Measuring the Overall Security of Network Configurations Using Attack Graphs

Lingyu Wang[1], Sushil Jajodia[2], and Anoop Singhal[3]

[1] Concordia Institute for Information Systems Engineering
Concordia University
Montreal, QC H3G 1M8, Canada
wang@ciise.concordia.ca
[2] Center for Secure Information Systems
George Mason University
Fairfax, VA 22030-4444, USA
jajodia@gmu.edu
[3] Computer Security Division, NIST
Gaithersburg, MD 20899, USA
anoop.singhal@nist.gov

Abstract. Today's computer systems face sophisticated intrusions during which multiple vulnerabilities can be combined for reaching an attack goal. The overall security of a network system thus can no longer be determined based on the number of vulnerabilities. To quantitatively assess the security of networked systems, one must first understand which and how vulnerabilities can be combined for an attack. Such an understanding becomes possible with recent advances in modeling the composition of vulnerabilities as *attack graphs*. Based on our experiences with attack graph analysis, we propose an *attack resistance* metric for assessing and comparing the security of different network configurations. The metric is based on intuitive properties derived from common sense. We first describe the metric at an abstract level as two composition operators with features for expressing additional constraints. We then consider two concrete cases. The first assumes the domain of attack resistance to be real number and the second represents resistances as sets of initial security conditions. We show that the proposed metric satisfies desired properties and adhere to common sense. At the same time, it generalizes a previously proposed metric that is also based on attack graphs. It is our belief that the metric will lead to novel quantitative approaches to vulnerability analysis, network hardening, and attack responses.

1 Introduction

Today's networked computer systems constitute the central component of information technology infrastructures in enterprises and in critical infrastructures, including power grids, financial data systems, and emergency communication systems. Protecting such systems against malicious intrusions is crucial to the economy and to our national security. Having a standard way for measuring various aspects of network security will bring together users, vendors, and labs in specifying, implementing, and evaluating the requirements and features of network security products. However, in spite of various efforts in standardizing security metric, a widely-accepted metric for network security is still largely unavailable. This is partly due to the fact that most researchers are still adopting a qualitative and imprecise view toward the evaluation of network security. For example, typical issues addressed in current research may ask following questions. Are all given critical resources in a network secure (topological vulnerability analysis)? Can a network be hardened in some way to secure the given resources (network hardening)? How to stop an ongoing intrusion from compromising given resources (attack response)?

The qualitative nature of these questions reflect the current focus on the qualitative, rather than quantitative, study of network security. This focus implies the inherent impreciseness in many research results and also indicates the needs for more research efforts on security metric. However, the lack of research on quantitative aspects of network security is natural. Assessing the overall security of a network requires a thorough understanding of the interplay between host vulnerabilities. That is, which and how can vulnerabilities be combined for an attack. This understanding is, however, difficult with existing security tools, such as vulnerability scanners and intrusion detection systems. These tools typically focus on identifying individual vulnerabilities or attacks, and are usually unaware of the relationships among vulnerabilities or attacks.

Recent advances in modeling compositions of vulnerabilities using *attack graphs* (a review of related work will be given in the next section) indicate that the research has progressed to a point where the quantitative study of network security is critical and, at the same time, possible. Attack graphs supplement vulnerability scanners with the missing information about relationships among vulnerabilities. Analyzing the correlated vulnerabilities thus provides a clear picture about what attacks might happen in a network and about their consequences. Attack graphs thus allow us to consider potential attacks in a particular *context* relevant to the given network. The current work is based on our past experiences with attack graph analysis [16, 15, 12, 30, 28, 29, 31, 20] and a practical tool, Topological Vulnerability Analysis (TVA) system, developed for generating attack graphs for more than 37,000 vulnerabilities taken from 24 information sources including X-Force, Bugtraq, CVE, CERT, Nessus, and Snort [12]. The presence of such a powerful tool demonstrates the practicality of using attack graphs as the basis for measuring network security.

Instead of measuring individual vulnerabilities and then wondering about their combined effect, this paper measures the overall security of a network using the context provided by an attack graph. Such a capability will enable us to answer important questions like (but are not limited to): How much efforts and time will it take to compromise a critical resource under each possible network configuration? Answers to such questions will allow system administrators to choose the optimal configuration that is the most resistant to potential attacks. More specifically, we propose an *attack resistance* metric for assessing and comparing the security of different network configurations. The metric is based on intuitive properties derived from common sense. For example, our metric will indicate reduced security when more attack paths exist, whereas it indicates increased security for longer and more difficult paths. To make the metric broadly applicable, we first describe it at an abstract level as two composition operators with functions that allow for expressing additional dependency relationship between resistances. We then consider two concrete cases. The first assumes the domain of attack resistance to be real number and the second represents resistances as sets of initial security conditions. For the first case, we propose to use operators that are analogous to the ones used in computing resistance of a series and parallel circuit. We study additional issues that arise due to the unique properties of attack graphs. For the second case, we show that the previously proposed metric is equivalent to our metric under certain conditions.

2 Related Work

An overview of various issues relevant to security metric is recently given in the proceedings of the 2001 Workshop on Information Security System Scoring and Ranking [2]. The NIST's efforts on standardizing security metric are reflected in the Technology Assessment: Methods for Measuring the Level of Computer Security [17] and more recently in the Security Metric Guide for Information Technology Systems [26], which describe the current state of practice of security metrics, such as that required by the Federal Information Security Management Act (FISMA). Another overview of many aspects of network security metric is given in [10].

