Scalable Security Model Generation and Analysis Using k-importance Measures

Jin B. Hong and Dong Seong Kim

Computer Science and Software Engineering,
University of Canterbury,
Christchurch,
New Zealand
jho102@uclive.ac.nz, dongseong.kim@canterbury.ac.nz

Abstract. Attack representation models (ARMs) (such as attack graphs, attack trees) can be used to model and assess security of a networked system. To do this, one must generate an ARM. However, generation and evaluation of the ARM suffer from a scalability problem when the size of the networked system is very large (e.g., 10,000 computer hosts in the network with a complex network topology). The main reason is that computing all possible attack scenarios to cover all aspects of an attack results in a state space explosion. One idea is to use only important hosts and vulnerabilities in the networked system to generate and evaluate security. We propose to use k-importance measures to generate a two-layer hierarchical ARM that will improve the scalability of model generation and security evaluation computational complexities. We use k_1 number of important hosts based on network centrality measures and k_2 number of significant vulnerabilities of hosts using host security metrics. We show that an equivalent security analysis can be achieved using our approach (using k-importance measures), compared to an exhaustive search.

Keywords: Attack Models, Network Centrality, Security Analysis, Security Metrics.

1 Introduction

Attack representation models (ARMs) (e.g., Attack Graphs (AGs) [1] and Attack Trees (ATs) [2]) are widely used for computer and network security analysis. One main usage of ARMs is to formulate security solutions to enhance the network security while minimising the cost [3, 4]. One of limitations using ARMs is a scalability problem [5], because computing all possible attack scenarios has s state space explosion problem. There are two main types of ARMs; Graph-based and Tree-based. Graph-based (e.g., logical attack graphs [6], multiple prerequisite graphs [7]) and tree-based (e.g., protection trees [8], attack countermeasure trees [9]) ARMs are non-state space models, but graph-based ARMs have an exponential number of possible attack paths in security evaluation. Hierarchical attack representation models (HARMs) [10, 11] have been proposed to improve the scalability of non-state space models, but the scalability problem still remains when evaluating network security of a very large sized network system.

T. Zia et al. (Eds.): SecureComm 2013, LNICST 127, pp. 270–287, 2013.

ARMs have different phases in its lifecycle. The preprocessing phase collects network information (e.g., network reachability, vulnerabilities), and the generation (or construction) phase combines these information to build an ARM. The representation phase stores and visualises the ARM, and the evaluation phase analyses the network security via the ARM. The modification phase updates the ARM when the network changes. Generation of graph-based ARMs is scalable, but the evaluation has a scalability problem. In contrast, the evaluation of tree-based ARMs is scalable only if the size of the ARM has a polynomial size complexity.

Model simplifications (e.g., graph aggregation [12], adjacency matrix clustering [13], graph simplification via collapsing similar nodes [7]) and heuristic methods (e.g., Particle Swarm [14], new heuristic algorithms [15]) are used to improve the scalability, but these methods require an ARM generated specific to their evaluation method using all network information. We propose to use k-importance measure to rank hosts in the network and vulnerabilities of the hosts, and generate ARMs only using those selected hosts and vulnerabilities to improve the scalability of generation and evaluation of ARMs. We use network centrality measures (NCMs) (e.g., degree, closeness, and betweenness [16, 17]) to rank k_1 number of hosts in the network, and security metrics (e.g., Common Vulnerability Scoring System (CVSS) [18], risk [9]) to rank $k_2$number of important vulnerabilities of the hosts.

NCMs identify characteristics of network components based on the structure. The structure of the network is important in a cyber attack, as some attacks (e.g., sequential attacks) progress based on how network components are connected. The reachability information is based on the network structure. As a result, NCMs can distinguish attack paths that are most likely be used in an attack than others. Security metrics reflect characteristics of vulnerabilities in hosts. Security metrics can be measured from real systems [19], cloud systems [20], emulations [21], and honey pots [22].

We propose to use a two-layer HARM to analyse network security [11], to capture network information (e.g., network reachability) in the upper layer and vulnerability information (e.g., CVSS) in the lower layer. NCMs are used on the upper layer to rank important hosts in the network, and security metrics on the lower layer to rank important vulnerabilities in the hosts.

The contributions of this paper are summarised as follows:

- To propose an efficient ARM generation method based on k-importance measures;
- To generate HARMs using k-importance measures and show that nearly equivalent security analysis can be achieved;
- To simulate scalability and accuracy of the HARMs (or ARMs) using k-importance measures against the exhaustive search method when analysing the network security.

The rest of the paper is as organised as follows. In Section 2, related work is given. In Section 3, an example network and HARMs are described in the phases of generation and evaluation, with an evaluation example based on risk analysis. Simulation results are shown in Section 4. Discussion is given in Section 5, and Section 6 concludes this paper.

2 Related Work

Security analysis using all possible attack scenarios can cover all set of known attacks. Various modelling techniques are proposed to improve the scalability of ARMs [6, 7, 10, 11], but computing all possible attack scenarios (e.g., full AG [1], attack response trees [23]) for a large sized network still suffers from a scalability problem [5]. Model simplifications and heuristic methods are widely used to improve scalability in the evaluation phase, but not in the generation phase.

Attack scenarios are often used to generate ARMs [23]. Chen *et al*. [24] used *compact* AG, similar to a logical AG in [6], to find n-valid paths that has less than n steps to reach the target, where n denotes the number of stepping stone hosts in the network. Further, they defined a *weighted-greedy* algorithm to find the optimal security solution in the evaluation phase. Their experiment results clearly showed that covering a larger set of attack scenarios is computationally expensive. Mehta *et al*. [25] ranked AG components based on attack probabilities. A full AG is constructed (i.e., capturing all possible attack scenarios [1]), which requires computing exponential number of attack paths in a large sized network. AG components cannot be ranked until full AG is generated. Sawilla *et al*. [26] used partial cuts in evaluation of AGs. Partial cuts divide network components by their importance, such that the relevance between a network component and the attack is decided based on the generated AG. However, the structure of the AG is heavily dependent on network reachability. That is, network structure affects how important network components are chosen in the AG. Various techniques (e.g., model simplification and heuristic methods) are proposed to improve scalability in the evaluation phase, but they did not consider reducing the computation complexity in the generation phase.

