An empirical study of a vulnerability metric aggregation method

Su Zhang, Xinming Ou
Kansas State University
Manhattan, KS, USA
{*zhangs84,xou*}*@ksu.edu*

Anoop Singhal
National Institute of Standards and Technology
Gaithersburg, Maryland, USA
psinghal@nist.gov

John Homer
Abilene Christian University
Abilene, Texas, USA
jdh08a@acu.edu

***Abstract*—Quantifying security risk is an important and yet difficult task in enterprise network risk management, critical for proactive mission assurance. Even though metrics exist for individual vulnerabilities, there is currently no standard way of aggregating such metrics. We developed a quantitative model that can be used to aggregate vulnerability metrics in an enterprise network, with a sound computation model. Our model produces quantitative metrics that measure the likelihood that breaches can occur within a given network configuration, taking into consideration the effects of all possible interplays between vulnerabilities. In order to validate the effectiveness (scalability and accuracy) of this approach to realistic networks, we present the empirical study results of the approach on a number of system configurations. We use a real network as the test bed to demonstrate the utility of the approach, show that the sound computation model is crucial for interpreting the metric result.**

***Keywords*-enterprise network security; attack graph; vulnerability metrics, quantitative risk assessment**

I. Introduction

Proactive security is an important part of mission assurance and critical-infrastructure protection. Metrics indicating inherent risk in a network system can help in prioritizing precious resources to improve security and reduce the possibility of mission interruption from successful cyber attacks. Quantifying a security level for large-scale networks has been a challenging work for a long time. System administrators currently have to act based on their experience rather than objective metrics and models. Much work has been done along the line of attack graph construction from network configuration in order to analyze network security [1], [2], [4], [5], [6], [7], [10], [11], [12], [13], [14], [15]. Attack graphs can show the cumulative effect of vulnerabilities throughout the network by visualizing the logical dependency between the attacker's initial position and the attacker's goal. However, attack graphs alone cannot tell how severe or dangerous these attack paths may be. CVSS metrics [8] have been developed to indicate certain security levels for each single vulnerability. Moreover, a number of works [2], [10], [16], [17], [18], [19] have attempted to compute the cumulative effect of vulnerabilities in a network quantitatively. In our prior work we proposed a sound approach [3] to aggregating vulnerability metrics in an enterprise network with a clear semantics on what the calculated metrics mean: the likelihood a "dedicated attacker" can successfully penetrate the system and achieve certain privileges. Compared to previous works, this approach is based on a dependency attack graph which can be computed more efficiently [9], [11], and it correctly handles a number of important graphical structures such as shared dependencies and cycles to ensure the correctness of the result. In order to rigorously evaluate the effectiveness of this algorithm, we performed a series of empirical studies of this vulnerability metric aggregation method.

This empirical study is important because it can reveal both the effectiveness and limitations of a risk assessment method. Earlier work on security metrics has also performed substantial empirical evaluation [10] on production systems. Our method is based on a modern attack graph that has efficient computation, and it calculates the metric directly on a dependency attack graph without expanding it to a state attack graph which may incur an exponential blow up [6], [10]. To evaluate the benefit of such a method requires empirical evaluation on both its metric results and its running time on realistic systems. The empirical evaluation can also identify incorrect model assumptions or input parameters that make the result unrealistic and unusable, providing a feedback loop to calibrate the metric model.

There are a number of challenges of the empirical study. The configuration information of network is hard to obtain due to privacy or commercial reasons. It is also difficult to determine a set of proper parameters for the risk assessment approach. Implementation requires much work as well.

In order to address the aforementioned difficulties, we firstly talked with the system administrator and requested daily scanning results of certain machines in the Computing and Information Sciences Department at Kansas State University, We built a database to store related vulnerability information in NVD, where we retrieve the relevant information for each host's vulnerabilities. A parser is developed to construct input for MulVAL [11], [12], the attack graph tool used in our research, to generate attack graphs which were then used to calculate the security metrics based on the algorithms from our prior work [3].