Closest to our work, Dacier et. al describe intuitive properties derived from common sense, which should be satisfied by any security metric [7, 8, 18]. They suggest to assess the difficulty of attacks in terms of time and efforts spent by attackers. They assume an exponential distribution for an attacker's success rate over time. Based on this Markov model, they propose to use the MTTF (Mean Time to Failure) to measure the security of a network. They discuss simple cases of combining such measures but do not study the general case. We borrow some of the intuitive properties stated by them, but we use a different way for combining individual measures into the overall attack resistance and we consider a more general case represented by attack graphs.

Our approach of using additional functions for modeling the effect of executed exploits on the resistance value of other exploits is inspired by the work by Balzarotti et. al [3]. However, their work focus on computing the minimum efforts required for executing each exploit, whereas our work computes the overall security of a network with respect to given critical resources. Also, their work does not take into account the kind of dependency that we model using additional functions. Such dependency reduces the difficulty of executing an exploit while not directly enabling it to be exploitable. The work by Pamula et. al introduces a metric based on attack graph [20], in this paper we show that their metric is a special case of ours under certain conditions.

A qualitative measurement of the risk of a network is given based on various forms of the exploitability (that is, whether it is possible to compromise the network) [4]. Another series of work compares software for their relative vulnerabilities to attacks using a fixed set of dimensions, namely, *attack surface* [11, 19, 13]. The work by Mehta et. al borrows Google's PageRank methodology to rank exploits in an attack graph [14]. Their technique is especially suitable for threat models of worms or other malicious software that spread in a random way in a large network. Our metric has a different threat model, that is attackers have memory and are rational, so in most cases they will not follow a random model.

Metrics for other perspectives of security, especially trust in distributed systems, are relevant to our research. For example, Beth et. al proposed a metric for measuring the trust in an identity established through overlapping chains of certificates [5]. The way they combine values of trust in each certificate into an overall value of trust proves to be useful in our study. Similarly, the design principles given by Reiter et. al are intended for developing metric of trust, but we found these principles applicable to our study as well [23]. The formal logic language introduced for measuring risks in trust delegation in the RT framework inspires us to describe our metric using abstract operators [6].

To obtain attack graphs, topological vulnerability analysis evaluates potential multi-step intrusions based on knowledge about vulnerabilities [7, 9, 18, 21, 32, 27]. Such analyses can be either forward starting from the initial state [21, 27] or backward from the goal state [24, 25]. Model checking was first used to analyze whether a given goal state is reachable from the initial state [22, 24] and later used to enumerate all possible sequences of attacks between the two states [25]. To avoid the exponential explosion in the number of such explicit attack sequences, a more compact representation of attack graphs was proposed based on the *monotonicity assumption* saying an attacker never needs to relinquish any obtained capability [1]. On the attack response front, attack graphs have been used for the correlation of attacks, the hypotheses of alerts missed by IDSs, and the prediction of possible future attacks [28, 29].

3 Preliminaries

This section first reviews the attack graph model and then discusses intuitions behind the proposed metric.

3.1 Attack Graph Model

We adopt the attack graph model used in Topological Vulnerability Analysis tool [12], which is one of the most advanced utilities for generating and analyzing attack graphs. This attack graph model is similar in nature to the earlier ones based on modified model checking [25], but it avoids the potential combinatorial explosion faced by the latter. More specifically, it makes a *monotonicity assumption* stating an attacker never relinquishes an obtained capability [1]. An attack graph can thus record the dependency relationship between exploits instead of recording all attack paths. The resulting attack graph has no duplicate vertices and hence has a polynomial size in the number of vulnerabilities multiplied by the number of connected pairs of hosts.

In our model, an *Attack graph* is a directed graph representing prior knowledge about vulnerabilities, their dependencies, and network connectivity. The vertices of an attack graph are divided into two categories, namely, *exploits* and *security conditions* (or simply *conditions* when no confusion is possible). First, exploits are actions taken by attackers on one or more hosts in order to take advantage of existing vulnerabilities. We denote an exploit as a predicate. For example, an exploit involving three hosts can be denoted using $v(h_s, h_m, h_d)$, which indicates an exploitation of the vulnerability v on the destination host h_d, initiated from the source host h_s, through an intermediate host h_m. Similarly, we write $v(h_s, h_d)$ or $v(h)$, respectively, for exploits involving two hosts (no intermediate host) or one (local) host.

Second, a security condition is a property of systems or networks relevant to a particular exploit in the sense that it is either required for that exploit or is satisfied by the exploit. We also use a predicate to represent a condition involving one or more hosts. For example, $c(h_s, h_d)$ indicates a security-related condition c involving the source host h_s and the destination host h_d. Similarly, a condition that only involves a single host can be written as $c(h)$. Examples of security conditions include the existence of a vulnerability, the existence of network connectivity or trust relationship between two hosts. It is worth noting that an attack graph usually includes exploits and conditions corresponding to normal services or functionality. Such services are included because they may help attackers in escalating their privileges when combined with other exploits, although they are not intended for that purpose. This also reflects the fact that not all exploits can be removed by hardening a network, so measuring the relative security becomes important.