Importance measures are used in some application domains. Cadini *et al*. [17] used NCMs to capture strengths and weaknesses of network safety. Georgiadis *et al*. [16] described network security using NCMs, but only implications of NCMs are described. Gallon *et al*. [18] integrated CVSS framework with AGs to construct an AG, but the structure of the ARM is the same with other AGs. Previous works using NCMs and security metrics to assess the performance network security were applied only in the evaluation phase of ARMs.

We propose k-importance measures in the generation phase of ARMs using important hosts and vulnerabilities. To the best of our knowledge, no other work considered this approach to improve scalability. We show that our approach can provide nearly equivalent security analysis of a large sized networked system in a scalable manner. That is, by using only a subset of network components, it reduces the computational complexity in all phases of ARMs lifecycle. Important hosts are chosen based on NCMs (e.g., degree, closeness, betweenness [16, 17]), and important vulnerabilities are chosen based on security metrics (e.g., CVSS [18], structural importance [9]). Generation and evaluation of ARMs using k-importance measures and their computation methods are presented in Section 3.

3 A Network and Attack Models

Analysing network security follows procedural steps; (i) gather information from the network (e.g., reachability, vulnerabilities), (ii) generate an ARM using given network information, (iii) analyse network security via ARM using security metrics, and (iv) update the ARM if there is any change in the network. Figure 1 shows the steps taken to analyse the network security, with additional feature to use *k*-importance measures. Computations of *k*-importance measures are processed at the beginning of the generation phase.

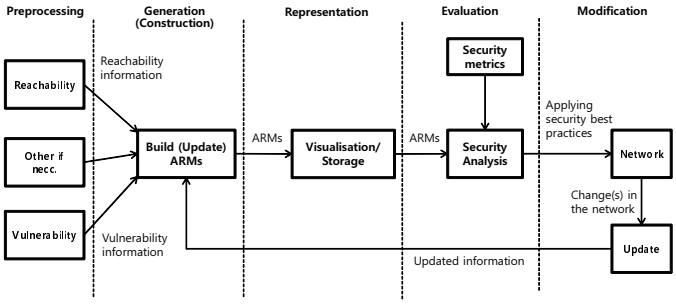


Fig. 1. Lifecycle of an ARM

3.1 Network Settings and the Attack Scenario

We use a network example as shown in Figure 2. We assume all host connections have the same cost (e.g., same throughput and attack probabilities), and the probability of attacks are the same for all hosts. However, different edge costs can be modelled with different edge weights, and also the probability of attacks can hold different values. H_1, H_2, H_3 and H_4 (intermediate host machines) are identical hosts with same vulnerabilities, and H_5 (a target machine) is running a virtual machine. We define an attack scenario with the location of an attacker outside the network (i.e., attack from Internet), and the goal is to compromise the target host (i.e., obtaining the administrator privilege of H_5).

Vulnerability information is collected from a real system using vulnerability scanners (e.g., *NESSUS* [27]). On the intermediate host machines (based on Windows XP SP1), about 60 known vulnerabilities are found. Some vulnerability information is not given (e.g., *NESSUS* plugin name, port number, CVSS BS, and Common Vulnerabilities and Exposures (CVE) ID). For automated generation of ARMs, it is difficult to interpret vulnerability description to model vulnerability interactions (e.g., difficulties in processing manual input of vulnerability information due to various language formats, such as grammar and choice of words). Therefore, we scope the list of known vulnerabilities only with identifications (e.g., CVE ID is given). Total 11 vulnerabilities are identified as shown in Table 1 with details including CVE ID, CVSS BS, impact, exploitability, confidentiality impact (CI) and access level. All vulnerabilities are accessible via the network (i.e., no local access is required) and no authentications are

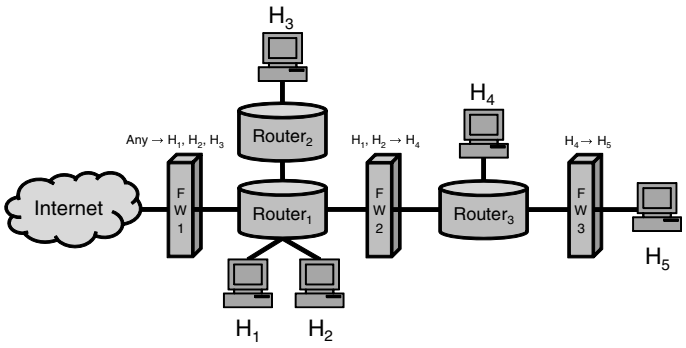


Fig. 2. A network example

required to exploit vulnerabilities. The CI is categorised into *None* (denoted as *N*), *Partial* (denoted as *P*), or *Complete* (denoted as *C*), where *None* means no information on the machine is accessible, *Partial* means a considerable amount of information could be accessible, and *Complete* means all information of the system is compromised, by the attacker respectively. The access level describes the privilege acquired by the attacker, where only a single vulnerability allowed the attacker to gain the administrator privilege.