II. VULNERABILITY METRIC AGGREGATION METHOD

The MulVAL attack graph [11] is used as a structural basis for security metrics calculation, although our approach should be easily adapted to other attack graphs generators with similar semantics [4], [5].

A. An example scenario

Figure 1 shows an example enterprise network, which will be used to illustrate a number of our security metrics algorithms.

Reachability and Host: There are three subnets and two firewalls (one internal and one external). The web server resides in DMZ which could be reached from Internet through the external firewall. The database server is in the internal subnet which contains sensitive information. It can only be reached through web server and the User subnet. The user workstations (used by normal employees) are all in the User subnet. All outbound connections from the User subnet are allowed by the external firewall.

Vulnerabilities: The web server has the vulnerability CVE-2006-3747 in the Apache HTTP service, which could be utilized by remote attackers to gain certain level of privilege to execute arbitrary code on the server. The database server has the vulnerability CVE-2009-2446 in the MySQL database service, by which attacker could gain administrator's privilege. The workstations contain the vulnerability CVE-2009-1918 in Internet Explorer; if an innocent user accessed malicious data through IE, the machine he is using will possibly be hacked. Usually, the system administrator would have limited time or energy to address all security issues. He would like to know which vulnerability is more dangerous or more urgent than others and deal with that first. Our approach could assist system administrators in this prioritization by offering quantitative metrics.

Attack-graph semantics: The lower part of Figure 1 is the MulVAL attack graph of the aforementioned network. Information about each node of the attack graph is found at the right side of the figure. A MulVAL attack graph has three types of nodes: (1) attack-step nodes, represented within the graph as circular-shaped AND-nodes. Each AND-node indicates a step of attack which could happen when all preconditions (either configuration nodes or privilege nodes) are held; (2) privilege nodes, represented within the graph as diamond-shaped OR-nodes. Each privilege node stands for a certain level of privilege which could be derived from any one of its predecessors (AND-nodes); (3) configuration nodes, which are not shown in this graph. Each configuration node represents a configuration condition of the network. For example, the network connections or vulnerability properties are all included in configuration nodes. As one example attack path, the attack graph shows that an attacker could first compromise the web server and then use it as an intermediate stop for his next step of attack on the database server (0-28-8-7-6-4-3-2-1).

Component metrics: The input of our metric model are component metrics, used as an indicator of each attack-step's likelihood of success. A number of these metrics are constructed based on CVSS metrics like Access Complexity (AC). The metrics can be regarded as conditional probabilities while all the preconditions for exploiting the vulnerability are satisfied. For example, if the vulnerability is hard to access (with a high value on AC), then even if all the preconditions are met, it will still have a small chance of being successfully utilized by attackers.

Assumptions in the metric model: We assume that an attacker knows the complete picture of the network, including network reachability, services running, and vulnerability information in applications. In other words, the adversary possesses the complete information in the attack graph. Further, we assume that the attacker will try all possible ways in the attack graph to compromise the system. In other words, the prior probability that an attack will be attempted is assumed to be one.

The output of our metric model is the likelihood of being hacked for individual machines. The major challenge in calculating the metrics with the above semantics is shared dependency and cycles in attack graphs. We developed techniques [3] to overcome these difficulties in our past work.

III. EXPERIMENT STRATEGY

To prepare the experiments, we performed a number of preliminary tasks.

- **Data Collection/Scanning**
  We scan seven windows servers running Windows Server 2003 at the CIS department of Kansas State University, by using the reference interpreter[1] of OVAL[2]. We have implemented a cron job to perform the scan daily and send the reports (XML files generated by OVAL) to a central repository.
- **Database Setup**
  In order to speed up the processing of data, we first extract all useful information from the NVD[3] data feeds (XML files including information of all vulnerabilities) into a separate mysql database, and build a tuple for each vulnerability in the database. The key of the tuple is the CVE ID, and other elements include CVSS metrics, attack range of the vulnerability (either remote service, local or remote client), consequences (compromising confidentiality, integrity or availability), and so on.

