the system is calculated as follows:

\[
v(C \leftarrow I) = \frac{1}{\min\{4 + 4, 5 + 2\}} = \frac{1}{7}.
\]

In other words, the cost of the second vulnerable path is less than the first one in the composite system. We can also calculate this quantity using Equation (4) as follows:

\[
v(C \leftarrow I) = \max\left\{ 0, \frac{1}{\min\{(\max\{4, 2\} + 4), (\max\{4, 5\} + 2)\}} \right\} = \frac{1}{7}.
\]

The unconditional vulnerability measures of the components are equal to zero because there is no unconditional path to reach unauthorized states. In other words, the system will not be vulnerable if event \(e\) does not occur.

**6. Case Study**

In this section, we represent a case study to illustrate the compositional vulnerability measurement process in a sample network. This example also demonstrates the effect of attacker capabilities on the vulnerability measure of a computer system.

Figure 6 depicts the configuration of a practical network. We want to quantify the vulnerability of this network using the compositional approach specified in Section 5.
Compositional Quantification of Computer System Vulnerability

The network consists of two subnets. The first one is DMZ, which contains a web server and some workstations. The second one is an internal subnet, which contains the database, files, application servers and workstations. There are two firewalls in the network, which separate DMZ from Internet and the internal subnet from DMZ.

The attacker is an external user who wants to get access to the internal subnet. Three security policies are specified by the administrator as follows:

1. Root access to the application servers is restricted to a specific user called admin.
2. Only admin has root access to the database.
3. Only admin has root access to the file server.

However, access to the firewalls, workstations or DMZ’s instruments does not violate the security policies. In other words, these components have no unauthorized state and only their service time will have an effect on the vulnerability measurement.

The data flow of the system is depicted in Fig. 7. Each request should pass the firewalls and DMZ before arriving at the internal network. To access the database or file server, a user should be authenticated first. If user is authenticated successfully, other requests could be sent to the application servers. These servers pass requests to the database or the file server. We have assumed that there are two authentication mechanisms in the first application server. Considering Fig. 7, we can specify following relations between components.

(i) Authentication mechanisms in AppSrv1 are connected to each other by the choice operator.
(ii) AppSrv1 is connected to the database by the synchronous parallel operator. In other words, the output of application server is used as database input. The same relationship exists between AppSrv2 and the file server.
(iii) The combination of AppSrv1 and the database is parallel with the combination of AppSrv2 and the file server.

AppSrv1 has two different authentication mechanisms. An attacker can compromise each of them and get user or root access to the application server. Suppose that the costs of gaining user and root access to the first authentication mechanism of AppSrv1 by an attacker are equal to $c_1$ and $c_1'$, respectively. These costs are equal to $c_2$ and $c_2'$ for the second authentication mechanism of AppSrv1. Gaining root access to AppSrv1 violates the security policies. Considering the choice operator...
between Auth1 and Auth2, the vulnerability of AppSrv1 is calculated as follows:
\[ \nu(\text{AppSrv1}) = \max\{\nu(\text{Auth1}), \nu(\text{Auth2})\} \]
\[ = \max\left\{ \frac{1}{\text{ST}_1 + c_1'}, \frac{1}{\text{ST}_1 + c_2'} \right\}, \]
where \(\text{ST}_1\) is the sum of the service time of FW1, DMZ and FW2.

\[ \text{ST}_1 = \text{ST}_F(\text{FW1}) + \text{ST}_F(\text{DMZ}) + \text{ST}_F(\text{FW2}). \]

Here \(\text{ST}_F\) is the service time of the forward transition from outside to the internal network.

If \(c_1'\) is less than \(c_2'\), the vulnerability measure of AppSrv1 will be equal to \(\nu(\text{Auth1})\).

AppSrv2 has only one authentication mechanism. Assume that the costs of gaining user and root access to AppSrv2 are \(c_3\) and \(c_4\), respectively. Thus, the vulnerability measure of AppSrv2 will be equal to \(1/(\text{ST}_1 + c_4')\).

The abstract model of AppSrv1 is shown in Fig. 8. If an attacker gains user access to the application server, she can get guest access to the database. If she gains root access to the application server, she will get normal user access to the database. Moreover, the intruder can compromise the application server and get user access to the database using his primitive user access to AppSrv1.