Directed edges in an attack graph inter-connect exploits with conditions. No edge directly goes between two exploits or between two conditions. First, an edge from a condition to an exploit denotes the *require* relation, which means the exploit cannot be executed unless the condition is satisfied. Second, an edge pointing from an exploit to a condition denotes the *imply* relation, which means executing the exploit will satisfy the condition. For example, an exploit typically requires at least two conditions, that is the existence of the vulnerability (which could be a normal service) on the destination host and the network connectivity between the two hosts. We formally characterize attack graphs in Definition 1.

Definition 1. *Given a set of exploits $\mathcal{E}$, a set of conditions $\mathcal{C}$, a* require *relation $require \subseteq \mathcal{C} \times \mathcal{E}$, and an* imply *relation $imply \subseteq \mathcal{E} \times \mathcal{C}$, an* **attack graph** G *is the directed graph $G(\mathcal{E} \cup \mathcal{C}, require \cup imply)$ ($\mathcal{E} \cup \mathcal{C}$ is the vertex set and $require \cup imply$ the edge set).*

One important semantics of attack graphs is that the require relation is conjunctive, whereas the imply relation is disjunctive. More precisely, an exploit cannot be realized until *all* of its required conditions have been satisfied, whereas a condition is satisfied if *any* of the realized exploits implies that condition. Sometimes only exploits in an attack graph are of interest, we thus remove conditions to obtain an *exploit dependency graph*. However, in such a graph edges between exploits may represent both the conjunctive and the disjunctive relationship. For example, in an attack graph if two exploits e_1 and e_2 both imply the same condition c_1, which is required by another exploit e_3, then e_3 can be executed after executing either e_1 or e_2 (since c_1 will be satisfied by any of them). On the other hand, if e_1 implies c_1, e_2 implies a different condition c_2, and both c_1 and c_2 are required by e_3, then e_3 cannot be executed before both e_1 and e_2 are. We shall need this observation later in the paper.

3.2 Motivating Example

To build intuitions about properties that a security metric should satisfy, we consider the well-known attack scenario as shown on the left hand side of Figure 1 (notice that this is an overly simplified example for illustration purposes and our metric and techniques are intended for more complicated cases where results usually cannot simply be obtained through observations). In this attack graph, exploits are depicted in ovals and conditions in cleartext. The critical condition that needs to be guarded is shown in a shaded oval. The attack graph basically indicates that an attacker on host 0 can obtain user privilege on host 1, either using an SSH buffer overflow attack or through the trust relationship established by uploading the .rhost file through FTP. The attacker can use the latter trick to obtain user privilege on host 2, either directly from host 0 or using host 1 as an intermediate stepping stone. The attacking goal, that is the root privilege on host 2, can then be obtained using a local buffer overflow attack.

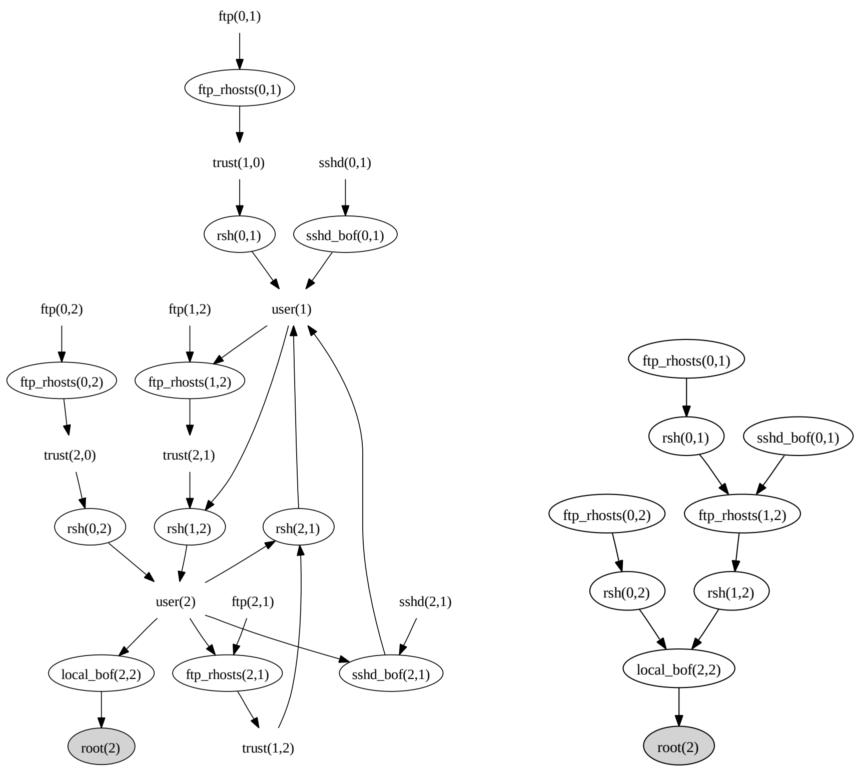


Fig. 1. An Example of Attack Graph and Exploit Dependency Graph

The right hand side of Figure 1 shows the exploit dependency graph. It is worth noting that in this specific case only disjunctive dependency relationship exists between exploits. For example, there are two alternative ways to reach the exploit $ftp\_rhosts(1, 2)$ and similarly two ways to reach the exploit $local\_bof(2, 2)$. In general, the dependency relationship between exploits can be both disjunctive and conjunctive, and the graph notation is thus not sufficient to distinguish between the two. Later we shall introduce a special notation for this purpose.

