This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE ICC 2011 proceedings

A Novel Comprehensive Network Security Assessment Approach

Chunlu Wang, Yancheng Wang, Yingfei Dong, and Tianle Zhang

***Abstract*—**Network security assessment is critical to the survivability and reliability of distributed systems. In this paper, we propose a novel assessment approach that supports automatic attack graph generation based on our correlated vulnerability database and quantitative vulnerability assessment utilizing Bayesian attack graphs. We also build a customized vulnerability scanner using Open Vulnerability and Assessment Language for facilitating our assessment and analysis. Different from existing solutions that manually assign probabilities to a Bayesian attack graph, we design a set of quantitative metrics to automatically analyze vulnerability and evaluate the proposed approach with real-world examples. Our results show the great potential of the proposed approach in further improving assessment quality.

Keywords-vulnerability; attack domain; bayesian attack graph; assessment; network security

I. INTRODUCTION

Measuring various aspects of network security plays an important role in protecting distributed systems against malicious intrusions. Despite that various network security techniques such as firewalls and intrusion detection systems have been developed, attackers are still able to exploit vulnerabilities and launch more and sophisticated attacks. Moreover, attackers often combine multiple vulnerabilities to achieve their goals. Thus, we cannot evaluate security only based on individual vulnerabilities as many existing solutions and have to integrate multiple measures to defeat sophisticated attacks. Recent works in the field focused on attack graphs (AGs), which examine all possible actions of an attacker for the entire network or hosts in the network. The AG generation is a process of correlating vulnerabilities in the network. Each vulnerability can be exploited to cause an unwanted effect on the network or a host. Hence, an AG represents the security status of the network, and it is a useful tool for network assessment and hardening.

Although several projects have been conducted and addressed different aspects of the problem, they have several limitations. (i) *Non-automation*. While many methods have been proposed to automatically generate AGs recently, network vulnerability assessment is still mainly done by time-consuming manual methods. As the AG generation is to correlate a number of vulnerabilities with their preconditions and post-conditions, it is still labor intensive and error-prone to determine these conditions for each vulnerability [1]. While several existing databases such as Bugtraq and NVD-CVE do have the descriptions of the preconditions and consequences of a vulnerability, the information is not stored as a separate attribute in these databases. As a result, it is not an integrated part of a vulnerability description, and has not been thoroughly examined. In addition, how to analyze the attack graph automatically is also a challenge. Therefore, an effective, automated, and extensible analysis method is still unavailable. (ii) *Lack of systematic quantitative metrics.* Although previous methods calculate security metrics based on several quantitative models [2, 3], these constructed models are often incomplete and are unable to generate systematic metrics. To address this issue, Liu et al. [4] proposed to use a Bayesian Network for quantitative assessment. However, they did not provide systematic assessment metrics to harden a network.

In this paper, we propose a new approach to address above issues by automating the collection of vulnerability information and designing an appropriate model and effective assessment metrics. Our main contributions are: (1) Our system integrates several existing resources and generates a database which contains a large amount of vulnerabilities and their exploit specifications with selected details. Since most vulnerability databases now available are not designed for generating AGs, most of the domain knowledge needs to be converted into standard exploit specifications for AG generation. Then the challenge is to select appropriate information from existing resources and integrate it into our database, preparing details for the proposed assessment model. Most existing assessment approaches concentrate on AG generation and ignore this issue. As the correctness of AGs depends on the quality of input datasets [5], our database facilitates us to design more quantitative solutions. (2) Different from common AGs, our assessment model is a Bayesian Attack Graph (BAG), which can be easily realized by automatic process as demonstrated with our toolkit. When a BAG is generated based on the nodes and edges derived from an AG, assessment metrics are derived using Bayesian inference. Such a separation makes us different all existing schemes: using BAGs to quantitatively evaluate comprehensive assessment metrics, instead of common AGs. (3) Different from previous methods that do not emphasize assessment metrics, we employ Fault Tree Analysis (FTA) to guide a systematic evaluation of a network. FTA is a mature method for dependency analysis which helps us find real causes of unreliability of a network and identify corresponding hardening solutions.

The remainder of this paper is organized as follows. We discuss related work in Section II, and present our system architecture in Section III. We present experimental evaluation in Section IV and conclude the paper in Section V.