Table 1. List of vulnerabilities in H_1-H_4

ID	CVE ID	CVSS BS	Impact	Exploitability	CI	Access Level	Authentication
v_1	CVE-2005-1794	6.4	4.9	10.0	P	None	None
v_2	CVE-2011-0661	10.0	10.0	10.0	C	None	None
v_3	CVE-2010-0231	10.0	10.0	10.0	C	None	None
v_4	CVE-2011-2552	7.8	6.9	10.0	N	None	None
v_5	CVE-1999-0520	6.4	4.9	10.0	P	None	None
v_6	CVE-2010-2729	9.3	10.0	8.6	C	None	None
v_7	CVE-1999-0505	7.2	10.0	3.9	C	Admin	None
v_8	CVE-2002-1117	5.0	2.9	10.0	P	None	None
v_9	CVE-2003-0386	4.3	2.9	8.6	P	None	None
v_{10}	CVE-2010-0025	5.0	2.9	10.0	P	None	None
v_{11}	CVE-1999-0497	0.0	0.0	10.0	N	None	None

A total of 23 vulnerabilities are found on the target machine (i.e., H_5 running a VMware ESXi). We assume that the attacker can only access network hosts via remote access (i.e., no physical access), then there are 11 vulnerabilities accessible via remote connections. The list of vulnerabilities of H_5 is shown in Table 2. In the authentication field, some vulnerabilities (CVE-2010-1142, CVE-2010-1141 and CVE-2008-2097) require a *SingleSystem* condition satisfied. These vulnerabilities require the attacker to have an access, such as command line, desktop session or web interface on the machine. Any vulnerability with disclosure of machine information (i.e., CI is

either P or C) allows the attacker to gain an access to the machine (i.e., *SingleSystem* authentication is satisfied). There are vulnerabilities without any CI (i.e., CI is N), but they allow the attacker to distribute softwares (e.g., Trojan horse) that could be used to gain access of network hosts. However, we do not consider these cases.

Table 2. List of vulnerabilities in H_5

ID	CVE ID	CVSS BS	Impact	Exploitability	CI	Access Level	Authentication
v_{12}	CVE-2011-1789	5.0	2.9	10.0	N	None	None
v_{13}	CVE-2011-1786	5.0	2.9	10.0	N	None	None
v_{14}	CVE-2011-1785	7.8	6.9	10.0	N	None	None
v_{15}	CVE-2011-0355	7.8	6.9	10.0	N	None	None
v_{16}	CVE-2010-4573	9.3	10.0	8.6	C	None	None
v_{17}	CVE-2010-3609	5.0	2.9	10.0	N	None	None
v_{18}	CVE-2010-1142	8.5	10.0	6.8	C	None	Single System
v_{19}	CVE-2010-1141	8.5	10.0	6.8	C	None	Single System
v_{20}	CVE-2009-3733	5.0	2.9	10.0	P	None	None
v_{21}	CVE-2008-4281	9.3	10.0	8.6	C	None	None
v_{22}	CVE-2008-2097	9.0	10.0	8.0	C	Admin	Single System

3.2 Computing k-importance Measures

In this subsection, we describe how to rank and select important hosts and vulnerabilities based on k-importance measures. k_1 denotes the number of important hosts and k_2 denotes the number of important vulnerabilities used to generate ARMs, respectively.

Ranking k_1 Number of Important Hosts. We use network reachability information in conjunction with NCMs to rank important hosts. Among many NCMs, we use only basic NCMs (e.g., degree, closeness and betweenness centrality measures) [17]. The degree centrality computes the popularity of a node (e.g., a host in a network graph) based on number of direct connections with other nodes (e.g., single-hop neighbour hosts), with its computational complexity of O(n) where n is the number of nodes in the graph. The closeness centrality computes the distance of a node to all other nodes, with its computational complexity of O(n^3) using Floyd algorithm [28]. The betweenness centrality computes the significance of a node between all node pairs, with its computational complexity of O(n^3) using Floyd algorithm. A problem using NCMs is when two or more nodes have the same rank. In this work, nodes with the same rank will be assigned with the same rank, and we will consider other approaches in future work. The normalised NCMs of the example network is shown in Table 3, where high values represent higher importance. Each NCM ranks are combined to give the overall rankings of the hosts. The final rank is determined by taking into account their ranks from each NCMs. A node with high scores in all three NCMs will have a higher rank (e.g., H_4 has highest score in all NCMs) than other nodes. *RankSum* values are calculated by adding up their ranks from each NCM (e.g., *RankSum* of H_1 is calculated by adding values of 1 (first equally important based on degree centrality), 1 (first equally important based on

closeness centrality), and 2 (second equally important based on betweenness centrality) resulting in value of 4). Then, the *RankSum* values are used to give the final rank of each host in an ascending order. Since H_1 and H_2 have the same value of *RankSum*, they are equally ranked overall.

Table 3. Network Centrality Measurements

	Degree	Closeness	Betweenness	*RankSum*	Final Rank
H_1	3/4	4/5	8/12	4	2
H_2	3/4	4/5	8/12	4	2
H_3	2/4	4/7	4/12	12	4
H_4	3/4	4/5	10/12	3	1
H_5	1/4	4/12	4/12	14	5

Another importance of using NCMs is to compute the proportion of important hosts (i.e., the value of k_1). The density of the network (i.e., the proportion of host interconnections in respect to the number of hosts) is one of the important factors because the number of available attack paths are proportional to the network density (i.e., there are more available attack paths in a dense network (e.g., fully connected or mesh topologies) than a sparse network (e.g., star or tree topologies)). We use closeness centrality to compute the value for k_1, because the closeness centrality directly measures the amount of host connections in the network. In a fully connected network, the sum of normalised degree centrality measure is equal to the number of hosts in the network (i.e., all network components are used in at least one attack path). The sum of degree centrality is three (out of five), which we will assign as the value of k_1 (i.e., $k_1 = 3$).

3.3 Ranking k_2 important Number of Vulnerabilities in Hosts

Various security metrics evaluate different aspects of vulnerabilities. Values are assigned to security metrics (e.g., CVSS base score (BS) [18]) and these values are relative to each other. For our example, we use CVSS BS to rank vulnerabilities. The rank based on CVSS BS is shown in Table 4 and Table 5 for intermediate hosts and target host vulnerabilities, respectively. The proportion of important vulnerabilities is chosen by their CVSS BSs. The average CVSS BS is calculated, and vulnerabilities with CVSS BS higher than the average are selected. We determine the k_2 values with threshold values set to the average CVSS BSs. The average CVSS BSs are 6.5 and 7.3 for intermediate hosts and the target host, respectively. Therefore, in this example, the value of $k_2^{host} = 5$, and the value of $k_2^{target} = 7$.

