Automatic Generation of Host-based Network Attack Graph

Shangqin Zhong Danfeng Yan Chen Liu
sqzhong@telestar.bj.cn *yandf@bupt.edu.cn* *Lchen@bupt.edu.cn*
State Key Laboratory of Networking and Switching Technology, Beijing University of Post and Telecommunication, Beijing, 100876, China

Abstract

Attack graph plays an important role in network security, as it directly shows the existence of vulnerabilities in network and how attackers use these vulnerabilities to implement an effective attack, the analysis on the attack graph or the simulation of dynamic attacks through attack graph can help us easily find out the vulnerabilities in network, and take corresponding security measures, in order to strengthen network security. Previous attack graph generation methods are generally not suitable for large network, because of their high complexity of time, high consumption of space, and the large scale of attack graphs. Based on substantive analysis of the vulnerabilities in network, this paper describes a model for automatically generating and analyzing network attack graph. Besides, a prototype system bases on this model has been designed. At last, this prototype system was tested by a model network we built, and it was proved to be simple, flexible, and efficient.

1. Introduction

Up to now, more than 30,000 vulnerabilities have been found and published on the website of the American NIST [7]. These vulnerabilities have been stored in the National Vulnerability Database of NIST. Besides, statistics in NIST also shows that, about 13 CVE (Common Vulnerabilities and Exposures) are found per day! The truth is that, computers in the network are facing more and more threats, and being use of attack graph to analyze the existing threats have become increasingly important.

According to the attacks which have been detected by now, it can be seen that vulnerabilities of computers in the network are used to contribute effective attacks. Frequently, attackers start from a host and attempt to attack a certain host in the network. Followed with the success of this attack, attackers will attack another host by using resources of the victim. The process continues, until the attackers reach their goals. Network attack graph is a convenient tool for analysis of network security [14]. Based on the vulnerabilities detected in the network, it describes the exploits launched by attackers and the dependencies of these exploits. Automatic generation of the network attack graph, which is a hotspot of the current research, has been studied a lot. Some classic models and algorithms [1-6, 8-13, 15] have been put forward. But they have limitations more or less, such as the inefficiency of algorithms, the large scale of the attack graphs [11], and so on. It is, therefore, necessary to work out a better and more viable method to generate network attack graph.

According to the research we have done, this paper makes some improvement on the existing models and algorithms. We build a model for generation and analysis of network attack graph, and also, realize a prototype system for automatically generating network attack graph.

The remaining parts are organized as follows. Section 2 describes the model for generating network attack graph. Based on the model we have built in section 2, section 3 talks about the algorithm of generating attack graph. The model and algorithm are applied in a realistic network described in Section 4. Finally, we draw conclusions and lay out future works in section 5.

2. Model for generation and analysis

We build a model for generation and analysis of network attack graph on the reference of known models [5, 9, 12, 15]. The model is simple and flexible. Without structure redundancy that exists in other models. It is more convenient to generate network attack graph automatically, as well as to analyze it.

The model is composed of three parts: attack rule library, description of hosts in network, and attacker profiles.

978-0-7695-3507-4/08 $25.00 © 2008 IEEE
DOI 10.1109/CSIE.2009.102

2.1. Attack rule library

Attack rule library contains all the existing attack rules, which are used to generate attack graph. It bases on the distillation and abstraction of information provided by CAPEC (Common Attack Pattern Enumeration and Classification) website [6]. Each attack rule in the library is constructed by vulnerability, and it can be represented as a five-tuples: Attack_Rule = (Rule_ID, Rule_Info, Src_Conditions, Effects_on_Dest, Weight).

Rule_ID identifies an attack rule. Each Rule_ID correlates to an attack rule, as well as a vulnerability. Since each vulnerability has its unique vulnerable ID, when describing a host in the network, its vulnerable list is made up of a group of vulnerable ID (a further description is made in section 2.2). Now we can assume that the ID of attack rule is equal to the vulnerable ID. By this way, as long as we know the ID of vulnerability, we will know the ID of attack rule too (i.e. we establish a direct relationship mapping between vulnerable ID and attack rule).