II. RELATED WORK

Many approaches have been proposed to assess various aspects of network security based on AGs. Kotenko et al. [6] proposed a multi-level model of attack scenarios to evaluate network security, which is able to analyze various situations, such as diversity of attacker's positions, intentions and experience levels. They further developed a security analysis system based on an AG to consider different attack scenarios, and defined several security metrics to analyze attacker's actions from different network points. Ingols et al. [7]

Contact Author: Chunlu Wang, wangcl@bupt.edu.cn, Beijing University of Posts and Telecommunications (BUPT), China; Yancheng Wang and Tianle Zhang are with BUPT; Yingfei Dong is with University of Hawaii, USA. Supported by the National High-Tech Research and Development Plan of China under Grant No. 2009AA01Z438, 2009AA01Z431; the National Natural Science Foundation of China under Grant No.60703021; the PLA General Armament Department Pre-Research Foundation of China under Grant No.9140A150601090Z08.

978-1-61284-233-2/11/$26.00 ©2011 IEEE

improved common AGs into multiple-prerequisite (MP) graphs, and built the NetSPA system to conduct recommendations of preventative actions. Although the system was able to generate an "all sources, all targets" graph for a 50,000-host network in under twelve minutes, they need to rebuild the MP graph for each potential recommend-action in order to find identical recommendations. Thus it is not scalable. GARNET system [8] improved the attack graph visualization technique of NetSPA, but still conducted assessment with on similar search algorithms without quantitative assessment. Topological Vulnerability Analysis (TVA) [5] is able to analyze vulnerability dependencies and generate AGs automatically. It captures network connectivity and integrates with host-based asset inventory technology. It simulates multi-step attacks on network configurations and exploit models, and identifies key vulnerabilities and automatically generates recommendations using AGs.

The closest work to ours is [4], which exploits BNs to model network vulnerabilities and determine a value representing the overall security of a network. BNs incorporate causality in the graph structure and provide a formal inference method to formulate conclusions. However, the method only solves some common problems for probability inference (e.g., the MPE problem), and they did not develop metrics from the perspective of vulnerability analysis. Meanwhile, their probability information of a BN is obtained manually. Despite these limitations, the results clearly indicate that a BN is an appropriate model for quantitative assessment.

BNs have also been applied for reliability modeling. Bobbio, et al. [9] discussed how the Fault Tree (FT) can be built with BNs. They provided a mapping between a FT and a BN, and showed how inference in the latter can be used to obtain classical FT parameters. Nicol et al. [10] argued that the evaluation of system dependability and system security have a lot in common. They suggested that many concepts used in FTA can be transferred to security analysis. We are inspired by these ideas and apply BNs for quantitative security assessment, different from the existing methods that mostly solve dependency analysis problems.

III. System Architecture

A. System Overview

Fig.1 is the overview of the assessment process. The *Data Repository* contains network configuration (e.g., topology) and security policy from an administrator, in addition to our vulnerability database introduced in the following subsection. This information is used for initializing Atomic Domains (ADs) and is converted into a format for attack graph generation. The *Vulnerability Scanner* consists of server-side agents and client-side agents. A client agent is installed at each host and controlled by a server agent. It summarizes current vulnerabilities on hosts to perform the *Network Analysis*. An *Attack Graph* is generated based on the Network Analysis and then annotated with attack probabilities (provided by administrators) to build a *Bayesian Attack Graph*. Via Bayesian inferences, we obtain our assessment results.

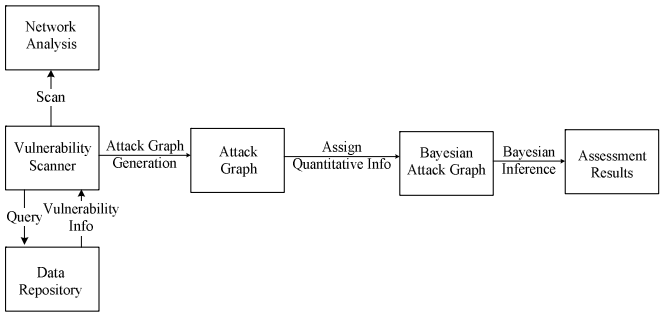


Figure 1. System overview.