3.4 Generation of HARMs

Generation of HARM Using All Network Information. We use a two-layer HARM to analyse security of the system. We generate the HARM in which an AG in the upper layer (e.g., using ARM generating tools, such as MulVAL [29]), and an AT in the lower

Table 4. Intermediate host vulnerability rankings

	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8	v_9	v_{10}	v_{11}
CVSS BS	6.4	10.0	10.0	7.8	6.4	9.3	7.2	5.0	4.3	5.0	0.0
Rank	6	1	1	4	6	3	5	8	10	8	11

Table 5. target host vulnerability rankings

	v_{12}	v_{13}	v_{14}	v_{15}	v_{16}	v_{17}	v_{18}	v_{19}	v_{20}	v_{21}	v_{22}
CVSS BS	5.0	5.0	7.8	7.8	9.3	5.0	8.5	8.5	5.0	9.3	9.0
Rank	8	8	6	6	1	8	4	4	8	1	3

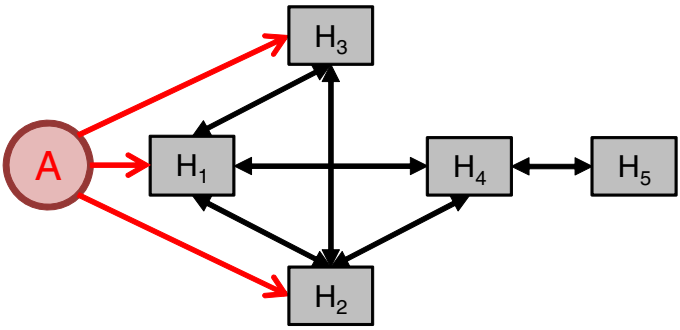
layer are used. The HARM of the example network is shown in Figure 3, where the attacker is denoted as A. H_1, H_2, H_3 and H_4 have identical lower layer ATs (i.e., the AT goal is defined as *Compromise Host*). We assume only one prerequisite condition is required when exploiting vulnerabilities with prerequisites (i.e., one vulnerability is chosen from a set of vulnerabilities satisfying the same condition).

Generation of a Reduced HARM Using k-importance Measures We generate a reduced HARM based on k-importance measures, denoted as ReHARM as shown in Figure 4. The size difference is significantly reduced in comparison to the HARM shown in Figure 3, even with the small network example. The selected important hosts with $k_1 = 3$ are H_1, H_2, H_4. Since H_5 is designated as the target, it must be included in the upper layer. If we want to assess security of the network system, only selected hosts are included in the AG model. The number of selected important vulnerabilities is $k_2^{*Host} = 5$ and $k_2^{*Target} = 7$ for intermediate hosts and the target host, respectively. It is also possible to generate a full HARM, then take into account important hosts and vulnerabilities to derive a ReHARM, which could be regarded as an abstract interpretation. However, it is an unnecessary step to generate the ReHARM because it can be derived directly from the preprocessing phase (i.e., with given network and vulnerability information).

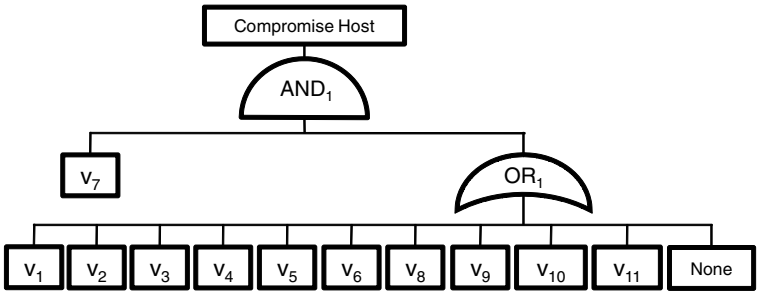
3.5 Security Evaluation

We analyse the risk associated with each attack path using the HARM with all details (e.g., a full HARM), denoted by R_{ap}. The computation of the risk is shown in equation (1) [9], where P_{goal} is the probability of an attack, and I_{goal} is the impact value. Note that it is possible to apply different security analysis by adopting different methods or even different ARM in the lower level of the ReHARM. To compute the risk, we use impact values directly from Table 1 and Table 2, and the exploitability is scaled by a factor of 10, to represent the probability of an attack. The exploitability value is computed by taking into account access vector, access complexity and authentication of vulnerability, which reflect characteristics of attack probabilities.

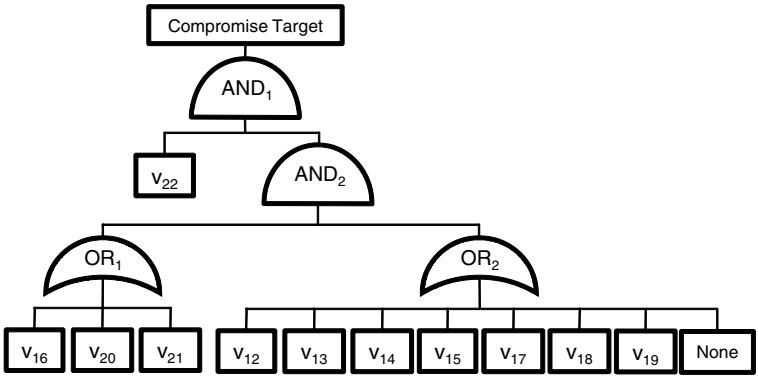
$$R_{ap} = P_{goal} \times I_{goal} \tag{1}$$