[1] http://oval.mitre.org/language/interpreter.html
[2] http://oval.mitre.org/
[3] National Vulnerability Database, http://nvd.nist.gov/

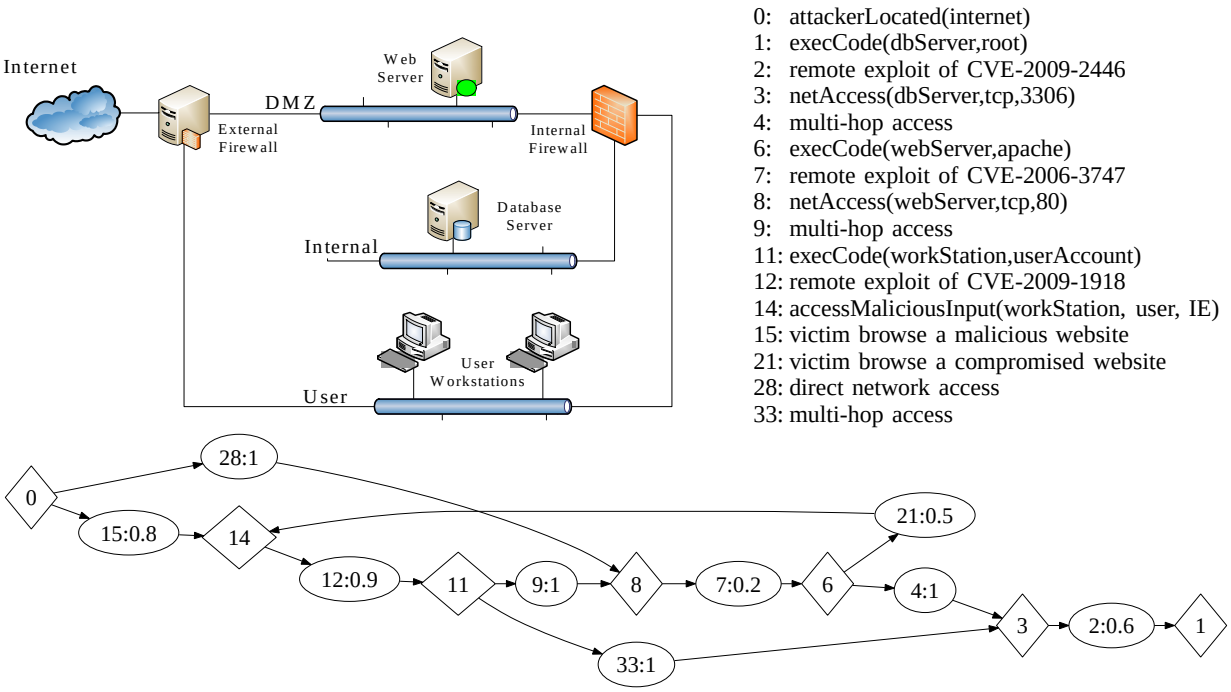


Figure 1. Example scenario and attack graph

- **Data Parsing/Input Construction**
  We then construct input data for the MulVAL attack-graph generator. For each machine, we parse its scanning report and obtain the CVE IDs of the vulnerabilities on this machine. By using these CVE IDs, we extract other information about the vulnerabilities through the database we built in the previous step. Meanwhile, we construct input for attack-graph generator based on the extracted information.

After we finish constructing the input data for risk assessment, we run our attack-graph generator and risk-assessment algorithms. Figure 2 illustrates the data flow for the empirical study.

We conducted two lines of experiments:

- **Empirical study on each single host.**
  We did experiments on each single host without considering the multi-host attack by assuming there is a direct connection between the attacker and each host. We then compare the security level of different hosts and present the result to the system administrator for verification.
- **The previous experiment repeated over time.**
  To observe the security metrics change trend over time, we did a number of experiments for each host at different points of time. We then analyze the detailed information returned from the vulnerability scan to confirm whether the risk trend indicated by metrics makes sense.