B. Our Vulnerability Database and Vulnerability Scanning

Constructing a Vulnerability Database for Quantitative Assessment. In order to build an AG, we need a database for the scanner to identify vulnerabilities. A few well-maintained vulnerability databases are National Vulnerability Database (NVD) [11], Bugtraq security database [12], and Open Source Vulnerability Database (OSVDB) [13]. We choose NVD as the main source and also collect information from other sources to enhance our knowledge for each vulnerability. NVD is considered as the most complete and up-to-date source. It is also synchronized with the Common Vulnerabilities and Exposure (CVE) referencing standard [14]. We use a separate agent to probe the NVD database at short intervals in order to obtain newly published vulnerabilities. In addition, because most databases such as Bugtraq and OSVDB support the CVE standard, useful information such as the solution to fix a vulnerability is also added into our database.

Besides the basic attributes of a vulnerability such as publishing date and description, our database also derive the exploit conditions from NVD. In exploit conditions, the preconditions represent how each vulnerability may be exploited, and the post-conditions encode the result of its exploitation [5]. For the sake of simplicity, we consider the condition as the access level obtained by an attacker on a host.

Vulnerability Summary for CVE-1999-0009
Original release date: 04/08/1998
Source: US-CERT/NIST
Access Vector: Network exploitable
Access Complexity: Low
Authentication: Not required to exploit
Impact Type: Provides administrator access

Figure 2. A vulnerability example.

A Vulnerability Example. Fig.2 is a vulnerability's detailed information obtained from NVD. From the cvss-guide [15] and the above "Assess Vector" description, we know that if the vulnerability exists on a host, an attacker can exploit the vulnerability by using any machine that connects to it. Thus the precondition of exploiting the vulnerability is "none" because it is network exploitable. Next, the "Impact Type" shows that an attacker can obtain the privilege of administrator via this vulnerability. Hence the post-condition of exploiting the vulnerability is "obtaining root access". We have written a

Java procedure to obtain these parameters and save them as the pre- and post-condition attributes in our database.

From the "Access Complexity", we can further know that the attack complexity of the vulnerability is "Low". In Common Vulnerability Scoring System (CVSS) [15], AC is a variable that has three values: 0.31(H), 0.61(M) and 0.71(L). Therefore, the attack complexity of the vulnerability is 0.71. Some previous works also use BNs to assess the network security [4]. They set the probability value experientially. However, the database contains more than 30000 vulnerabilities and new ones are being added each day. So it is time-consuming to assign the probability value for each vulnerability manually. Although the probability value can only be 0.31, 0.61 or 0.71 (as illustrated in Sec.III.D), our method can automatically generate a BAG, while previous methods can not.

Vulnerability Scanning. In order to track network vulnerabilities, we developed a customized vulnerability scanner based on Open Vulnerability and Assessment Language (OVAL) [19], an open language to determine whether vulnerabilities exist on a system. Our scanner is different from common scanners such as Nessus. First, OVAL is a standard language of NIST. Its vulnerability description is in consistent with CVE, so it naturally fits with our vulnerability database. Second, as our scanner sits on each host, it will not lose any vulnerability by holding root privilege. Nessus local agents only provide very limited functions. As we mostly consider assessment for mission critical networks, the client installation is justifiable for security concerns.

C. Atomic-Domains-Based Attack Graph Generation

Attack graphs clearly depict how individual vulnerabilities can be combined by an intruder to force a network into an unsafe state. In order to generate AGs, we propose a novel atomic-domain-based approach which consists of two steps: atomic domain initialization and AG generation.

TABLE I. SUMMARY OF EXPLOITS

Vulnerability/Exploit		Victim host	Pre-condition	Post-condition
ap	Apa.Chunked Code buff.Ovf.	W	User	Root
tns	Oracle TNS listen buff.Ovf	D	User	Root
t_1	Trust Remote login(Any to W)	W	User	User
t_2	Trust Remote login(W to D)	D	User	User

Atomic domain initialization. Each *atomic domain* represents a host with a specific privilege. An attack can have privileges: *none, user*, and *root*. To generate the AG, we first find the available vulnerability set for each atomic domain. By analyzing vulnerabilities existing in the sub-network composed by hosts connected with the attacker, we initialize atomic domains of the attacker. We then initialize atomic domains concerning the next host, and so on. By decomposing the network into atomic domains, we simplify the AG generation. More importantly, when a change occurs, we only alter settings of corresponding ADs. Fig.3 shows an example network from [18]. Table I shows all vulnerabilities of the network, including their exploit conditions and hosts having these vulnerabilities. The result of AD initialization, i.e. vulnerabilities available (VA) for each AD, are shown in Table II, where ipDUser represents the user privilege of host D. Here we have a monotonicity assumption, i.e., an attacker's privilege always increases when he is launching attacks. Thus, vulnerabilities available of ipWRoot do not contain the vulnerability "ap".