(a) An AG in the upper layer

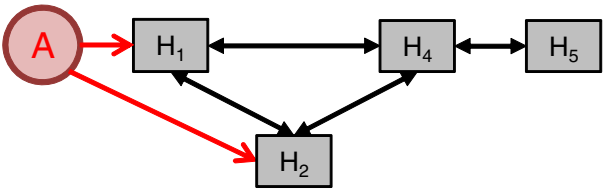


(b) An AT of intermediate hosts in the lower layer

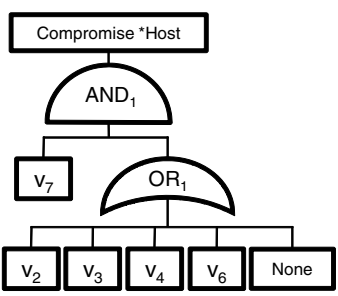


(c) An AT of the target host in the lower layer

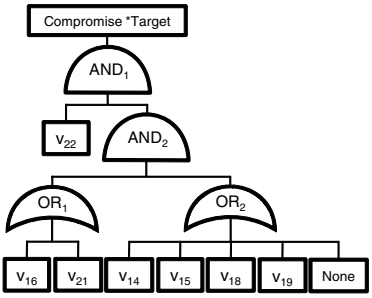
Fig. 3. The HARM of the example network



(a) A reduced AG in the upper layer



(b) A reduced AT of intermediate hosts in the lower layer



(c) A reduced AT of the target host in the lower layer

Fig. 4. The ReHARM of the example network

Risk Computation Using the HARM. First, we compute P_{host} (i.e., probability of an attack on intermediate hosts), as shown in equation (2). Note that $P_{None} = 1$ (i.e., not choosing to exploit a vulnerability from the list has a probability of one).

$$\begin{aligned} P_{host} =& P_{v_7} \times P_{host_{OR_1}} \\ =& 0.39 \times [1 - ((1 - P_{v_1}) \times (1 - P_{v_2}) \times (1 - P_{v_3}) \times (1 - P_{v_4}) \times (1 - P_{v_5}) \\ & \times (1 - P_{v_6}) \times (1 - P_{v_8}) \times (1 - P_{v_9}) \times (1 - P_{v_{10}}) \times (1 - P_{v_{11}}) \times (1 - P_{None}))] \\ =& 0.39 \end{aligned} \tag{2}$$

Now we compute I_{host} as shown in equation (3). Note that $I_{None} = 0$ (i.e., not exploit a vulnerability has no impact).

$$\begin{aligned} I_{host} =& I_{v_7} + I_{host_{OR_1}} \\ =& 10.0 + max(I_{v_1}, I_{v_2}, I_{v_3}, I_{v_4}, I_{v_5}, I_{v_6}, I_{v_8}, I_{v_9}, I_{v_{10}}, I_{v_{11}}, I_{None}) \\ =& 10.0 + 10.0 \\ =& 20.0 \end{aligned} \tag{3}$$

Similarly, we can compute $P_{target} = 0.8$ and $I_{target} = 30.0$.

Now, we compute all possible attack paths, based on the HARM shown in Figure 3 using exhaustive search. The list of all possible attack paths and their risk analysis are summarised in Table 6. Each attack path is presented with sequences of the hosts. We observe the highest risk value is 8.52 (from paths pa_3 and pa_6).

Table 6. Risk analysis of attack paths

Path Number	Attack path	P_{goal}	I_{goal}	R_{ap}
pa_1	$H_1H_2H_4H_5$	0.047	90.0	4.27
pa_2	$H_1H_3H_2H_4H_5$	0.019	110.0	2.04
pa_3	$H_1H_4H_5$	0.122	70.0	8.52
pa_4	$H_2H_1H_4H_5$	0.047	90.0	4.27
pa_5	$H_2H_3H_1H_4H_5$	0.019	110.0	2.04
pa_6	$H_2H_4H_5$	0.122	70.0	8.52
pa_7	$H_3H_1H_2H_4H_5$	0.019	110.0	2.04
pa_8	$H_3H_1H_4H_5$	0.047	90.0	4.27
pa_9	$H_3H_2H_1H_4H_5$	0.019	110.0	2.04
pa_{10}	$H_3H_2H_4H_5$	0.047	90.0	4.27

Risk Analysis Using the ReHARM. We show risk analysis using ReHARM. The same calculation as the risk analysis of the HARM is used. We denote the risk analysis using ReHARM as R^*_{ap}. First, we compute P^*_{host} as shown in equation (4).

$$\begin{aligned} P^*_{host} =& P_{v_7} \times P^*_{host_{OR_1}} \\ =& 0.39 \times [1 - ((1 - P_{v_2}) \times (1 - P_{v_3}) \times (1 - P_{v_4}) \times (1 - P_{v_6}) \times (1 - P_{None}))] \\ =& 0.39 \times [1 - 0] \\ =& 0.39 \end{aligned} \tag{4}$$

Now we compute I^*_{host}, as shown in equation (5).

$$\begin{aligned} I^*_{host} =& I_{v_7} + I^*_{host_{OR_1}} \\ =& 10.0 + max(I_{v_2}, I_{v_3}, I_{v_4}, I_{v_6}, I_{None}) \\ =& 10.0 + 10.0 \\ =& 20.0 \end{aligned} \tag{5}$$

Similarly, we can compute $P^*_{target} = 0.7776$ and $I^*_{target} = 30.0$.

Based on the ReHARM as shown in Figure 4, we compute all possible attack paths using exhaustive search. Table 7 shows the risk analysis based on ReHARM. The highest risk value is 8.28 (from paths pa^*_2 and pa^*_4), which is nearly equivalent to the risk analysis using the HARM shown in Table 6.

4 Simulation Results

We conduct simulations to investigate the effectiveness of security analysis using k-importance measures. Figure 5 shows the example network used for simulations. We were not able to use a real system with a large number of hosts to show the scalability of our proposed approach. The network consisted of 1000 hosts. 500 hosts were assigned in the DMZ network, 500 hosts were assigned in the Internal network, and one

Table 7. Risk analysis of attack paths using ReHARM

Path Number	Attack path	P^*_{goal}	I^*_{goal}	R^*_{ap}
pa^*_1	$H_1H_2H_4H_5$	0.046	90.0	4.15
pa^*_2	$H_1H_4H_5$	0.118	70.0	8.28
pa^*_3	$H_2H_1H_4H_5$	0.046	90.0	4.15
pa^*_4	$H_2H_4H_5$	0.118	70.0	8.28

target host was assigned in the Database network. Firewalls, denoted as FW_1 and FW_2, controls the data flow in the network, restricting access to the Internal and Database networks from outside. The attack scenario is for an attacker located outside the network to compromise the target host. All hosts were assigned with 10 vulnerabilities, where a single vulnerability (v_{root}) granted the admin privilege when exploited, two vulnerabilities (v^1_{user} and v^2_{user}) granted the user privilege, and the rest does not change the privilege status. To exploit v_{root}, the attacker must exploit either v^1_{user} or v^2_{user}. There is no restriction to exploit all other vulnerabilities. We use Intel(R) Core(TM)2 Quad CPU @ 2.66GHz with 3.24 GB of RAM on a Windows XP SP3 machine, and the simulation was coded in Python.