We make several observations in Figure 1. First, the two loops via exploit $ftp\_rhosts(2, 1)$ and $sshd\_bof(2, 1)$ are both removed. These both allow the attacker to obtain the user privilege on host 1 for the second time, after the privilege has already been obtained (otherwise the two exploits cannot be executed). An attacker can certainly make redundant attacking efforts at will, but a security metric should assume the most efficient attackers and reflect the security of a network in the worst-case scenario. That is, *a metric should always measure the smallest attacking efforts required for reaching the attack goal*.

Second, the exploits in Figure 1 clearly have different difficulty in terms of the time and efforts required for their execution. For example, the ftp_rhosts and rsh exploits both take advantage of normal services in a clever way, and they usually do not require much time or efforts if the attacker has the basic knowledge about the attack (between the two type of exploits, rsh may be slightly easier than ftp_rhosts in the sense that the latter requires crafting the .rhost file). On the other hand, both the $sshd\_bof$ and the $local\_bof$ are buffer overflow attacks, which require significantly more knowledge and time than the previous two because a buffer overflow attack usually requires bruteforce efforts to determine proper parameters. This example thus shows *different exploits have different difficulties in terms of efforts and time required for their execution*.

Third, there are three possible attack paths (that is, sequences of attacks) reaching the attack goal, as shown on the right hand side of Figure 1, the left path (that is, the one through $ftp\_rhosts(0,2)$ and $rsh(0,2)$) requires the smallest amount of efforts. The middle path requires slightly more efforts since it involves both host 1 and host 2. The right path demands the most efforts because it requires an additional buffer overflow attack $sshd\_bof(0,1)$. Recall the above argument that security should be measured as the smallest efforts required to reach the goal. It seems that the left path is a good candidate to be used as the measure of overall security. However, it is important to notice that when multiple paths coexist in an attack graph, reaching the attack goal is actually easier than if only one of these paths exists (even if the path requires the smallest amount of efforts). Intuitively, more attack opportunities mean less security, because attackers will have a better chance to reach the attack goal. In this specific case, even though the middle and the right paths are more difficult than the left one, they nevertheless represent possibilities for attacks and thus they do reduce the overall security of the network. That is, *multiple attack paths together are less secure than any of the paths alone*.

Finally, assuming the middle attack path is followed by an attacker, it can be argued that the exploit $ftp\_rhosts(1,2)$ may be slightly easier than its predecessor $ftp\_rhosts(0,1)$. To launch the same type of attack for the second time, the attacker will benefit from his/her experiences and tools that have been accumulated while launching the attack for the first time. It is, however, not possible to add an edge between these two exploits in attack graph, because the exploit

$ftp\_rhosts(1,2)$ does not directly depend on $ftp\_rhosts(0,1)$ (with $rsh(0,1)$ in the middle). This implies that an additional relation is needed to encode such dependency relationship between exploits, which is different from the $imply$ or $require$ relations already encoded in attack graphs. In another word, *executing an exploit may change the difficulty of executing another exploit, even if the two do not directly depend on each other in the attack graph*.

The above requirements are largely common sense that should be satisfied by a security metric. The rest of the paper proposes a security metric based on the attack graph model by taking these requirements into consideration.

4 An Attack Resistance Metric

This section proposes an attack resistance metric based on the attack graph model. We first discuss the metric in a generic form. We then discuss two concrete cases to illustrate the metric in more details. We address various issues encountered while computing the metric from a given attack graph.

4.1 A Generic Framework

We propose to measure the *attack resistance* of a network configuration based on the composition of measures of individual exploits. Ideally, the resistance of each type of exploits in terms of efforts and time should be represented as a total order, such as using real numbers (the next section considers how individual resistances can be combined when attack resistance is represented as a real number). Unfortunately, although clearly desired, the information and resources required by this ideal situation are largely not yet available. It is, however, usually possible to estimate an approximate ordering or a partial ordering on the domain of attack resistance. We shall also consider another case where the resistance of individual exploit is simply the set of initial conditions (that is, conditions that are not implied by other exploits) required by the exploit.

Different applications may define the attack resistance of individual exploits in significantly different way. To make our metric broadly applicable, we describe the metric in a generic form while leaving the individual measures uninterpreted. Central to the model are two types of composition operators, denoted as $\oplus$ and $\otimes$. The two operators correspond to the disjunctive and conjunctive dependency relationship between exploits in an attack graph, respectively. Based on the intuitive properties mentioned in Section 3.2, the two operators should satisfy that $r_1 \oplus r_2$ is no greater than r_1 or r_2, whereas $r_1 \otimes r_2$ is no less than r_1 and r_2, with respect to a given ordering on the domain of attack resistance.

In addition to the two composition operators, we introduce a function $\mathcal{R}()$ that maps a set of exploits to another exploit and its resistance value. The function is intended to capture a special kind of dependency relationship between exploits. That is, executing some exploits may affect the resistance value of another exploit, even though the latter cannot be executed yet. In most cases, this effect will be to assign a lower resistance value to the affected exploit. For example, exploits involving the same vulnerability should be related together using this function such that successfully exploiting one instance of the vulnerability reduces the resistance of others due to the attacker's accumulated experiences and tools. We shall also show that this function is useful in handling the non-tree structure of attack graphs. We summarize the model in Definition 2.