Rule_Info is a brief description of attack rule. In order to describe how attacker use the vulnerabilities of a host to implement an attack, it's necessary to add some information about the attack beside the edge in attack graph, which is also benefit for the analysis of attach graph.

Src_Conditions represents the necessary conditions in the attacker's host that are needed to launch an effective attack. If the target host has the vulnerabilities related to the attack rule, it is assumed that it satisfies the conditions of an effective attack. So when build the attack rules, we only need to provide the essential conditions of attacker's host, conditions in target host is not necessarily needed. An effective attack will be launched as long as the attacker's host satisfies the conditions in Src_Conditions.

Effects_on_Dest represents the effects of an attack to the victim. If the vulnerabilities of victim are used, definitely, some effects will be caused in the victim. That is why attacker launches an attack.

Weight reflects the difficulty of the realization of an attack. The larger the weight is, the more difficulty to realize an attack, and the higher ability required by the attacker. We quantify it to a value by taking a comprehensive consideration of TLE (Typical Likelihood of Exploit) and ASKR (Attacker Skill or Knowledge Required), which are used in the website of CAPEC to describe some information of an attack. Both the TLE and ASKR are described in CAPEC as Low, Medium, and High. We let Low, Medium, High of TLE be the value of 3, 2, 1. Respectively, For ASKR, we make Low, Medium, and High correlated to the value of 1, 2, and 3. The value of the Weight is equal to the product of TLE and ASKR. In this way, a value of Weight arranging from 1 to 9 shows the difficulty of an attack.

2.2. Description of hosts in network

To make a complete description of network, we need the following information: the connecting relation among hosts in network, the vulnerabilities of each host, and some other information of a host. So each host should maintain a Link list, a Vulnerability list, and an Info list.

2.2.1. Link list. Each attack in the network can be resolved to a series of atomic exploits. Each atomic exploit can be deemed as an attack from a host to another direct-connected host. "Two direct-connected hosts" means the two hosts are connected through physical circuit, without a third host existing between them. A host's Link list records all its directly connected hosts.

2.2.2. Vulnerability list. Every host in network may have vulnerabilities. When these vulnerabilities are used, an effective attack may happen. Each vulnerability owns an ID marked by Vulnerability_ID, and each Vulnerability_ID has an unique correlated Rule_ID. Therefore, Vulnerability list is made up of a group of Vulnerability_ID.

2.2.3. Info list. It's not enough to completely describe the host and the status of network only through the information given in Link list and Vulnerability list. The Info list notes some other information about the host, such as the hardware type, operating system, available services and ports, and so on.

2.3. Attacker profiles

Attacker profiles contain some information about the attackers, and can be described by a two-tuples: (Host_IP,Goal_IP). Host_IP represents the host used by attackers to launch an attack, while Goal_IP represents the target host that the attackers want to attack. The two fields in the attack profiles have the following combinations: (Host_IP,null), (null,Goal_IP), and (Host_IP,Goal_IP). One specific combination should be given before generating the attack graph.

2.3.1. Without a certain goal IP (Host_IP,null). This combination only specifies the host of attacker, but does not specify the target host which attacker want to attack. Through this method, after attack graph is

generated, we can easily find out all the victims that can be attacked by an attacker from Host_IP.

2.3.2. Without attacker's IP (null,Goal_IP). This combination only specifies the target host, without a definition of the host of attacker. Through the attack graph generated in this method, we can find out all the hosts in network that can launch attack to the Goal_IP.

2.3.3. Attacker and goal are both given (Host_IP,Goal_IP). This combination specifies not only the target host, but also the host of attacker. In this situation, an attack graph shows all the attack routes from the attacker's host to the target hosts.