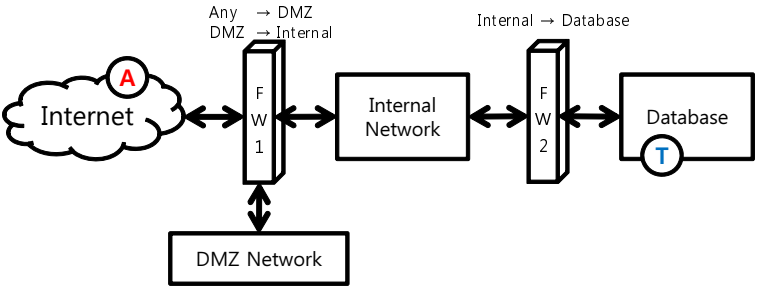


Fig. 5. An example network for simulation

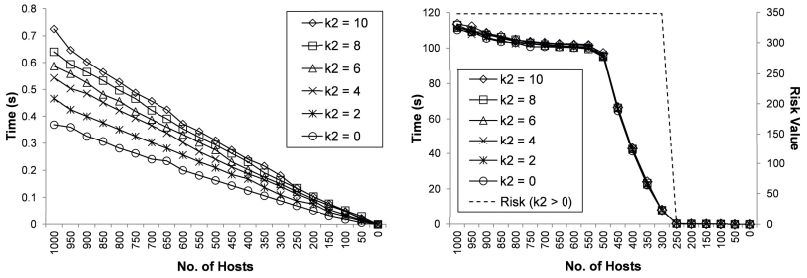
4.1 Security Analysis Based on Host Importance

Risk of the network system is analysed, where different vulnerabilities are assigned with difference impact values chosen reasonably and randomly (v_{root} with 10, v^1_{user} and v^2_{user} with 5, and the rest with 1). We assume that the probability of an attack success on all vulnerabilities is one, but we can assign real probability values as in Section 3. We compare security analysis using exhaustive search and k-importance measures. We use degree centrality measures to rank k_1 important hosts. First, we consider all network hosts to compute the risk of the example network in Figure 5. Then, we continuously compute the risk value by generating the ReHARM by varying values of k_1. As the number of hosts modelled reduces, generation and evaluation times reduces although equivalent risk analysis is kept. The simulation result when $k_2 = 10$ is shown in Table 8. Generation and evaluation times are shown in Figure 6.

Table 8. Security analysis using k_1 values ($k_2 = 10$)

No. of hosts	Generation time (s)	Evaluation time (s)	Risk value	No. of attack paths
1000	0.725	113.515	348	55942475
900	0.603	108.358	348	55942475
800	0.528	104.626	348	55942475
700	0.456	102.873	348	55942475
600	0.372	101.780	348	55942475
500	0.309	96.998	348	5699925
400	0.241	42.796	348	3274425
300	0.181	8.172	348	848925
200	0.103	0.250	0	0
100	0.047	0.172	0	0
0	0.0	0.0	0	0

The density of simulation network is 0.006 (i.e., each host on average has a direct connection to six other hosts). The simulation result shows that the risk value is still equivalent when the network size has reduced by 70% (i.e., $k_1 = 300$). The generation time consistently reduces as the number of hosts modelled decreases as shown in Figure 6(a). The evaluation time decreases steadily down to 50% of hosts modelled. When the number of attack paths reduced, the evaluation time decreases rapidly. When the number of hosts modelled are reduced to 250, the hosts directly affecting the risk analysis are removed, such that the risk output is misleading. This is shown in Figure 6(b). Also, we can observe that changing number of vulnerabilities (i.e., k_2) does not affect the scalability of the evaluation phase. The optimal solution using degree centrality measures was found at $k_1 = 266$.



(a) Generation time using k_1 values (b) Evaluation time using k_1 values

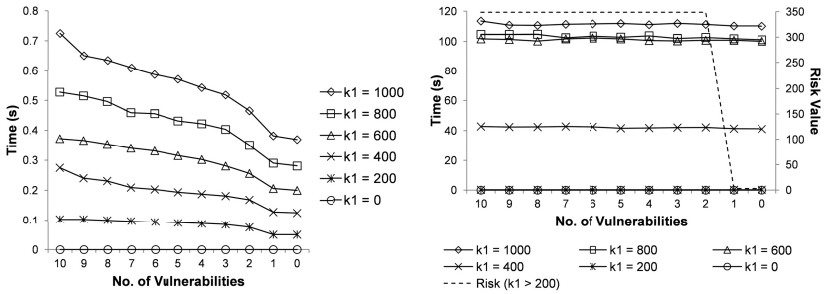
Fig. 6. Performance of security analysis using k_1 values

4.2 Security Analysis Based on Vulnerability Importance

We rank vulnerabilities with given impact value information. All network hosts are modelled to investigate the performance of risk analysis when k_2 number of important vulnerabilities are considered in the risk analysis. The simulation result is shown in Table 9. Generation and evaluation times are shown in Figure 7.