Attack graph Generation. After analyzing the available vulnerability set for each atomic domain, the process for AG generation can be achieved by communications among all atomic domains. Starting from atomic domains of the attacker, we activate atomic domains concerning about the current attacks. Then other atomic domains can be activated in the same way. Finally, the AG for the network can be obtained till all atomic domains are been activated.

Fig.4 is the AG of the small network that contains five hosts. Each machine has full access to another's vulnerabilities. The whole AG generation (including scanning) takes 18 seconds with our method. The graph is not constrained by specific starting and ending points. Each host can be the attack start/goal. If the network owner designates a critical server as the attack goal, he will know how it can be compromised from all possible starting points.

D. Our Assessment Solution

BNs-Based Computation. To compute assessment metrics, we need a quantitative model to provide context for overall network security. We use a Bayesian network combined with BN inference to achieve this. We formally define a Bayesian Attack Graph (BAG) as follows.

Definition 1. Let a set $X=\{X_1,\ldots,X_n\}$ of discrete variables, for each X_i with predecessors $Pa_1, \ldots, Pa_n$, a *conditional probability table* (*CPT*) is a table which specifies the variable's *conditional probability distribution* (*CPD*) $P[Xi| Pa_1,\ldots, Pa_n]$.

When an AG integrates with the quantitative information represented by CPTs, it is called a *BAG*. In such a graph, each node is a Bernoulli random variable which only has a value of true or false. Each edge represents an exploit derived from the AG, where CPTs represent the probability dependency relationships among the nodes.

Definition 2. A BAG is a 4-tuple (S, τ, S_0, s_s), where S is a set of states, τ: $S\times S\rightarrow$ [0, 1] is the transition relation represented by CPT, $S_0 \subset S$ is a set of initial states, and $s_s \in S$ is the success state for the intruder.

After constructing the BAG based on an independence assumption as explained in the following, the *joint probability distribution* (*JPD*) $P(X_1, X_2, \ldots, X_n)$ for the given network can be decomposed into a set of CPDs by using the fundamental chain rule:

$$P(X_1, X_2, \ldots, X_n) = \prod_{i=1}^{n} P(X_i \mid Pa_i). \quad (1)$$

Constructing Bayesian Attack Graph. As described in Sec.III.B, we use the Attack Complexity in CVSS to assign the probability values of CPT. The limitation here is the value (e.g., $\pi(v_k)$ in the following) can only be 0.35, 0.61 and 0.71. Although it is not a continuous probability variable, it does not affect our assessment. Because we are more concerned about which vulnerability is more critical, not to what degree a single vulnerability's importance is.

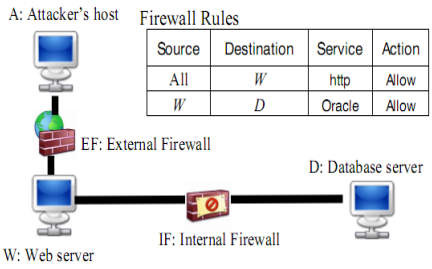


Figure 3. Example network.

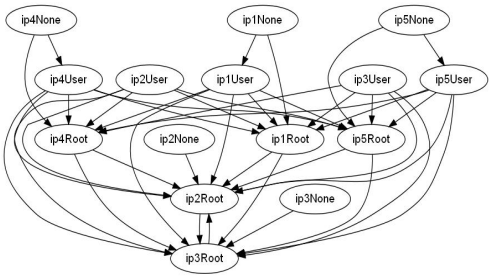


Figure 4. An example of attack graph.