3. Generation of network attack graph

3.1. Architecture of the model

After gathering the information of network, we are able to generate a description of the hosts. Associated with the attack rule library and the attacker profiles given by network security analyst, the attacker-graph-generator is able to generate an attack graph of network through the algorithm describe in this paper. Figure 1 shows the architecture of this system.

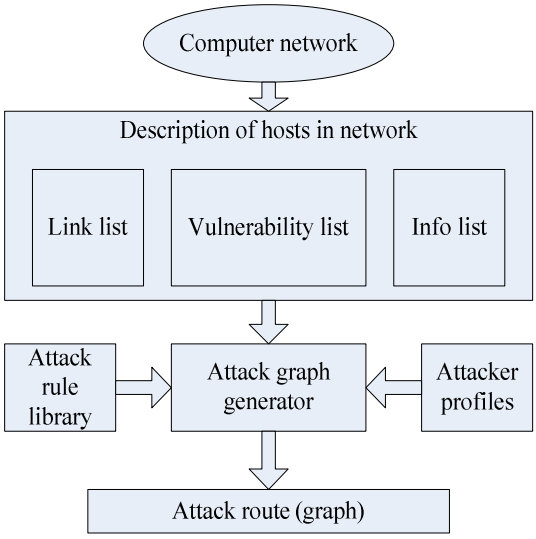

Figure 1. Architecture of network graph generation

3.2. Algorithm to generate network attack graph

In the process of launching an attack, attackers make use of vulnerabilities in victims. They begin with attacking a host that possesses vulnerabilities, and then they launch another attack by using the victim's resources, until they reach their goals. Attackers will not launch any meaningless attacks. Therefore, no loops exist in attack graph. Furthermore, through the discussion about attacker profiles in section **2.3**, we can conclude that different attack graphs will be generated with different attacker profiles. However, basic algorithmic principles for generating attack graphs are generally the same.

Based on the analysis above, this paper illustrates a way to find out all the victims, after the attacker's host is given. We adopt a positive, breadth-first algorithm to generate network attack graph. Its basic principle as follows:

(1) Put the node that stands for the attacker's host into an empty Node_queue.

(2) If the pointer of Node_queue is not null, the host that is pointed by the pointer will be regarded as the attacker's host in this exploit. From the hosts that directly connected to the attacker's host, we can find out which of them can be attacked by the attacker. If they are found and they haven't existed in the Node_queue, we put each of them in the Node_queue. That means an attack, from the attacker's host to the newly added host, has happened. At the same time, the pointer will point to another element in the Node_queue next to the attacker's host.

(3) Continue step (2), until the pointer of Node_queue is null, which means no more elements are in the Node_queue. At this time the algorithm comes to an end. The detailed description of the algorithm is shown as below:

```
Algorithm：Attack-graph_Generation (Input)
Input：Model for generation and analysis
Output：Attack route (graph)
(1) Host→Node_queue;
(2) int i=0;
(3) while(Node_queue[i]!=null)
(4) { int j=0;
(5)   Link_list=the link list of Node_queue[i];
(6)   while(Link_list[j]!=null)
(7)    { if(Link_list[j] is not int the Node_queue)
(8)      { int k=0;
(9)        Vul_list=the vulnerability list of Link_list[j];
(10)        while(Vul_list[k]!=null)
(11)        { Vul_ID=Vul_list[k];
(12)          Rule=the attack rule whose Rule_ID is equal to Vul_ID;
(13)          if(Src_Conditions in the Rule are included in the Info_list of Link_list[j])
(14)          { Link_list[j]→Node_queue;
(15)            Make an edge from Node_queue[i] to Link_list[j];
(16)            Put Rule_Info, and Weight beside the edge; }
(17)        k=k+1; } }
(18)     j=j+1; }
(19)  i=i+1; }
```

Figure 2. Algorithm to generate attack graph

Nodes in attack graph generated based on the above algorithm represented hosts in the network. The attack graph contains attack routes from the attacker's host to all its victims. The condition in the 7th row of the algorithm guarantee that no loop exists in the attack graph, and also, each attack route is the shortest one. Namely that, attack routes in the attack graph represent the shortest routes from attackers' host to its victims.