Table 9. Security analysis using k_2 values ($k_1 = 1000$)

No. of Vulnerabilities	Generation time (s)	Evaluation time (s)	Risk value
10	0.725	113.515	348
9	0.650	110.796	348
8	0.634	110.576	348
7	0.609	111.343	348
6	0.587	111.608	348
5	0.572	111.858	348
4	0.544	111.108	348
3	0.519	111.920	348
2	0.466	111.218	348
1	0.381	110.264	3
0	0.369	110.233	3



(a) Generation time using k_2 values (b) Evaluation time using k_2 values

Fig. 7. Performance of security analysis using k_2 values

Table 10. Naive and optimal solution comparison

	Generation time (s)	Evaluation time (s)	Risk value	No. of attack paths
Naive	0.725	113.515	348	5942475
Optimal	0.097	0.578	348	24255

The generation time shows constant improvement, because there are a few numbers of components to generate in the lower layer. However, there are no improvements shown in the evaluation time for all k_1 values. It shows that k_2 values do not affect the performance of evaluation. If vulnerability models become more complex (i.e., multiple paths in exploiting vulnerabilities), the computational complexity of lower layer will also increase. The optimal solution is found when $k_2 = 2$. The naive solution compared to the optimal solution is shown in Table 10. The optimal solution shows approximately 87% generation time improvement and 99.5% evaluation time improvement respectively. We will investigate to find optimal k_1 and k_2 values in our future work.

5 Discussion

We used k-importance measures to generate a ReHARM. A risk analysis showed that an equivalent security solution can be achieved, while the size of the HARM is significantly reduced. Accuracy and performances of security analysis using k-importance measures are investigated through simulation. The NCM (e.g., closeness centrality) effectively ranked important hosts based on the network topology, and nearly equivalent risk value is computed. Moreover, the time performance was also improved for generating and evaluating the HARM.

However, the security analysis of the example system showed that not all vulnerabilities associated security metrics. Also, a single security metric often does not capture various effects of vulnerabilities (e.g., high CVSS BS, but low structural importance), and other requirements to satisfy the success of an attack are not well defined (e.g., privilege requirements). Network topologies and attack goals determine which hosts in a network are important in an event of an attack. The proportion of the network hosts unused in an attack also depends on the network density (e.g., a sparse network and a dense network), such that determining the value of k_1 is difficult. In addition, an attacker located inside the network reduces the coverage of the network, but NCMs cover all network hosts. Lastly, attack on less important hosts and vulnerabilities are not properly addressed.

5.1 Vulnerabilities without Security Metrics

Using a vulnerability scanner *NESSUS* [27], about 60 vulnerabilities of a real host machine were reported. However, only 11 vulnerabilities had CVSS BS, which gives set of security metrics associated with these vulnerabilities. There were textual description of vulnerabilities (e.g., Vulnerability Synopsis and Vulnerability Description), but this is difficult to process automatically. Moreover, 10 vulnerabilities are scanned without any descriptions. Incomplete security data makes difficult to automate and analyse the network security. Other sources of vulnerability scanners and security metrics will be investigated in our future work.

5.2 Categorised Vulnerability Ranking

A single security metric cannot capture all aspects of vulnerabilities. The attack goal changes how vulnerabilities will be exploited by an attacker. For example, an attack goal of gaining the admin privilege defines a subset of vulnerabilities that must be exploited to achieve this, but an attack goal to hijack a communication of a host defines a different subset of vulnerabilities to achieve this goal. So, there needs a definition of vulnerability categories that satisfy different attack goals. Then vulnerabilities can be ranked within each subset. In addition, the ranking of vulnerabilities can be combined from vulnerability rankings based on various security metrics, such as CVSS BS rankings and structural importance rankings [9]. Improvements and effectiveness of categorising and combining vulnerability rankings should be further investigated.

5.3 Network Features for k_1 Selection

The network topology defines possible attack scenarios. A dense network (e.g., fully connected or mesh network topologies) allows the attacker to take many different attack paths to reach the target, so that a large proportion of network hosts will be used in at least one attack path. In contrast, a sparse network (e.g., star or tree network topologies) limits the number of attack paths, and the number of unused hosts (i.e., hosts that does not benefit the attacker in any attack scenario) increases. Therefore, the value of k_1 will depend on the attack scenario and the network topology. The effect of different network topologies for determining the value of k_1 importance measures needs to be studied, and the relationship between the number of hosts and the value of k_1 for the same network topology needs to be taken into account.

5.4 Modelling Attackers Located Inside the Network

Ranking important hosts using NCMs incorporates only the reachability of network hosts. However, an attacker located inside the network allows the attacker to bypass some of the network security (e.g., firewalls). Compared with an attack from outside the network, the scope of an attack is much smaller (i.e., only a subset of network hosts are considered) because the distance to the target is much closer. In such case, the ranking of important hosts using NCMs is inaccurate, because all network hosts are considered in NCMs. Therefore, one needs to consider an effective method to rank important hosts accurately for such attack scenarios.

5.5 Attack on Less Important Hosts and Vulnerabilities

By enforcing security only on important hosts and vulnerabilities allow attackers to exploit less important hosts and vulnerabilities, and security analysis based on important hosts and vulnerabilities cannot capture such attacks. One solution is that if all attack paths are covered with selected set of hosts and vulnerabilities, then any attack scenarios, regardless of using important or less important hosts and vulnerabilities, are covered. However, a naive approach to check the coverage of attack paths is computationally expensive (i.e., exponential number of attack paths need to be checked). More efficient method to cover all attack paths with a set of hosts and vulnerabilities will be studied in our future work.

6 Conclusion

Security analysis using the ARMs allows users and system administrators to become aware of vulnerable network components and configurations, and security solutions can be enforced or suggested to enhance the network security. However, existing ARMs have a scalability problem when the network size becomes large. Generating an ARM requires all the network information, and simplifications and heuristic methods are used in evaluation to improve the scalability. That is, not all network information is required for security analysis. Therefore, we proposed to use k-importance measures to improve

generation and evaluation of ARMs. $k_1$number of important hosts and k_2 number of important vulnerabilities are ranked and selected to generate an ARM (e.g., a two-layer HARM) and to evaluate the network security. We described methods to rank important hosts using NCMs and vulnerabilities using security metrics. We showed equivalent security solutions can be achieved using k-importance measures, while the performances improved in both generation and evaluation in terms of time and computation requirements. We also showed that time and computation requirements can be optimised by selecting appropriate number of important hosts and vulnerabilities, which showed a significant improvement compared to the exhaustive search method.