When the user receives the confirmation response, he can send other requests to the application server. The service time of sending an authentication request, receiving its response and sending a second request is calculated as follows:

\[ \text{ST}_2 = 2 \times \text{ST}_1 + \text{ST}_B(\text{FW1}) + \text{ST}_B(\text{DMZ}) + \text{ST}_B(\text{FW2}), \]

where \(\text{ST}_B\) is the backward service time from the internal network to the outside.

The service time to gain user and root access to each application server are calculated by the following equations:

\[ \text{ST}_{\text{user-access}}^{\text{AppSrv1}} = \text{ST}_2 + \min\{c_1, c_2\} = \text{ST}_2 + c_2, \]
\[ \text{ST}_{\text{root-access}}^{\text{AppSrv1}} = \text{ST}_2 + \min\{c_1', c_2'\} = \text{ST}_2 + c_1', \]
\[ \text{ST}_{\text{user-access}}^{\text{AppSrv2}} = \text{ST}_2 + c_3, \]
\[ \text{ST}_{\text{root-access}}^{\text{AppSrv2}} = \text{ST}_2 + c_3'. \]

We have supposed that \(c_1'\) is less than \(c_2'\) but \(c_1\) is greater than \(c_2\). Considering Fig. 8, the service time of the output events of AppSrv1 are computed as follows:

\[ \text{ST}_{\text{output}}^{\text{AppSrv1}} = (\text{ST}_1|I_1) + \text{ST}_{\text{user-access}}^{\text{AppSrv1}} + \text{ST}_{\text{root-access}}^{\text{AppSrv1}}, \]
\[ \text{ST}_{\text{output}}^{\text{AppSrv1}} = \min\{\text{ST}_1|I_1, \text{ST}_{\text{user-access}}^{\text{AppSrv1}}, \text{ST}_{\text{root-access}}^{\text{AppSrv1}}\}, \]
\[ = \min\{\text{cost(attack)} + \text{cost(\text{DB-user-access})}, \text{cost(\text{DB-user-access})} + \text{cost(\text{FileSrv-guest-access})}\}. \]

Assume that AppSrv2 has the same construction as AppSrv1. Thus, the service time equations of AppSrv2 are similar to the AppSrv1 equations. Note that AppSrv2 gains access to the file server instead of the database. Therefore, in these equations, we should replace cost(\text{DB-guest-access}) and cost(\text{DB-user-access}) with cost(\text{FileSrv-guest-access}) and cost(\text{FileSrv-user-access}), respectively.

An intruder can crack the database password if he gains guest access to it. He can also get user access and then use a buffer overflow attack to gain root access. The abstract model of the database is depicted in Fig. 9.

According to the model, conditional vulnerability measures of the database are calculated as follows:

\[ \nu(\text{DB}|I_1) = \max\left\{ \frac{1}{\text{cost(attack)}} , \frac{1}{\text{cost(\text{DB-user-access})} + \text{\text{cost(\text{DB-user-access})}}} \right\}, \]
\[ \nu(\text{DB}|I_2) = \frac{1}{\text{cost(\text{attack})}}. \]

Considering the synchronous parallel composition of AppSrv1 and the database and using Theorem 5.4, the following simplified equation is achieved to measure the vulnerability of the composed system:

\[ \nu(C \leftarrow I) = \max\left\{ \nu(C_1 \leftarrow I), \nu(C_2 \leftarrow I) \right\} \times \frac{1}{\min_{\text{paths(C_2|ST_{\text{output}})} + 1/\nu(C_2 \leftarrow I|\text{Input})}} \right\} \]

(11)
Using this equation, the vulnerability measure of the composed system is calculated as follows:

\[ v_1 = \text{MAX} \left\{ v(\text{AppSrv1}), v(\text{DB}), \right\} \]

\[ \times \frac{1}{\text{MIN}[\text{ST}_{\text{AppSrv1}}^{t_1} + 1/v(\text{DB}/I_1), \text{ST}_{\text{AppSrv1}}^{t_1} + 1/v(\text{DB}/I_2)]} \]  

(12)

Suppose that the file server and the database have the same model. The vulnerability measure of the composition of AppSrv2 and FileSrv is calculated by the following equation:

\[ v_2 = \text{MAX} \left\{ v(\text{AppSrv2}), v(\text{FS}), \right\} \]

\[ \times \frac{1}{\text{MIN}[\text{ST}_{\text{AppSrv2}}^{t_2} + 1/v(\text{FS}/I_1), \text{ST}_{\text{AppSrv2}}^{t_2} + 1/v(\text{FS}/I_2)]} \]  

(13)

where FS is the abbreviation of FileSrv.