4. Experiment and analysis

4.1. Network environment

To test the model and algorithm, we build an experimental network. The topology of the experimental network shows as Figure 3. As can be seen, Host IP0 is the attacker's host, which uses Windows XP as its operating system. It can send HTTP request to host IP1, which is an IIS server and uses Windows 2000. Host IP2, whose OS is Windows XP, provides an application written using Java's AWT to host IP1. Host IP3 is an EFTP server, which can be accessed by hosts of IP1, IP2 and IP4. Host IP4 uses FreeBSD, and provides API to hosts of IP2 and IP3.

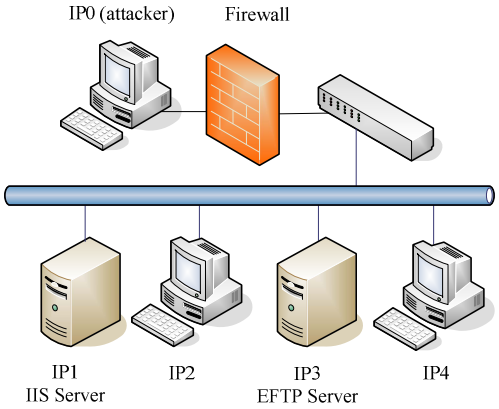

Figure 3. Topology of network

4.2. Network attack model

Based on the topology of the experimental network and the information provided by CAPEC website, we can see that host IP1 owns vulnerability numbered 11, and host IP2 owns vulnerability numbered 30, while host IP3 owns vulnerability numbered 45, and Host IP4 exists vulnerability numbered 8. Table 1 shows the information about the Link list, Vulnerability list, and Info list of hosts in the above network.

Table 1. Description of hosts in network

HostID	Link_list	Vul_list	Info_list
IP0	{ IP1,IP2, IP3,IP4 }	{…}	{… , Windows XP, HTTP request to IP1, …}
IP1	{ IP0,IP2, IP3,IP4 }	{…, 11, …}	{… Windows 2000, Symbolic link to IP3, Be able to latch onto privileged thread in IP2, …}
IP2	{ IP0,IP1, IP3,IP4 }	{…, 30, …}	{… Windows XP, Symbolic link to IP3, Can use API of IP4, …}
IP3	{ IP0,IP1, IP2,IP4 }	{…, 45, …}	{… Windows 2000, Can use API of IP4, …}
IP4	{ IP0,IP1, IP2,IP3 }	{…, 8, …}	{… FreeBSD, Symbolic link to IP3, …}

According to the existing vulnerabilities in the hosts, the attack rules used in this experiment are listed in table 2.

Table 2. Attack rules used in the experiment

Rule_ID	Rule_Info	Src_Conditions	Effects_on_Dest	Weight
8	Buffer overflow in an API call	The target host exposes an API to the user	Denial of service; Run arbitrary code; Information leakage; Data Modification.	2
11	Cause Web server Misclassification	Execute HTTP request to Web server	Information leakage; Privilege escalation.	4
30	Hijacking a privileged thread of execution	Be able to latch onto a privileged thread	Privilege escalation; Run arbitrary code.	9
45	Buffer overflow via symbolic links	Be able to create symbolic link on the target host	Denial of service; Run arbitrary code; Information leakage; Data Modification.	1

4.3. Result and analysis

By using the attack model build in section **4.2** as the input for the algorithm describes in section **3.2**, we generate an attack graph shown as Figure 4. As can be seen, nodes stand for hosts, and the edges stand for exploits. The weights of edges reflect the difficulty of successful attacks.