References

1. Sheyner, O., Haines, J., Jha, S., Lippmann, R., Wing, J.: Automated generation and analysis of attack graphs. Technical report, CMU (May 2002)
2. Schneier, B.: Secrets and Lies: Digital Security in a Networked World. John Wiley and Sons Inc. (2000)
3. Albanese, M., Jajodia, S., Noel, S.: Time-efficient and cost-effective network hardening using attack graphs. In: Proc. of Dependable Systems and Networks (DSN 2012). IEEE Computer Society, Los Alamitos (2012)
4. Roy, A., Kim, D., Trivedi, K.: Scalable optimal countermeasure selection using implicit enumeration on Attack Countermeasure Trees. In: Proc. of Dependable Systems and Networks (DSN 2012). IEEE Computer Society, Los Alamitos (2012)
5. Lippmann, R., Ingols, K.: An Annotated Review of Past Papers on Attack Graphs. ESC-TR-2005-054 (2005)
6. Ou, X., Boyer, W., McQueen, M.: A scalable approach to attack graph generation. In: Proc. of ACM Conference on Computer and Communications Security (CCS 2006). ACM (2006)
7. Ingols, K., Lippmann, R., Piwowarski, K.: Practical attack graph generation for network defense. In: Proc. of Computer Security Applications Conference, ACSAC 2006 (2006)
8. Edge, K.: A Framework for Analyzing and Mitigating the Vulnerabilities of Complex Systems via Attack and Protection Trees. PhD thesis, Air Force Institute of Technology (2007)
9. Roy, A., Kim, D., Trivedi, K.: Attack Countermeasure Trees (ACT): towards unifying the constructs of attack and defense trees. Security and Communication Networks 5(8) (2012)
10. Xie, A., Cai, Z., Tang, C., Hu, J., Chen, Z.: Evaluating network security with two-layer attack graphs. In: Proc. of Computer Security Applications Conference, ACSAC 2009 (2009)
11. Hong, J., Kim, D.: HARMs: Hierarchical Attack Representation Models for Network Security Analysis. In: Proc. of the 10th Australian Information Security Management Conference on SECAU Security Congress, SECAU 2012 (2012)
12. Noel, S., Jajodia, S.: Managing attack graph complexity through visual hierarchical aggregation. In: Proc. of the 2004 ACM Workshop on Visualization and Data Mining for Computer Security (VizSec 2004), pp. 109–118. ACM (2004)
13. Noel, S., Jajodia, S.: Understanding complex network attack graphs through clustered adjacency matrices. In: Proc. of the 21st Annual Computer Security Applications Conference (ACSAC 2005), pp. 160–169 (2005)
14. Abadi, M., Jalili, S.: A particle swarm optimization algorithm for minimization analysis of cost-sensitive attack graphs. The ISC International Journal of Information Security (ISeCure 2010) 2(1), 13–32 (2010)
15. Islam, T., Wang, L.: A Heuristic Approach to Minimum-Cost Network Hardening Using Attack Graph. In: Proc. of New Technologies, Mobility and Security, NTMS 2008 (2008)

16. Georgiadis, G., Kirousis, L.: Lightweight centrality measures in networks under attack. Complexus 3(1), 147–157 (2006)
17. Cadini, F., Zio, E., Petrescu, C.-A.: Using centrality measures to rank the importance of the components of a complex network infrastructure. In: Setola, R., Geretshuber, S. (eds.) CRITIS 2008. LNCS, vol. 5508, pp. 155–167. Springer, Heidelberg (2009)
18. Gallon, L., Bascou, J.: Using CVSS in Attack Graphs. In: Proc. of the Sixth International Conference on Availability, Reliability and Security (ARES 2011), pp. 59–66 (2011)
19. Sharma, A., Kalbarczyk, Z., Barlow, J., Iyer, R.: Analysis of security data from a large computing organization. In: Proc. of Dependable Systems Networks, DSN 2011 (2011)
20. Zhu, Y., Hu, H., Ahn, G., Huang, D., Wang, S.: Towards temporal access control in cloud computing. In: Proc. of Annual IEEE International Conference on Computer Communications (INFOCOM 2012), pp. 2576–2580 (2012)
21. Mirkovic, J., Benzel, T., Faber, T., Braden, R., Wroclawski, J., Schwab, S.: The DETER project: Advancing the science of cyber security experimentation and test. In: Proc. of IEEE International Conference on Technologies for Homeland Security (HST 2010), pp. 1–7 (2010)
22. Alata, E., Nicomette, V., Kaaniche, M., Dacier, M., Herrb, M.: Lessons learned from the deployment of a high-interaction honeypot. In: Proc. of Sixth European Dependable Computing Conference (EDCC 2006), pp. 39–46 (October 2006)
23. Zonouz, S., Khurana, H., Sanders, W., Yardley, T.: RRE: A game-theoretic intrusion Response and Recovery Engine. In: Proc. of IEEE/IFIP International Conference on Dependable Systems Networks (DSN 2009), pp. 439–448 (2009)
24. Chen, F., Liu, D., Zhang, Y., Su, J.: A scalable approach to analyzing network security using compact attack graphs. Journal of Networks 5(5) (2010)
25. Mehta, V., Bartzis, C., Zhu, H., Clarke, E., Wing, J.: Ranking attack graphs. In: Zamboni, D., Kruegel, C. (eds.) RAID 2006. LNCS, vol. 4219, pp. 127–144. Springer, Heidelberg (2006)
26. Sawilla, R., Skillicorn, D.: Partial cuts in attack graphs for cost effective network defence. In: Proc. of IEEE Conference on Technologies for Homeland Security, HST 2012 (2012)
27. Beale, J., Deraison, R., Meer, H., Temmingh, R., Walt, C.: The NESSUS project. Syngress Publishing (2002)
28. Floyd, R.: Algorithm 97: Shortest path. Commun. ACM 5(6), 345 (1962)
29. Ou, X., Govindavajhala, S.: Mulval: A logic-based network security analyzer. In: Proc. of the 14th USENIX Security Symposium (USENIX Security 2005), pp. 113–128 (2005)