Now we can use Theorem 5.3 to calculate the vulnerability measure of the whole system as follows.

\[ v = \text{MAX}[v_1, v_2]. \]  

(14)

Now, we use some arbitrary values and compare the quantification results for two different intruders who have different capabilities. Assume that the first attacker is stronger than the second one. The transition costs and the service time are demonstrated in Table 1.

| TABLE 1. Sample values of the service time and event costs for two intruders. |
|--------------------------------|-----------------|-----------------|
| Parameters                     | Intruder1 | Intruder2 |
| STF(FW1) = STB(FW1)            | 1          | 2              |
| STF(FW2) = STB(FW2)            | 1          | 2              |
| STF(DMZ) = STB(DMZ)            | 2          | 4              |
| c1                              | 6          | 12             |
| c2                              | 3          | 10             |
| c3                              | 5          | 11             |
| c1'                             | 26         | 30             |
| c2'                             | 30         | 40             |
| c3'                             | 28         | 35             |
| cost(DB-guest-access)           | 1          | 1              |
| cost(DB-user-access)            | 1          | 1              |
| cost(attack)                    | 5          | 7              |
| cost(FileSrv-guest-access)      | 1          | 1              |
| cost(FileSrv-user-access)       | 1          | 1              |
| cost(attack1)                   | 20         | 24             |
| cost(attack2)                   | 10         | 15             |
| cost(attack3)                   | 2          | 6              |

Using these values, the vulnerability measure of the network against attackers I1 and I2 will be equal to 0.0435 and 0.0263, respectively. In other words, a stronger attacker can violate security policies easier. Thus, the network is more vulnerable against the first intruder than the second one.

Now, suppose that administrator installs some security patches on the database server which harden the buffer overflow attack. If the cost of the buffer overflow attack increases to 25, the vulnerability measure of the system against the first attacker will decrease to 0.0333. One of the advantages of our model is that we can simply replace a component by another one and still use previous measures of unchanged components. For example, if we change just the cost of the buffer overflow attack, only the conditional vulnerability measures of the database, \( v(\text{DB}/I_1) \) and \( v(\text{DB}/I_2) \), will change from 1/12 and 1/2 to 1/20 and 1/25, respectively; and all other calculated measures could be used in new computation. Therefore, we can simply calculate \( v_1, v_2 \) and \( v \) using Equations (12)–(14), respectively.

Obviously, the size of the model is the maximum size of its components because we analyze each component separately. For example, there are four components in the mentioned case study, and each one includes four or five states. Thus, the state space will be equal to 5. Although all \( 5 + 4 + 5 + 4 = 18 \) states should be analyzed, each component could be examined separately. In other words, they do not occupy memory simultaneously.

If we wanted to analyze the entire system as a global component, the system size would reach \( 5 \times 4 \times 5 \times 4 = 400 \) states because all these components work in a parallel manner. This simple example has been presented to show the size of the state machine that is used to model the system. In the more complicated and detailed systems, the size would rapidly increase and reach intolerable values. The compositional approach to analyze the model can reduce the size of the state machine, particularly when the system could be divided into many small components.

7. DISCUSSION

To quantify the vulnerability of the system, we should estimate the model parameters by numerical values. In the case study presented in Section 6, we used some arbitrary values. However, some of these parameters which related to the system delay time could be calculated through component properties specified by manufacturers. Other parameters which depend on the attacker capabilities could be estimated by the method introduced in [25] by McQueen et al.