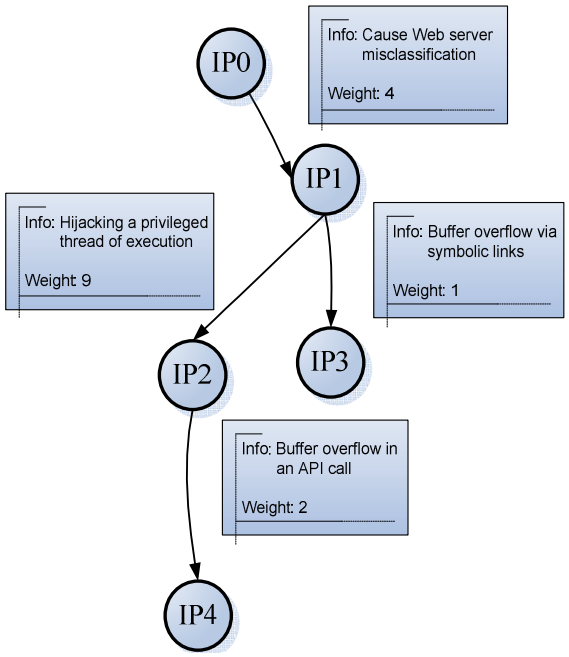

Figure 4. Attack graph of experimental network

From the attack graph generated in this experiment, we can see that there exist attack routes from host IP0 to other hosts. The route between IP0 and a host in attack graph represents the shortest attack route from attacker to this host. The sum of weights from host IP0 to host IP3 is 5, while it is 13 from host IP0 to host IP2. It means, from host IP0, it is easier to attack host IP3 than to attack host IP2. Obviously, network security manager should pay more attention to the safety of host IP3. Besides, we can also see that if the attacker wants to attack other hosts in the network, he must attack host IP1 first. So host IP1 is a critical node of the network, which should be pay special attention to.

Compared to the previous algorithms, the algorithm put forward by this paper has better time and space complexity. Suppose the number of the hosts in the network is n, the total number of the attack rules is K, and the maximum number of items in Info_list of hosts is I, the time complexity of this algorithm is $n^3 + K \times I \times n^2$. Besides, the attack graph generated in this method is compact. At most, n nodes are included in the attack graph, i.e. all hosts can be attacked by attacker. So it solves the problem of scale perfectly.

Table 3 shows the comparison between this method and some classic methods.

Table 3. Comparison to some classic methods

Paper	Time complexity	Scale
Scalable, Graph-Based Network Vulnerability Analysis [9]	$n^6 E$ n: the number of hosts E: the number of exploits	very large
Topological Analysis of Network Attack Vulnerability [10]	$n^6 E$ n: the number of hosts E: the number of exploits	very large
An Effective Method To Generate Attack Graph [12]	$M \times H \times P \times R$ M: the maximum number of exploits included in an attack path H: the number of hosts P: the number of protocols R: the maximum number of attack rules related to a certain protocol	large
Method described in this paper	$n^3 + K \times I \times n^2$	very small

As can be seen, previous methods generally have very high time complexities. And the scales of attack graphs generated by these methods are too large, which make the attack graphs hard to be analyzed. So it is hard for these methods to be put into use. However, the algorithm described in this paper not only has low time complexity, but also generates attack graphs of very small scale.

Furthermore, this algorithm only maintains a Node_queue to store the nodes generated in the attack graph. The Node_queue contains n nodes at most, thus, it has low consumption of space.

5. Conclusion

Previous attack graph generation methods are generally not suitable for large network, for their high complexity of time, high consumption of space, and the large scale of attack graphs. Since there isn't a perfect method to generate attack graph till now, building model more reasonably, this paper put forward a simple, flexible, and efficient method to automatically generate attack graph. Based on this method, we realized a prototype system to generate network attack graph, and it was proved to be feasible and effective by experiments. Of course, the prototype

system needs to be improved. Generating the Attack Rule Library and building model automatically are the directions of our further research.