Definition 2. *Given an attack graph $G(\mathcal{E} \cup \mathcal{C}, require \cup imply)$ with attack goals $g \subseteq \mathcal{C}$, the attack resistance metric is composed of*

- *A total function $r() : \mathcal{E} \to \mathcal{D}$,*
- *a total function $R() : \mathcal{E} \to \mathcal{D}$,*
- *an operator $\oplus : \mathcal{D} \times \mathcal{D} \to \mathcal{D}$,*
- *an operator $\otimes : \mathcal{D} \times \mathcal{D} \to \mathcal{D}$, and*
- *a function $\mathcal{R}() : \mathcal{E} \to \mathcal{E} \times \mathcal{D}$.*

We call the set $\mathcal{D}$ the **domain** *of resistance, $r(e)$ the* **individual resistance** *(or simply resistance) of an exploit e, $R(e)$ the* **cumulative resistance** *of e.*

The main tasks in implementing this metric for a specific application is to populate the individual resistance by defining the function $r()$, to determine suitable operators $\oplus$ and $\otimes$, to capture additional dependency relationship between exploits using the function $\mathcal{R}$, and finally to decide how the cumulative resistance function $R()$ should be computed based on these information. The cumulative resistance of each attack goal then provides a quantitative measure as how likely that attack goal can be achieved, or equivalently, how vulnerable the corresponding resource is with the given network configuration.

4.2 Attack Resistance As Real Numbers

We now consider a concrete case where the domain of resistance $\mathcal{D}$ is the non-negative real number. Analogous to the resistance of a series and parallel circuit, we define $\oplus$ as the reciprocal of the sum of the reciprocal of individual resistance values. That is, $\frac{1}{r_1 \oplus r_2} = \frac{1}{r_1} + \frac{1}{r_2}$. The operator $\otimes$ is simply addition. Recall our discussions about the relative difficulty of different type of exploits in Section 3.2. Suppose we assign the value 10 to be the resistance of each $sshd\_bof$

and $local\_bof$, the value 2 and 1 to each ftp_rhosts and rsh exploit, respectively, as depicted on the left hand side of Figure 2. The cumulative resistances can then be computed as follows, where $r()$ stands for the individual resistance and $R()$ the cumulative resistance (we shall not consider the function $\mathcal{R}$ for the time being). The final results are shown in the right hand side of Figure 2.

- $R(rsh(0,1)) = r(ftp\_rhosts(0,1)) + r(rsh(0,1)) = 2 + 1 = 3$
- $R(ftp\_rhosts(1,2)) = 1/(1/R(rsh(0,1)) + 1/r(sshd\_bof(0,1)))+$ $r(ftp\_rhosts(1,2)) = 1/(1/3 + 1/10) + 2 \approx 4.3$
- $R(rsh(1,2)) = R(ftp\_rhosts(1,2)) + r(rsh(1,2)) \approx 4.3 + 1 = 5.3$
- $R(rsh(0,2)) = r(ftp\_rhosts(0,2)) + r(rsh(0,2)) = 2 + 1 = 3$
- $R(local\_bof(2,2)) = 1/(1/R(rsh(0,2)) + 1/R(rsh(1,2)))+$ $r(local\_bof(2,2)) = 1/(1/3 + 1/5.3) + 10 \approx 11.9$

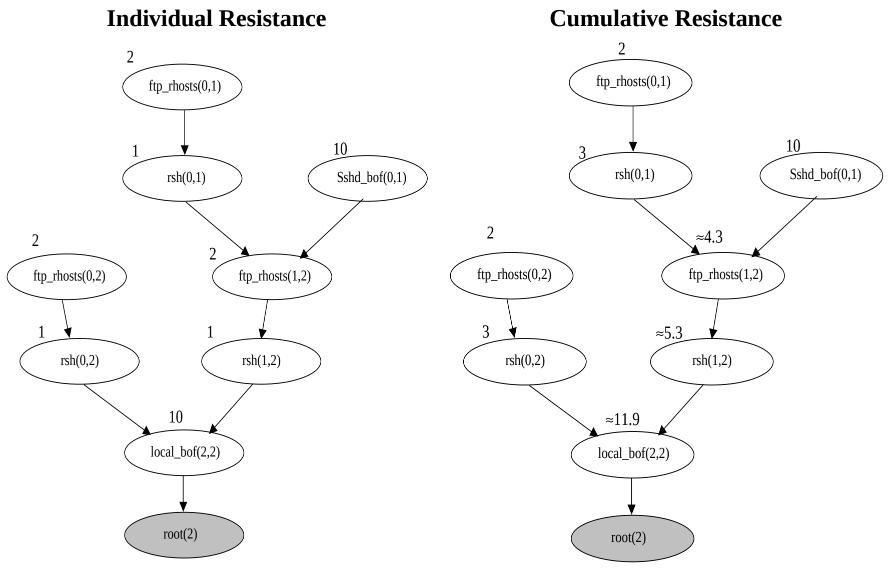
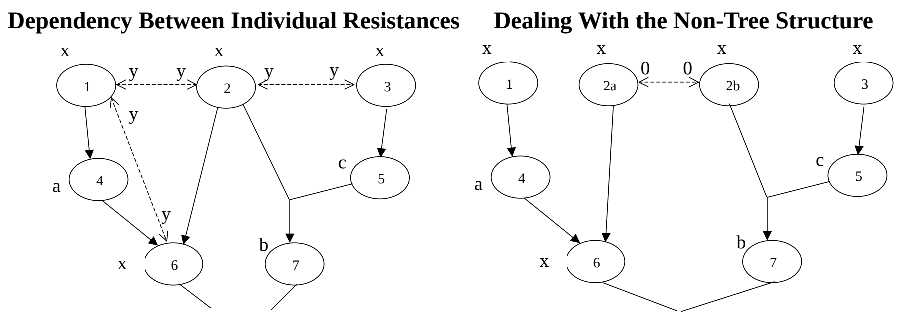