They have divided the attack process into three sub-processes. Process 1 is for the case where at least one vulnerability is known and its exploit code is available. Process 2 is for the case where at least one vulnerability is known but no exploit code is available to the attacker. Process 3 is the identification of new vulnerabilities and exploits. McQueen et al. have proposed the
following formula:

\[ T = t_1 P_1 + t_2 (1 - P_1) (1 - u) + t_3 (1 - P_1) u, \]

where \( T \) is the expected value of the time to attack; \( t_1, t_2 \) and \( t_3 \) are the expected value of process 1, 2 and 3, respectively, \( P_1 \) is the probability that process 1 is successful; and \( u \) is the probability that process 2 is unsuccessful. These parameters are thoroughly defined and estimated in [25] and used in [1, 26].

We can utilize the same formula in our model to calculate the event delay time considering attacker capabilities. States in our model are equivalent to the components of McQueen’s model. Therefore, \( T \) approximates to ED. It would be the best approximation of ED through available attack time estimation methods. Moreover, we can use AM/V, introduced by McQueen et al., as the skill level function, where AM is the average number of the vulnerabilities for which exploits can be found or created by the attacker given her capabilities and \( V \) is the number of available vulnerabilities on the specified state.

Another issue which should be discussed in our method is about attacker capabilities and classification. We have considered that every attacker can violate security policies by spending enough time. In other words, compromising a system is not a binary characteristic of the attacker. However, skilled attackers can compromise systems faster than beginners. This approach seems to be consonant with the real attack process because if an attacker spends enough time, he will finally be able to compromise a system. For instance, most of unpublished attacks will eventually be revealed to the public.

Nevertheless, one would expect to find the minimum initial attacker capabilities for compromising the system. For example, it would be essential for an attacker to have local access to the network as an insider to compromise a special system. In other words, an outsider attacker would not be able to violate the security policies of the system. In this case, some existing work like [13, 20, 21], which study hardening methods or weakest adversary analysis, could be utilized. But, in this paper we focus on the vulnerability measurement and complexity reduction method. Nonetheless, if an attacker does not have a special capability that is essential to trigger an attack event, we assign zero to the event probability. Therefore, the cost of event tends to infinity and the attacker cannot violate the system policy through a relevant path. Thus, the system will be secure against this attack by considering attacker capabilities.

Moreover, we can define a threshold for vulnerability tolerance and compare the calculated vulnerability measure with this threshold. If the vulnerability measure of the system against an intruder exceeds a specified threshold, we can conclude that the system is vulnerable against that attacker.

For example, in the case study mentioned in Section 6, the system vulnerability will tend to zero if both application servers are secure. In other words, if an attacker is not able to gain user or root access to the application servers, she cannot violate the security policies. If the inner firewall denies all external arrival packets, only insider attackers could compromise the system. Thus, it is not possible for an external attacker to gain any access to the application server. The probability of this attack will be equal to zero and consequently the attack cost tends to infinity. In this case, we can conclude that the system is secure against external attackers. Obviously, even the firewall could be violated by the attacker and the system would be compromised. Thus, it seems better to use a specific threshold instead of infinity.

It would be interesting to know which kind of attacker can violate security policies easier. The attacker who has the capabilities that are essential to travel through the shortest path can violate security policies easier. However, these capabilities depend on system configuration and vulnerabilities. As some interesting future work, we can find these capabilities and also find the vulnerabilities which should be fixed to reduce the vulnerability measure to an acceptable value. The vulnerability measure of systems against various kinds of attackers requires a concrete classification of attackers and needs more research in the future.

One of the advantages of using a compositional approach in model checking is state reduction. Suppose that the states of component \( i \) could be represented by \( n_i \) Boolean variables. It is not a real situation but stated just to investigate the performance of the model. In this case, \( 2^{n_i} \) states should be investigated for each component and therefore \( \sum 2^{n_i} \) states for the whole system. If we do not use the compositional method or any other state reduction technique, this quantity will soar and reach \( 2^{\sum n_i} \) states. Clearly, in a real situation where system states are represented by more complex variables such as integers, these quantities will be more than the above calculated measures. However, using this simple example would demonstrate the advantage of using our compositional approach in model checking.

One would expect to verify the model and perform model fitting. For this purpose, it is essential to have some standard measures and compare them with the estimated values obtained from model analysis. At this time, we could not find a standard measure for the vulnerability of the systems. Therefore, model fitting is not possible in this case. In other words, the final goal of this paper is to define a new metric that has compositional property and surmounts shortcomings of previous vulnerability measurement methods which are stated in Section 2. For this purpose we have represented a new metric to quantify vulnerability and therefore compare the security of different systems against various kinds of attackers.