6. Acknowledgements

This research was funded in part by the National Nature Science Foundation of China under contract 2006AAZ01Z448, and in part by the National 242 Information Security Program, under contract 2007A14. Besides, this paper is finished with the help offered by all members in our research centre, especially Qingmei Jia, Shuang Qian, Wenbin Wang, Yu Lu. Thank you for your cooperation.

7. References

[1] C.A. Phillips and L. P. Swiler. "A graph-based system for network vulnerability analysis". Proceedings of the DARPA Information Survivability Conference and Exposition, 2000, pages 71—79.

[2] Hamid Reza Shahriari and Rasool Jalili. "Vulnerability Take Grant (VTG): An efficient approach to analyze network vulnerabilities". Computers & Security, 2007, Volume 26, Issue 5, Page(s) 349-360.

[3] Jeannette M. Wing. Attack graph generation and analysis. "Conference on Computer and Communications Security", Proceedings of the 2006 ACM Symposium on Information, computer and communications security, Taipei, Taiwan.

[4] L. P. Swiler, C.A. Phillips, D.Ellis, and S.Chakerian. "Computer-Attack Graph Generation Tool". Proceedings of the Second DARPA Information Survivability Conference & Exposition (DISCEX II), Los Alamitos, California, 2007, vol.II, pp.307-321.

[5] Man Dapeng, Zhang Bing, Yang Wu, Jin Wenjin, Yang Yongtian. "A Method for Global Attack Graph Generation". Networking, Sensing and Control, 2008. ICNSC 2008. IEEE International Conference, 2008, Page(s):236 – 241.

[6] MITRE Website.
http://capec.mitre.org/data/dictionary.html, 2008

[7] NIST Website.http://nvd.nist.gov/, 2008.

[8] O.Sheyner, S.Jha, J.M.Wing, R.P.Lippmann, and J.Haines. "Automated Generation and Analysis of Attack Graphs". In 2002 IEEE Symposium on Security and Privacy. Oakland, California, 2002.

[9] P.Ammann, D.Wijesekera, and S.Kaushik. "Scalable, Graph-Based Network Vulnerability Analysis". Proceedings of the 9th ACM Conference on Computer and Communications Security, New York: ACM Press, 2002 pp.217-224.

[10] S.Jajodia, S.Noel, and B.O'Berry. "Topological Analysis of Network Attack Vulnerability". Managing Cyber Threats: Issues, Approaches and Challenges, V.Kumar, J.Srivastava, and A.Lazarevic, Eds. Dordrecht, Netherlands: Kluwer Academic Publisher, 2003.

[11] S. Noel and S. Jajodia. "Managing attack graph complexity through visual hierarchical aggregation". In Proceedings of the 2004 ACM workshop on Visualization and data mining for computer security, Washington DC, USA.

[12] Tao Zhang, Ming-Zeng Hu, Dong Li, Ling Sun. "An effective method to generate attack graph". Machine Learning and Cybernetics, 2005. Proceedings of 2005 International Conference on.IEEE.

[13] T.Tidwell, R.Larson, K.Fitch, and J.Hale. "Modeling Internet Attacks". Proceeding of the Second Annual IEEE SMC Information Assurance Workshop, United States Military Academy, West Point, New York, June 2001: IEEE Press, pp.54-59.

[14] Vaibhav Mehta, Constantinos Bartzis, Haifeng Zhu, Edmund Clarke, and Jeannette Wing. "Ranking Attack Graphs"[C].Carnegie Mellon University, 2006.

[15] Xinming Ou, Yayne F.Boyer and Miles A.McQueen. "A Scalable Approach To Attack Graph Generation". Conference on Computer and Communications Security, Proceedings of the 13th ACM conference on Computer and communications security, Alexandria, Virginia, USA. 2006.