Fig. 2. An Example of Attack Resistance in Real Number

According to our discussions in Section 3.2, the cumulative resistance of the whole network should be smaller than the cumulative resistance of each possible attack path. The cumulative resistance for each attack path reaching the goal can be computed by simply adding (that is, the $\otimes$ operator) individual resistance values along the path. The results for the three attack paths in Figure 2 are 13, 16, and 23, from left to right. Clearly, the accumulative resistance of the whole network, 11.9, is indeed smaller than any of the three values, satisfying the intuitive requirements given in Section 3.2. We may also notice that the composition (using the operator $\oplus$) of these three resistance values is about 5.5, which is less than the computed resistance 11.9. This reflects the fact that the three paths are not disjoint. The value 5.4 is computed under the implicit assumption that the three paths are disjoint, which is not the case here. Intuitively, having common exploits among different paths may increase the overall attack resistance, because the attacker must execute these exploits no matter what path they follow. Our metric naturally takes into consideration the overlapping portion of the paths.

The function $\mathcal{R}$ Next we consider the function $\mathcal{R}$, that is the effect of executed exploits on the individual resistance of other exploits. The previous example is not sufficient for this purpose. Instead we consider the abstract example given in the left hand side of Figure 3, where the dotted lines represent following facts. Between the exploit 1 and the exploit 2, executing one will change the other's individual resistance from the original value x to a new value y. Similar relationship exists between the exploit 2 and the exploit 3, and between the exploit 1 and the exploit 6. Notice the special notation between the exploits 2, 5, and 7, which denotes the conjunctive relationship between the exploits 2 and 5. That is, exploit 7 cannot be executed unless both exploit 2 and 5 are already executed (this may happen when the exploit 2 and the exploit 5 both imply different conditions, and both conditions are required by the exploit 7).


Dependency Between Individual Resistances
Dealing With the Non-Tree Structure
x
y
1
2
3
4
5
6
7
2a
2b
a
b
c
0
Fig. 3. Examples of the Function $\mathcal{R}$

The left hand side of Figure 3 shows three possibilities in dealing with the function $\mathcal{R}$. First, the effect of $\mathcal{R}(6) = (1, y)$ (that is, executing the exploit 6 will change the individual resistance of the exploit 1 as $r(1) = y$) can be safely ignored, because the exploit 6 can never be executed before executing the exploit 1. On the other hand, the individual resistance $r(6)$ can now simply be changed from x to y, because any execution of exploit 6 implies that the exploit 1 has already been executed (which in turn implies a change in $r(6)$). Second, there is no naturally induced order between the execution of the exploit 1 and that of the exploit 2, so they can be executed in any order. Intuitively, these two exploits *meet* at the exploit 6 in the sense that we combine these two resistance values in the same formula when we compute $R(6) = 1/(1/(r(1) + r(4)) + 1/r(2)) + r(6)$. At that point, the last composition operator used is $\oplus$ (that is, the exploit 4 and the exploit 2 are disjunctive). We can then conclude that any minimal attack path (that is, an attack path with no proper subsets being a valid attack path) including the exploit 6 will include either the exploit 1 or the exploit 2, but not both. The effect of $\mathcal{R}(1) = (2, y)$ and $\mathcal{R}(2) = (1, y)$ can thus be ignored.

Third, when the exploit 2 and the exploit 3 meet at the exploit 7 (when we compute $R(7) = r(2) + r(3) + r(5) + r(7)$, the last composition operator we use is $\otimes$. This reflects the fact that the exploits 2 and 3 must both be executed in order to reach exploit 7, although the executions can be in any order. If exploit 2 is executed before exploit 3, then we have $r(2) = x$ and $r(3) = y$; if exploit 3 is first executed, we have $r(3) = x$ and $r(2) = y$. However, we can never have $r(2) = r(3) = y$ because a change only happens after an execution. In this case, we compute the cumulative resistance of exploit 7 for both cases: $r(2) = x$, $r(3) = y$ and $r(3) = x$, $r(2) = y$. We then choose the smaller result as the cumulative resistance of the exploit 7. This choice ensures that the computed cumulative resistance will be no greater than the cumulative resistance computed by following any attack path leading to the exploit 7. The above discussion covers all possible cases, because when two exploits eventually meet (that is, their resistances are combined), they must meet either at one of themselves (the case of the exploit 1 and the exploit 6), or at a different exploit.