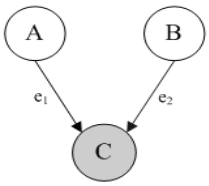
TABLE II. THE RESULT OF INITIALIZATION

ADs	Vulnerability Available
ipARoot	{ap,t_1}
ipWUser	{ap,tns,t_2}
ipWRoot	{tns,t_2}
ipDUser	{ap,tns,t_1}
ipDRoot	{ap,t_1}

Several existing methods also used the scores given by CVSS as the vulnerability metric [3]. Different from our approach, they scaled the scores of a given vulnerability to a value between 0 and 1 and interpreted it as the probability of exploiting the particular vulnerability. Although the Base Score (BS) in CVSS provides severity ratings for individual vulnerabilities, it includes the damage degree of exploiting the vulnerability to the system. For example, many vulnerabilities' BS is set to 10. However, it does not mean the probability of successfully exploiting the vulnerability is 1.

As mentioned before, one common assumption on the construction of BAG is: Given a node X, each parent node of X can independently influence the state of X [4], i.e., the vulnerability exploitation scores are independent. Existing methods use conditional probabilities to remove the above restrictive assumption. For instance, Bobbio et al. [9] argued that it can be relaxed by setting the entries of the CPT in a suitable way. For the sake of simplicity, we do not model the above case.

A Bayesian Attack Graph Example. Fig.5 shows an example of Bayesian attack graph which contains three atomic domains A, B and C. Edge e_1 and edge e_2 indicate the dependence relationship between the atomic domains. Each edge corresponding to a vulnerability exhibited on host C. We generate a CPT for each node, where C=1 indicates the atomic domain C is activated successfully. P(e_1) and P(e_2) are the probabilities of successfully exploiting the corresponding vulnerabilities, obtained from the vulnerability database. After specifying CPTs for all nodes, we obtain a JPD over all the variables by using equation (1). We then perform Bayesian inferences to compute the metrics.



A	B	P (C=0)	P(C=1)
0	0	1.0	0.0
1	0	1-P(e_1)	P(e_1)
0	1	1-P(e_2)	P(e_2)
1	1	[1-P(e_1)][1- P(e_2)]	1-[1-P(e_1)][1- P(e_2)]

Figure 5. A BAG example.

Fault-Tree-Based Analysis Metrics. Based on BAG, we present three evaluation metrics derived from FTA.

Unreliability of the Top Event (TE). In a BAG, a *Top Event (TE)* is a state with respect to the security property P. Assume that we are given a BAG (S, τ, S_0, s_s), where $S=\{s_1, s_2, \ldots, s_s\}$, $S_0=\{s_1, \ldots, s_k\}$ ($1 \le k < s$). The unreliability of the TE is given by the following equation:

$$P(TE=1) = p(s_s=1) = \sum_{s_1,\ldots s_k} P(s_1,\ldots s_k, s_s=1). \quad (2)$$

It is an overall metric for the whole network. When the administrator carries out a set of hardening measures and runs our system again, he will find the value has reduced and the network security has improved.

Criticality of Bottom Events (BEs). *Bottom Events (BEs)* are where the failure events enter the FT. In a BAG (S, τ, S_0, s_s), BE corresponds to S_0.Without loss of generality, we assume that there are multiple initial states s_0^1, s_0^2, ..., s_0^j. The criticality of bottom event s_0^k is a posterior distribution:

$$P(s_0^k \mid TE=1)\ (1 \le k \le j). \quad (3)$$

Our method is different from the measure in [16], which derived a logic proposition to represent the TE in terms of initial conditions (e.g., BEs). Although their approach helps compute the minimum-cost network hardening options, it still performs a qualitative analysis. In our model, however, we concentrate on computing the criticality of each BE. It is a quantitative analysis and can quantify how each BE affects the TE. In addition, as their approach is based on graph searching, when the AG changes (e.g., a vulnerability has been patched.), the proposition has to be built again. We only need modifying the CPT (without rebuilding the AG), and using the Bayesian inference to compute this metric. Hence, when the generation of AG takes a long time, our approach is more efficient.