There are some compositional quantitative methods that analyze system performance. However, we do not use them to quantify system security. In the remaining of this section, we discuss the issue which motivated us to introduce a new model rather than adjusting existing ones.

Two main approaches for performance analysis are queuing systems [27] and Petri nets [28]. But traditional forms of these approaches are not helpful for complex system design [29]. The
The combination of a process algebra and Markov chain was constructed complex model, these methods would not be useful to analyze the system components. Since the purpose of our model is to use Markov chain, although this method reduces the state space, we cannot calculate system performance using available analyses. However, the constructed model should be analyzed using a continuous time process algebra [36]. But they do not present a global solution. Thus, we cannot use them to design a model. Process algebra introduced a method that could easily model compositionality [33]. In this approach, complicated models are constructed from smaller blocks. An abstraction mechanism provides a method that makes the internal structure invisible. The combination of a process algebra and Markov chain was considered as a solution to analyze large-scale industrial systems in [34]. This study embarked on some approaches [35–37] which are subsumed as stochastic process algebra. The calculus of Interactive Markov Chains developed in [29] introduces compositional aggregation algorithms to reduce state space.

Two other compositional approaches to estimate the steady-state probabilities are stated in [32]. Courtois’ approximate method and Takahashi’s iterative method calculate steady-state probabilities by decomposing state space and analyzing each part separately. These methods can solve some special problems but they do not present a global solution. Thus, we cannot use them to design a model.

Probabilistic Input/Output Automata (PIOA) which has been introduced in [38] provides a method for modeling a distributed system. On the basis of the time independency property, a combinational rule has been stated. Thus, interactive PIOAs could construct a more complex PIOA.

Clearly, the goal of all studies mentioned above is to represent an abstract model that could hide details of internal events and solve or at least reduce the state-space explosion problem. For example, suppose that there are two components working concurrently and synchronized on a single event. The composition of these components could be easily modeled by performance evaluation process algebra [36]. However, the constructed model should be analyzed using a continuous time Markov chain. Although this method reduces the state space, we cannot calculate system performance using available analyses of its components. Since the purpose of our model is to use vulnerability measures of system components to analyze the constructed complex model, these methods would not be useful here.

8. CONCLUSION AND FUTURE WORK

In this paper, we introduced a new metric to quantify the vulnerability of computer systems. Using the vulnerability measure, the system administrators or managers could prioritize available security solutions and countermeasures and make better decisions about system hardening.

The vulnerability measure is not an absolute value but it depends on the threatening adversary and its capabilities. Accordingly, the proposed method utilizes potential attacks on the system and the attacker’s capabilities to model the system. Thus, the system vulnerability measures against different attackers might not be similar.

The main advantage of the approach is its compositional property. We can calculate the vulnerability measure of a complex system using the pre-calculated vulnerability quantities of its components. Leveraging the divide and conquer approach, the system is broken down to some components whose vulnerability measures are pre-calculated or at least easier to calculate. On the other hand, replacing a component by another one does not affect the vulnerability measures of other components. Therefore, they can be reused to analyze the new system.

Another advantage of the method is that it takes the penetration delay time into account in addition to the probability of accessing unauthorized states. Thus, the vulnerability measures of different systems that have the same probability to reach unauthorized states could be compared.

We have assumed that all unauthorized states have the same importance and impact level. But it seems that this assumption is not always true. For instance, if an intruder gains root access to an asset, she can do more damage to the system in contrast to the situation in which she has only a normal user access. Thus, extending the approach to consider different impact levels of unauthorized states could be a direction of future work.

Moreover, as we stated in Section 7, an interesting research could be attacker classification and measuring vulnerability against different kinds of attackers. Furthermore, finding the set of vulnerabilities that should be fixed to reduce the vulnerability measure to an acceptable value could be a useful outcome. Conducting research into these subjects would be an appealing work in the future.

FUNDING

This work was supported by the Research Institute for Information and Communication Technology (ITRC) [13274/500].

REFERENCES

Workshop on Quality of Protection (QoP ’07), Virginia, USA, October 29, pp. 49–54. ACM, New York.