The Non-Tree Structure of Attack Graphs Unlike the nice tree structure in the attack graph in Figure 1, it can be noticed that on the left hand side of Figure 3 both the exploit 6 and the exploit 7 depend on the exploit 2, and the graph is not a tree. This is relevant because the cumulative resistance of this network should be different from another network where the exploit 6 and the exploit 7 depend on two different exploits. This issue, however, can be easily handled using the function $\mathcal{R}$ as follows. We split the exploit 2 into two identical copies, say, exploit 2a and exploit 2b, as shown on the right hand side of Figure 3. We then need the constraint that the resistance of these two exploits will never be added in computing a cumulative resistance, because they actually represent a single exploit. This constraint can be easily modeled as $\mathcal{R}(2a) = (2b, 0)$ and $\mathcal{R}(2b) = (2a, 0)$. We can now compute the metric as usual since the exploit dependency graph becomes a tree.

4.3 Attack Resistance As Sets of Initial Conditions

We consider another concrete case where each exploit's individual attack resistance is the set of initial conditions (that is, conditions not implied by any exploit) required by that exploit. This measure can be easily obtained from the attack graph itself. The attack resistance in terms of the set of initial conditions has a very different meaning from the attack resistance discussed in the previous section. Here the resistance indicates conditions that must be satisfied before an intrusion is possible, instead of the efforts and time spent during the actual intrusion. A weakest-adversary metric was recently proposed based on the set of initial conditions [20]. Different network configurations can be ordered based on their relative security, if a subset relationship exists between the sets of initial conditions required for reaching attack goals in the two attack graphs. We show that this metric is equivalent to a special case of our metric by using the set of initial conditions as individual resistance.

Figure 4 shows two network configurations that are comparable based on initial conditions [20]. The left hand side depicts an attack scenario similar to the one in Figure 1 but only involves two hosts. The right hand side shows a different scenario where the attacker is forced by a firewall to exploit the sendmail buffer overflow vulnerability on a third host as an intermediate step. It can be observed that in both cases the goal requires all the exploits to be executed, that is all the dependency relationship is conjunctive. We use set union as the operator $\otimes$ (we do not need the $\oplus$ operator in this case). The cumulative resistance of the exploit $local\_bof(2)$ is thus simply the collection of all initial conditions in both cases. Clearly, the resistance in the first case is a proper subset of the resistance in the second case, and hence the second case has more resistance to potential attacks. This result is the same as reported previously [20].

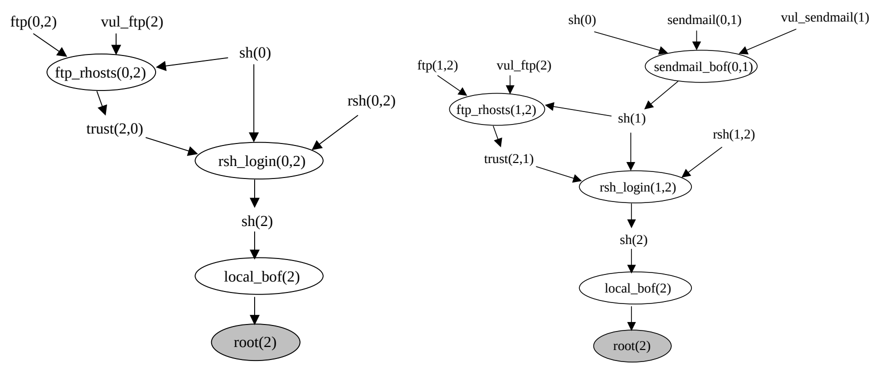


Fig. 4. Two Comparable Network Configurations

5 Conclusion

Nowadays, qualitative and imprecise arguments are usually the basis for making decisions in securing a network. These arguments can mislead the decision making and as a result cause the reconfigured network to be in fact less secure. This paper descried a novel attack graph-based attack resistance metric for measuring the relative security of network configurations. Centric to the metric were two composition operators for computing the cumulative attack resistance from given individual resistances. An additional function allowed the metric to take into consideration the dependency between individual attack resistances. We demonstrated the metric through two concrete cases. First, attack resistance was modeled as a real number, and the case was analogous to computing the resistance of a series-parallel circuit. We showed that the proposed metric satisfied intuitive requirements mentioned in the literature. Second, attack resistance was defined as the set of initial conditions required by each exploit. We showed that our metric in this case resembled the weakest-adversary metric previously proposed. It is our belief that the proposed metric will lead to novel quantitative approaches to vulnerability analysis, network hardening, and attack response.