The Most Critical System Component. Any vulnerability can be exploited to cause an unwanted effect on a system. In order to compute this effect, we measure the criticality of the component. Let $V=\{v_1, v_2, \ldots, v_n\}$ be the set of vulnerabilities of a BAG and v_k be a vulnerability in V. Let e_v be the exploit corresponding to v_k, whose transition probability is $\pi(v_k)$.The criticality of the component v_k is

$$C_k = P(TE=1) - P(TE=1 \mid \pi(v_k)=0.0)\ (1 \le k \le n). \quad (4)$$

We consider vulnerability v_k as the most critical component if and only if there is no vulnerability v_j ($1 \le j \le n$, $j \ne k$) in V such that $C_j > C_k$.

Input:
V – set of vulnerabilities
S – set of atomic domains (set of nodes of attack graph)
s_s – the success state for the intruder
U –the unreliability of TE
P – set of probabilities in CPD
Output:
Criticality of Each Component (Vulnerability)
Algorithm:

1. **Let** C be a set
2. **For** each $s_i \in S$, let $pa_1, pa_2, \ldots, pa_n$ be the parent nodes of s_i, let L be a list which contains 2^n arrays.
3. **For** each array $a_k \in L$ and $p_j \in P$, let $a_k[0]= pa_1, \ldots, a_k[n-1]= pa_n$, thus $a_k[j]=1$ or 0 ($0 \le j \le n-1$)

4. **Let** $a_k[n]=P(s_i=1)= 1-\prod_j (1-a_k[j]*p_j)$
5. **Let** $a_k[n+1]=P(s_i=0)=1-P(s_i=1)$
6. **End** For
7. **End** For
8. // **C**onstruct CPT for each node
9. **Let** m be the size of S and J be a list which contains 2^m arrays
10. **For** each array $a_j \in J$, let a_j [k] be the variable s_k in S ($0 \le k \le m-1$) and pa_k be the parent nodes of s_k
11. **Let** a_j [m]= $\prod_k p(s_k \mid pa_k)$
12. **End** For
13. // Construct JPD for the Bayesian attack graph
14. **For** each $v_k \in V$, let s_i be the node corresponding to the result of exploiting v_k
15. **Let** pa_i be the set of parent nodes of s_i
16. **For** $s_j \in pa_i$ let $x = P(s_j \rightarrow s_i)$
17. **Let** $P(s_j \rightarrow s_i) = 1-(1-x)/(1-\pi(v_k))$
18. **End** For
19. **Reconstruct** CPT of s_i
20. **Reconstruct** JPD for the Bayesian attack graphs
21. **Let** U' be the new unreliability of TE and C_k be the criticality of v_k
22. **For** each array $a_j \in J$
23. **If** s_s=1 { $U' = U' + a_j$ [m]}
24. **End** for
25. //Compute the unreliability of TE
26. $C \leftarrow C_k = U - U'$
27. **End** For
28. **Return** C

Figure 6. Algorithm for computing criticality of each component.

Our method is different from that of Sheyner et al. [17]. They remove one vulnerability at a time from the intruder's arsenal, and then they run the model checker. As the produced AG's size has reduced, a user is able to judge the criticality of the removed vulnerability visually. However, it is usually infeasible and quite time-consuming to replicate the generation of AG for each vulnerability, especially when there are a large amount of vulnerabilities. Our method, without rebuilding an AG, only needs changing some probability values of the vulnerability's CPT in the BAG (e.g., p (e_1) in Fig.5). Fig.6 shows a procedure that more precisely describes this process. Based on the generated BAG, we make a small change to the quantitative info (line 16-17), and then we do Bayesian inference again. Compared to [16] [17], this is our improvement as the time required to do Bayesian inference is much less than that of AG generation.

Complexity Analysis. In the stage of attack graph generation, the worst-case complexity for a network including m hosts is $O(m^2)$. In the stage of network assessment, the complexity of computing the unreliability of the TE and criticality of BE are both $O(n2^n)$, where n is the number of nodes in a BAG. The computing process is shown in Fig.6, whose time is mainly spent on constructing JPD. The complexity of finding the most critical vulnerability is $O(vn2^n)$, where v is the total number of vulnerabilities.