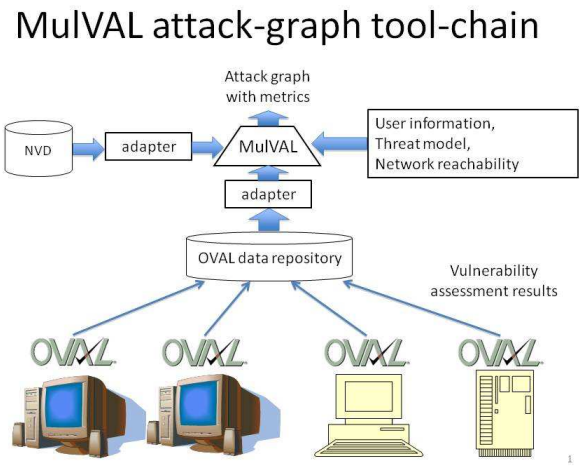


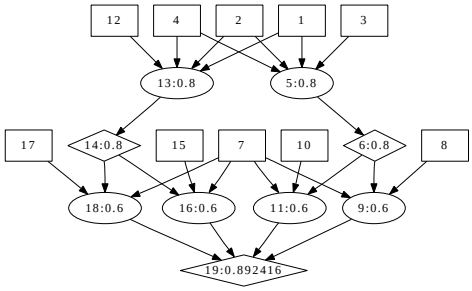
Figure 2. MulVAL attack-graph tool-chain

IV. Experimentation Result

When we saw the first batch of risk assessment results from the production systems, it quickly became clear that the original graphical model is insufficient to capture some hidden correlations important in gauging the risk levels. Thus we introduced additional modeling artifacts to capture them for the subsequent experiments.

Modeling artifacts for capturing hidden correlations: Figure 3 indicates two attack graphs with security metrics of two servers. From the two attack graphs we can tell that server (a) has many more vulnerabilities (on more applications) than server (b). This could be easily observed

(a)



(b)

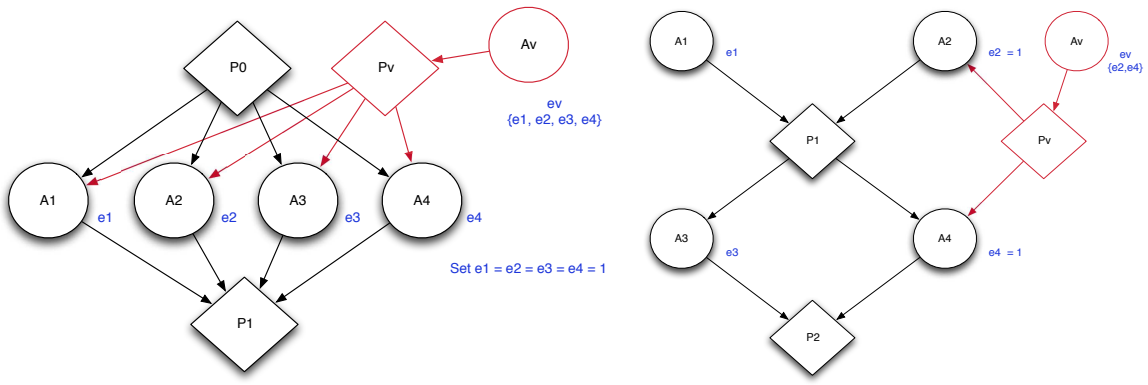
Figure 3. Attack graphs from two production servers