References

1. P. Ammann, D. Wijesekera, and S. Kaushik. Scalable, graph-based network vulnerability analysis. In *Proceedings of the 9th ACM Conference on Computer and Communications Security (CCS'02)*, pages 217–224, 2002.
2. Applied Computer Security Associates. Workshop on. In *Information Security System Scoring and Ranking*, 2001.
3. D. Balzarotti, M. Monga, and S. Sicari. Assessing the risk of using vulnerable components. In *Proceedings of the 1st Workshop on Quality of Protection*, 2005.
4. P. Balzarotti, M. Monga, and S. Sicari. Assessing the risk of using vulnerable components. In *Proceedings of the 2nd ACM workshop on Quality of protection*, 2005.
5. T. Beth, M. Borcherding, and B. Klein. Valuation of trust in open networks. In *Proceedings of the Third European Symposium on Research in Computer Security (ESORICS'94)*, pages 3–18, 1994.
6. P. Chapin, C. Skalka, and X.S. Wang. Risk assessment in distributed authorization. In *3rd ACM Workshop on Formal Methods in Security Engineering: From Specifications to Code*, 2005.
7. M. Dacier. Towards quantitative evaluation of computer security. Ph.D. Thesis, Institut National Polytechnique de Toulouse, 1994.
8. M. Dacier, Y. Deswarte, and M. Kaaniche. Quantitative assessment of operational security: Models and tools. Technical Report 96493, 1996.
9. D. Farmer and E.H. Spafford. The COPS security checker system. In *USENIX Summer*, pages 165–170, 1990.
10. K.S. Hoo. Metrics of network security. White Paper, 2004.
11. M. Howard, J. Pincus, and J. Wing. Measuring relative attack surfaces. In *Workshop on Advanced Developments in Software and Systems Security*, 2003.

12. S. Jajodia, S. Noel, and B. O'Berry. Topological analysis of network attack vulnerability. In V. Kumar, J. Srivastava, and A. Lazarevic, editors, *Managing Cyber Threats: Issues, Approaches and Challenges*. Kluwer Academic Publisher, 2003.
13. K. Manadhata, J.M. Wing, M.A. Flynn, and M.A. McQueen. Measuring the attack surfaces of two ftp daemons. In *Quality of Protection Workshop*, 2006.
14. V. Mehta, C. Bartzis, H. Zhu, E.M. Clarke, and J.M. Wing. Ranking attack graphs. In *Recent Advances in Intrusion Detection 2006*, 2006.
15. S. Noel and S. Jajodia. Correlating intrusion events and building attack scenarios through attack graph distance. In *Proceedings of the 20th Annual Computer Security Applications Conference (ACSAC'04)*, 2004.
16. S. Noel, S. Jajodia, B. O'Berry, and M. Jacobs. Efficient minimum-cost network hardening via exploit dependency grpahs. In *Proceedings of the 19th Annual Computer Security Applications Conference (ACSAC'03)*, 2003.
17. National Institute of Standards and Technology. Technology assessment: Methods for measuring the level of computer security. NIST Special Publication 500-133, 1985.
18. R. Ortalo, Y. Deswarte, and M. Kaaniche. Experimenting with quantitative evaluation tools for monitoring operational security. *IEEE Trans. Software Eng.*, 25(5):633–650, 1999.
19. J. Wing P. Manadhata. Measuring a system's attack surface. Technical Report CMU-CS-04-102, 2004.
20. J. Pamula, S. Jajodia, P. Ammann, and V. Swarup. A weakest-adversary security metric for network configuration security analysis. In *Proceedings of the 2nd ACM workshop on Quality of protection*, pages 31–38, New York, NY, USA, 2006. ACM Press.
21. C. Phillips and L. Swiler. A graph-based system for network-vulnerability analysis. In *Proceedings of the New Security Paradigms Workshop (NSPW'98)*, 1998.
22. C.R. Ramakrishnan and R. Sekar. Model-based analysis of configuration vulnerabilities. *Journal of Computer Security*, 10(1/2):189–209, 2002.
23. M.K. Reiter and S.G. Stubblebine. Authentication metric analysis and design. *ACM Transactions on Information and System Security*, 2(2):138–158, 5 1999.
24. R. Ritchey and P. Ammann. Using model checking to analyze network vulnerabilities. In *Proceedings of the 2000 IEEE Symposium on Research on Security and Privacy (S&P'00)*, pages 156–165, 2000.
25. O. Sheyner, J. Haines, S. Jha, R. Lippmann, and J.M. Wing. Automated generation and analysis of attack graphs. In *Proceedings of the 2002 IEEE Symposium on Security and Privacy (S&P'02)*, pages 273–284, 2002.
26. M. Swanson, N. Bartol, J. Sabato, J. Hash, and L. Graffo. Security metrics guide for information technology systems. NIST Special Publication 800-55, 2003.
27. L. Swiler, C. Phillips, D. Ellis, and S. Chakerian. Computer attack graph generation tool. In *Proceedings of the DARPA Information Survivability Conference & Exposition II (DISCEX'01)*, 2001.
28. L. Wang, A. Liu, and S. Jajodia. An efficient and unified approach to correlating, hypothesizing, and predicting intrusion alerts. In *Proceedings of the 10th European Symposium on Research in Computer Security (ESORICS 2005)*, pages 247–266, 2005.
29. L. Wang, A. Liu, and S. Jajodia. Using attack graphs for correlating, hypothesizing, and predicting intrusion alerts. *Computer Communications*, 29(15):2917–2933, 2006.
30. L. Wang, S. Noel, and S. Jajodia. Minimum-cost network hardening using attack graphs. *Computer Communications*, 29(18):3812–3824, 11 2006.
31. L. Wang, C. Yao, A. Singhal, and S. Jajodia. Interactive analysis of attack graphs using relational queries. In *Proceedings of 20th IFIP WG 11.3 Working Conference on Data and Applications Security (DBSec 2006)*, pages 119–132, 2006.
32. D. Zerkle and K. Levitt. Netkuang - a multi-host configuration vulnerability checker. In *Proceedings of the 6th USENIX Unix Security Symposium (USENIX'96)*, 1996.