IV. EXPERIMENTAL EVALUATPON

We test our tool on a practical network of 15 hosts as shown in Fig.7. In this network, each host has full access to one another. There is an attacker outside the firewall; he only has four directly-accessible hosts: Ip1, Ip2, Ip3 and Ip4. His eventual goal is to disrupt the functioning of the database. For which, the intruder needs the root privilege on the database host Ip15. We first scanned the entire network, which took 149 seconds, and the scanner found 51 vulnerabilities. Host 15 was designated to be the attack goal and a AG was generated based on atomic domains. Fig.8 shows the resulting graph. It contains four BEs (marked by the shaded oval) and one TE (marked by the shaded octagon). By using the approach mentioned in Sec.III, we constructed the BAG. Then, Bayesian inference was used to compute the proposed assessment metrics. The whole process took 658 seconds in total.

We assign the prior probabilities to BEs as follows: P(ip1None=1) = P(ip2None=1) =P(ip3None) =P(ip4None) =0.5. The experiment results are as follows: (i) The unreliability of the TE is 0.6. In other words, the probability that the attacker can achieve his goal is 0.6. (ii) As shown in Fig.9, the most critical bottom event is IP3None. Thus it is better for the network owner to monitor host 3 by a single network-based IDS or disconnect the relationship between host 3 and the attacker host. (iii) The most critical component of the system is the vulnerability CVE-2009-0100 on Ip3 (as shown in Table III). Patching it will improve the network security effectively.

We add five hosts to the network and run the tool again. Here are the results: (i) The unreliability of the TE is 0.69. As the number of hosts has increased, the intruder has more attack opportunities. So he can achieve his goal more easily. (ii) The most critical bottom event is ip3None in Fig.9; while compared with the above experiment, the gaps of criticality between four BEs has decreased. Thus the impact of each BE to TE has weakened.

We compare our approach with that of [4]. The topology and vulnerabilities of the network are the same. As we have added one more attacker, thus the AG generated (as shown in Fig.10) is a little bit different.

As mentioned in Sec.III, our approach is different from [4]. By exacting the probability information from the vulnerability database, our BAG can be constructed automatically. As a result, our tool requires no manual labor in the overall process from network scanning to analysis. Furthermore, Table IV shows our exploit probability is similar to theirs which was set experientially. So, when it is difficult to set the exploit probability manually, (e.g., there are many new vulnerabilities in the network), our method can solve the same problem but faster. Here are the experiment results: The unreliability of the TE is 0.44. The most critical basic event is ip0Root and v_{23} of host 2 is the most critical vulnerability. These metrics show us how to harden a network. Liu et. al [4] did not discussed such security metrics for network hardening as we.

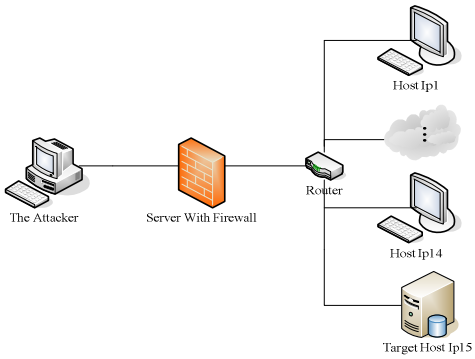


Figure 7. Experimental network setting.

TABLE IV. QUANTITATIVE INFO OF BAG

Vulnerability	Exploit Probability[4]	Exploit Probability in Database
v_{21}	0.6	0.61
v_{22}	0.3	0.31
v_{23}	0.8	0.71
v_{31}	0.3	0.31
v_{32}	0.8	0.71
v_{41}	0.3	0.31
v_{42}	0.8	0.71
v_{43}	0.5	0.61

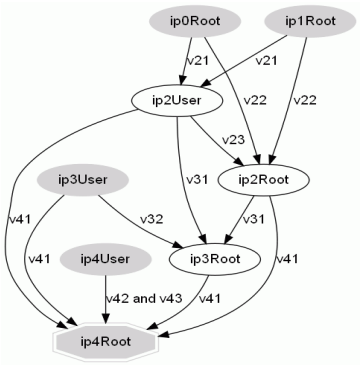


Figure 10. The attack graph.

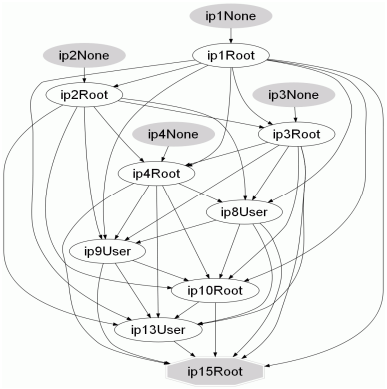


Figure 8. BAG of the example.