from the difference of the density of the two attack graphs (one of them is so dense that it appears almost like a line in the limited paper space). However, the difference of the cumulative security metrics of the two servers are not that obvious: server (a) is 0.99 while server (b) is 0.89. As we noticed, all the other servers in our department have more vulnerabilities than (b), meaning the security metrics are always close to 1. Intuitively, a question needs to be asked: is this the true reflection of these machines security situation? We consulted our system administrator and he thought that intrinsic differences among the vulnerabilities is an important factor while gauging the risk. If a host has one application with ten vulnerabilities, it will have lower security risk than one with ten different applications with one vulnerability in each. If an attacker is not familiar with a specific application, then even if this application has ten vulnerabilities, he still has a very low chance of utilizing these security holes. However, if there are ten applications each with one vulnerability, and if the attacker happens to know one of these applications well, he would have a much higher chance of compromising the machine successfully. The attack graph did not sufficiently capture the dependency between vulnerabilities and applications. Server (a) has 67 vulnerabilities in four applications while server (b) has only four vulnerabilities in two applications. Therefore, the metric calculated for server (a) should be much higher than server (b), which is not the case for the results from the original model.

Another hidden correlation arises in multi-stage attacks. Suppose an attacker just compromised one machine through a vulnerability from application A. If his subsequent targets have the same or similar vulnerability also from application A, he would have a very high chance of success since he should have known the underlying structure of application A very well. Suppose that exploiting a vulnerability has a 0.6 success likelihood. Based on our current attack-graph, a two-step attack utilizing the same vulnerability would give the attacker 0.36 (multiplied by two 0.6) chance of success. However, the experience of hacking the first target would lead to a much higher success possibility of the second step (almost 1). Therefore, the metrics would be close to 0.6 rather than 0.36, if we account for such hidden correlations.

We created additional modeling artifacts in our graphical model used in calculating the metrics, in order to capture these hidden correlations. The vulnerabilities belonging to the same application are grouped into one node representing a successfully exploitation on any of them. The access complexity metrics of the grouped vulnerability is equal to the lowest value from the vulnerabilities in the group. This schema not only applies to single machine (Figure 4.a) but also to multiple hosts (Figure 4.b) to capture the hidden correlations among multiple hosts. The likelihood of successfully exploiting at least one of the vulnerabilities within the same group is associated with the virtual node A_V, which is the parent of the original exploit nodes. This way the hidden correlation is correctly captured. In (a), an attacker success (failure) in exploiting A_V is equivalent to success (failure) in hacking $A_1, ...A_4$. The schema rectified the previous distorted metrics (which is extremely close to 1) by grouping similar vulnerabilities (*i.e.* $A_1, ...A_4$). In (b), if an attacker managed the expertise of hacking through A_2, it will succeed in A_4 as well since the two involve the same vulnerability.

Returning to the servers modeled in Figure 3, when running the algorithms on the extended graphical model, the difference in the calculated metrics for the two hosts widened: 0.98 vs 0.73. This difference conforms to our intuitive assessment based on system administrator's feedback. Also, the number of vulnerable applications (same as number of grouped vulnerabilities) becomes the number of exploits instead of number of vulnerabilities, reducing the attack-graph size as well (see Figure 5).



(a) (b)

Figure 4. Modeling artifacts for capturing hidden correlations

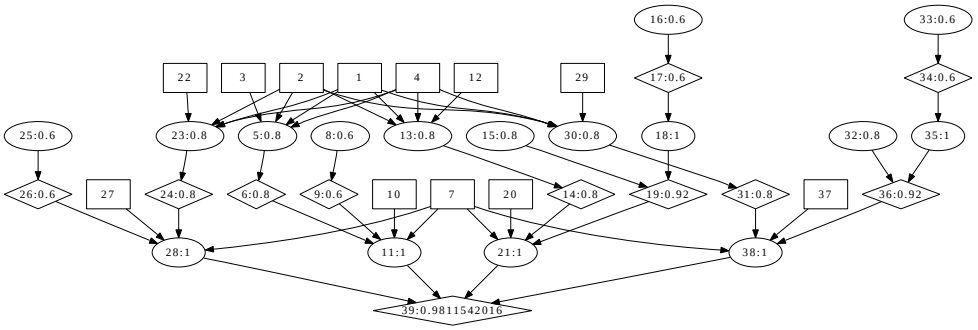
The calculated metrics of the two hosts are still very high which means there are vulnerabilities existing on the two hosts need to be patched. After we reviewed the attack graph, we realized that most of the vulnerabilities are client side and would need to be triggered through user actions. However, most of these applications have a low probability of being invoked by users, due to the functionalities of these servers, whereas we assigned a likelihood of 0.8 for all of them. This indicates that assigning such component metrics based on context is necessary in order to make the measured metrics reflect the true security situation of the network.