TABLE III. CRITICALITY OF EACH COMPONENT

CVE-ID	Host	Criticality
2009-0100	Ip3	0.43
2009-0087	Ip1	0.29
2009-0087	Ip2	0.29
2009-0238	Ip3	0.23
2009-0087	Ip3	0.22

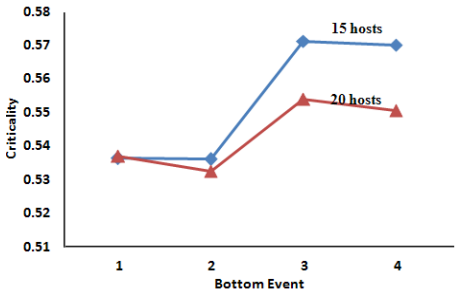


Figure 9. Criticality of bottom event.

V. CONCLUSIONS

We propose a new comprehensive network security assessment approach. We use atomic domains to simplify AG generation, and achieve efficient quantitative assessment by taking advantage of BNs and FTAs. We will further evaluate our approach on large networks and optimize our method tool.

REFERENCES

[1] R. P. Lippmann, and K. W. Ingols, "An Annotated Review of Past Papers on Attack Graphs," Tech. Report, Linoln Lab, MIT, 2005.

[2] O. Sheyner, J. Haines, S. Jha, R. Lippmann, and J. M. Wing, "Automated generation and analysis of attack graphs," The Proc. of IEEE SP, Oakland, CA, May 2002.

[3] H. L. Vu, K. K. Khaw, T. Chen, and F. Kuo, "A new approach for network vulnerability analysis," In Proc. of LCN 2008, 2008.

[4] Y. Liu, and H. Man, "Network vulnerability assessment using bayesian networks," In Proc. of SPIE, 2005.

[5] S. Jajodia, and S. Noel, "Topological Vulnerability Analysis: A Powerful New Approach for Network Attack Prevention, Detection, and Response," Indian Stat. Institute Monograph Series, Singapore (2009).

[6] I. V. Kotenko, and M. Stepashkin, "Attack Graph Based Evaluation of Network Security," In Proc. of CMS, 2006.

[7] K. W. Ingols, R. Lippmann, and K. Piwowarski, "Practical attack graph generation for network defense," In Proc. of ACSAC, 2006.

[8] L.Williams, R. Lippmann, and K. W. Ingols, "Garnet: A graphical attack graph and reachability network evaluation tool," In Proc. of VizSEC, 2008.

[9] A. Bobbio, L. Portinale, M. Minichino, and E. Ciancamerla, "Improving the analysis of dependable systems by mapping fault trees into Bayesian networks," Reliability Engineering and System Safety, vol. 71, no. 3, pp.249–260, March 2001.

[10] D.M. Nicol, W.H. Sanders, and K.S. Trivedi, "Model-Based Evaluation: From Dependability to Security," IEEE Transactions on Dependability and Secure Computing, Vol. 1, No.1, January-March 2004.

[11] NVD: National Vulnerability Database. Accessed from http://nvd.nist.gov/ on Mar 2010.

[12] SecurityFocus. Accessed from http://www.securityfocus.com/ on Mar 2010.

[13] OSVDB: Open Source Vulnerability Database. Accessed from http://www.osvdb.org/ on Mar 2010.

[14] CVE: Common Vulnerabilities and Exposures Database. Accessed from http://www.cve.mitre.org on Mar 2010.

[15] CVSS: Common Vulnerability Scoring System. Accessed from http://www.first.org/cvss/cvss-guide.html on Mar 2010.

[16] L. Wang, S. Noel, and S. Jajodia, "Minimum-cost network hardening using attack graphs," Computer Communications 29(18), 11 (2006).

[17] O. Sheyner, and J. Wing, "Tools for Generating and Analyzing attack Graphs," In Proc. of FMCO, pages 344–371, 2004.

[18] R. Hewett and P. Kijsanayothin, "Host-Centric Model Checking for Network Vulnerability Analysis," In Proc. of ACSAC, Washington, DC, USA, IEEE Computer Society, pp. 225-234, 2008.

[19] Open Vulnerability and Assessment Language. Accessed from http://oval.mitre.org/ on Mar 2010.