A. Experiment on individual machines

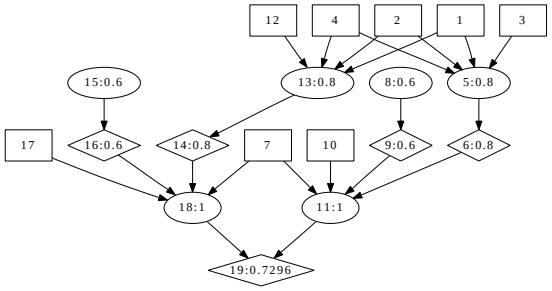
For this experiment, we run our risk-assessment algorithms against several different machines of CIS department at Kansas State University. The evaluating results indicate the security levels for these different machines. The departmental network has a fairly simple network topology. We assume all machines are directly connected to the Internet (where attacker located) and without considering multi-host attacks. For this configuration, the functions of servers and their cumulative metrics results are shown in Table I. In the table, the numbers indicate the likelihood various machines can be successfully compromised by an attacker. In order to justify the differences between the metrics, we reviewed their scanning reports. For example, on the report of November 4th 2010, machine1 is safer than machine4 in terms of the metrics (0.52 vs 0.816) with normal user privilege. After reviewing the scanning reports of the two machines, we are assured that our calculated metrics conform to the security level of the machines. For example, machine1 has two groups of service vulnerabilities (both under services with normal user privilege from the Windows system). Attackers could have two different major paths to exploit it. One representative is CVE-2010-3139 (exists in a number of Windows systems) and the other is CVE-2010-0820 (in Windows 7 only). Therefore the attacker could launch attacks either through certain libraries insecurely loaded by Windows Progman Group Converter (CVE-2010-3139) if he knew the user is using a generic Windows operating system. Or the attacker could utilize malformed LDAP messages (CVE-2010-0820) if he knew the victim machine is running Windows 7. Both vulnerabilities are fairly easy (with low Access Complexity metrics). Therefore, for machine1, the attacker has two easily accessible and independent paths to compromise it. As for machine4, not only it has the aforementioned two vulnerabilities in machine1, but also has two other user-privilege service security holes that could be utilized by attackers. One is CVE-2009-3959, Acrobat 9.x (before 9.3) allowing attackers remotely execute arbitrary code via a malformed PDF document easily (AC is low). The other is CVE-2009-4764, where an attacker could execute the EXE files embedded at the pdf files through Adobe reader remotely with reasonable amount of cost/effort (AC is medium). Therefore by having two additional independent attack paths, machine4 has a higher risk metric than machine1 with normal user privilege. Besides, machine4 has one local vulnerability CVE-2010-3959 (while machine1 does not) which could be used by attackers to further escalate their privilege from normal user to root through a crafted CMAP table in an OpenType font. Thus an attacker could not compromise machine1 with a root privilege but can gain the administrative privilege on machine4 with a likelihood of 0.539.

B. Experiments over time series

In order to observe the trend of security levels over time through our approach, we did the same experiment on the individual machines at different points of varying, created MulVAL input files representing network of each time spot, and evaluated them with the current implementation of our algorithm. We carefully reviewed vulnerability scanning reports for all the hosts we used in our experiments. The trend of the machines' security levels conform to our metrics. The change could be either an increase or decrease of vulnerability number or change of CVSS



(a)



(b)

Figure 5. Attack graphs with modeling artifacts for capturing hidden correlations

Host	Function	Individual metric over time					
		11/04/2010		12/19/2010		02/17/2011	
		user	root	user	root	user	root
machine1	Printing	0.52	0.0	0.52	0.0	0.52	0.27
machine3	Scanning	0.853	0.054	0.853	0.054	0.853	0.054
machine2	Camera video collection	0.988	0.028	0.988	0.028	0.988	0.028
machine4	DeepFreeze	0.816	0.539	0.816	0.539	0.816	0.280
machine5	Active Directory Mirror	0.958	0.141	0.958	0.141	0.958	0.141
machine6	Camtasia Relay	0.616	0	0.616	0	0.616	0.32
machine7	DNS/Active Directory	0.992	0.028	0.994	0.028	0.994	0.028

Table I
PROBABILITY OF COMPROMISE FOR INDIVIDUAL MACHINES OVER TIME

vectors. For example, machine6 has three grouped service vulnerabilities. One is on the general Windows framework. Among the vulnerabilities grouped into this one, the lowest Access Complexity (AC) level is medium. Another group of vulnerabilities fall into Windows 7, the lowest AC of which is low. The rest of the service vulnerabilities belongs to IIS, the easiest accessible level of these vulnerabilities is medium. While taking CVSS Access Complexity metrics into consideration, and based on our attack graph rules, we can derive the facts that attacker could execute arbitrary code as a normal user with probability 0.616. We did our experiments over three time spots: November 4th 2010, December 19th 2010 and February 17th 2011. We found that most of the metrics for attacker executing arbitrary code as a normal user did not change because the number and the Access Complexity of the vulnerabilities (service vulnerabilities running with user privilege) did not change. On the other hand, the metric for attacker running arbitrary code on machine6 as an administrator rose from 0 (in November and December) to 0.32 (in February). This change is attributable to a local exploitable vulnerability detected in February. Since the attacker has a certain chance (0.616) of executing arbitrary code as a normal user, along with the local exploitable vulnerability, he could escalate his privilege to root with probability 0.32. The Access Complexity of this group of vulnerabilities is low. Similarly, for machine1, there are two grouped vulnerabilities from November to February and all of the services are running under normal user privilege. Therefore the attacker has the same set of attack paths to compromise the host with normal user's privilege. There is only the one local exploitable vulnerability, first detected in February. The attacker could have one more

attack path to compromise the machine with root privilege; thus the risk metrics for machine1 root privilege is raised from 0 to 0.27. See table I for all the results.

V. Conclusion

We have presented an empirical study of a vulnerability metrics aggregation approach. The approach is sound in that, given component metrics which characterize the likelihood that individual vulnerabilities can be successfully exploited, the model computes a numeric value representing the cumulative likelihood for an attacker to succeed in gaining a specific privilege or carrying out an attack in the network. We confirmed the metric model's effectiveness by evaluating it on a number of servers in a departmental network. By analyzing the security level trend over time, we conclude that the metrics computed by our approach conformed to the real security situation change (*i.e.* increase or decrease of vulnerabilities or a change of a vulnerability's severity) of the scanned machines.

VI. Acknowledgment

This material is based upon work supported by U.S. National Science Foundation under grant no. 1038366 and 1018703, AFOSR under Award No. FA9550-09-1-0138, and HP Labs Innovation Research Program. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation or Hewlett-Packard Development Company, L.P.

References

[1] Paul Ammann, Duminda Wijesekera, and Saket Kaushik. Scalable, graph-based network vulnerability analysis. In *Proceedings of 9th ACM Conference on Computer and Communications Security*, Washington, DC, November 2002.

[2] Marc Dacier, Yves Deswarte, and Mohamed Kaâniche. Models and tools for quantitative assessment of operational security. In *IFIP SEC*, 1996.

[3] John Homer, Xinming Ou, and David Schmidt. A sound and practical approach to quantifying security risk in enterprise networks. Technical report, Kansas State University, 2009.

[4] Kyle Ingols, Richard Lippmann, and Keith Piwowarski. Practical attack graph generation for network defense. In *22nd Annual Computer Security Applications Conference (ACSAC)*, Miami Beach, Florida, December 2006.

[5] Sushil Jajodia and Steven Noel. Advanced cyber attack modeling analysis and visualization. Technical Report AFRL-RI-RS-TR-2010-078, Air Force Research Laboratory, March 2010.

[6] Richard Lippmann and Kyle W. Ingols. An annotated review of past papers on attack graphs. Technical report, MIT Lincoln Laboratory, March 2005.

[7] Richard P. Lippmann, Kyle W. Ingols, Chris Scott, Keith Piwowarski, Kendra Kratkiewicz, Michael Artz, and Robert Cunningham. Evaluating and strengthening enterprise network security using attack graphs. Technical Report ESC-TR-2005-064, MIT Lincoln Laboratory, October 2005.

[8] Peter Mell, Karen Scarfone, and Sasha Romanosky. *A Complete Guide to the Common Vulnerability Scoring System Version 2.0*. Forum of Incident Response and Security Teams (FIRST), June 2007.

[9] Steven Noel and Sushil Jajodia. Managing attack graph complexity through visual hierarchical aggregation. In *VizSEC/DMSEC '04: Proceedings of the 2004 ACM workshop on Visualization and data mining for computer security*, pages 109–118, New York, NY, USA, 2004. ACM Press.

[10] Rodolphe Ortalo, Yves Deswarte, and Mohamed Kaâniche. Experimenting with quantitative evaluation tools for monitoring operational security. *IEEE Transactions on Software Engineering*, 25(5), 1999.

[11] Xinming Ou, Wayne F. Boyer, and Miles A. McQueen. A scalable approach to attack graph generation. In *13th ACM Conference on Computer and Communications Security (CCS)*, pages 336–345, 2006.

[12] Xinming Ou, Sudhakar Govindavajhala, and Andrew W. Appel. MulVAL: A logic-based network security analyzer. In *14th USENIX Security Symposium*, 2005.

[13] Cynthia Phillips and Laura Painton Swiler. A graph-based system for network-vulnerability analysis. In *NSPW '98: Proceedings of the 1998 workshop on New security paradigms*, pages 71–79. ACM Press, 1998.

[14] Oleg Sheyner, Joshua Haines, Somesh Jha, Richard Lippmann, and Jeannette M. Wing. Automated generation and analysis of attack graphs. In *Proceedings of the 2002 IEEE Symposium on Security and Privacy*, pages 254–265, 2002.

[15] Laura P. Swiler, Cynthia Phillips, David Ellis, and Stefan Chakerian. Computer-attack graph generation tool. In *DARPA Information Survivability Conference and Exposition (DISCEX II'01)*, volume 2, June 2001.

[16] Lingyu Wang, Tania Islam, Tao Long, Anoop Singhal, and Sushil Jajodia. An attack graph-based probabilistic security metric. In *Proceedings of The 22nd Annual IFIP WG 11.3 Working Conference on Data and Applications Security (DBSEC'08)*, 2008.

[17] Lingyu Wang, Sushil Jajodia, Anoop Singhal, and Steven Noel. k-zero day safety: Measuring the security risk of networks against unknown attacks. In *Proceedings of the 15th European Symposium on Research in Computer Security (ESORICS 10)*, Athens, Greece, September 2010. Springer Verlag.

[18] Lingyu Wang, Anoop Singhal, and Sushil Jajodia. Measuring network security using attack graphs. In *Third Workshop on Quality of Protection (QoP)*, 2007.

[19] Su Zhang, Doina Caragea, and Xinming Ou. An empirical study of using the national vulnerability database to predict software vulnerabilities (submitted). 2011.