Automatic generation algorithm of penetration graph in penetration testing

Xue Qiu[1]
Beijing Key Laboratory of Network Technology
Beihang University
Beijing, China
qagreessive@sina.com

Shuguang Wang[3]
Beijing Key Laboratory of Network Technology
Beihang University
Beijing, China
Shuguang.wang1990@gmail.com

Qiong Jia[2]
Beijing Key Laboratory of Network Technology
Beihang University
Beijing, China
jiaqiong1219@126.com

Chunhe Xia[4]
Beijing Key Laboratory of Network Technology
Beihang University
Beijing, China
xch@buaa.edu.cn

LiangShuang Lv[5]
Beijing Key Laboratory of Network Technology
Beihang University
Beijing, China
lls@buaa.edu.cn

Abstract—**Penetration graph is a kind of attack graph which is widely used in penetration testing. It is an import tool to analyze security vulnerabilities in the network. However, the previous research on the generation methods of penetration graph have met a lot of challenges. Some methods are out of date and not applicable for practical scenarios; some may possibly leave out the import attack paths; some do not consider the probability of exploitation of each attack path and some failed to solve the problem of circle path and combination exploitation. We propose an automatic generation algorithm of penetration graph that optimizes the network topology before generating the penetration graph, which can reduce the redundant information effectively. We combine the penetration graph generation method with the CVSS (Common Vulnerability Scoring System) information together, increase the reliability of each attack path. Experiment result shows that the method can generates multi-path correctly and effectively, which can clearly show the structure of network, facilitates the testers' analysis of the target network, and provides reference for executing penetration testing.**

Keywords- Penetration testing; Penetration graph; Automatic generation; matrix; vulnerability

I. INTRODUCTION

Penetration testing is a new network security audit method developed in recent years, which is used by testers to imitate the real attack and gain unauthorized access to the target gradually on the premise of guaranteeing the security of the target[1-3]. However, the variety of current network vulnerabilities makes the process of penetration testing quite complex. Most of the previous testing methods heavily rely on the professional experience of testers to exploit the vulnerabilities.

In order to increase the probability of successful penetration testing, testers can derive penetration graph from network topology and vulnerabilities, and simulate the execution of penetration testing to provide reference for the real penetration testing. As a new tool, penetration graph can show the network structure clearly and facilitate the analysis of the relationship among vulnerabilities of the target network for testers.

However, because of redundancy and complexity, traditional penetration graph expands rapidly with the extension of the scale of network. In order to solve this problem, this paper proposes an automatic penetration graph generation method. Experiments show that this method is clearer and more effective than traditional method.

The remainder of this paper is organized as follows: Section Ⅱ introduces the related work of attack graph research. In section Ⅲ, we present a detailed explanation of the proposed penetration graph generation algorithm. Then in section Ⅳ, we implement a prototype system based on our proposed automatic generating algorithm. In section Ⅴ, we will show soundness of this algorithm by conducting some relevant experiments. Finally, we give the conclusions.

II. RELATE WORK

Attack graph was first proposed by Swiler in 1998[4], in which the nodes can identify attack stag, for example, the class of machines the attacker has accessed and the user privilege level he or she has compromised. The arcs represent attacks or stages of attacks. There are two types of attack graphs: the state-based attack graphs and the attribute-based attack graphs. TBALE I shows the differences between them.

TABLE I. Comparison Between The Two Attack Graphs

Graph type	State-based attack graphs	Attribute-based attack graphs
Nodes	State of target network and attacker	System conditions and atomic attack
Arcs	Attacks	The relationship between nodes
Applicable scene	Small-scale networks	Large-scale networks
Attack path	The state transition between nodes	The attack behaviors between nodes

In state-based attack graphs, the number of attack paths will increase exponentially along with the scale and the number of target vulnerability, so it cannot be applied to large-scale networks. Attribute-based attack graphs has better scalability, it can be applied to large-scale network[5].Therefore, we concentrate on the attribute-based attack graphs here.

Ingols et al.[6] created a network model using firewall rules and network vulnerability scans, then they used the model to compute network reachability and attack graphs which can represent potential attack paths by adversaries to exploite known vulnerabilities in server software. But the threat and countermeasure models they used were out of date and not applicable for practical scenarios.

Alhomidi et al.[7] proposed a method to explore the graph. Each attack path is considered as an independent attack scenario from the source of attack to the target. They developed a genetic algorithm (GA) to determine the risks of attack paths and produce useful numeric values for the overall risk of a given network. This method provides a way of exploring a large number of possible attack paths. However, it may leave out the high risk paths.

Security metric model is widely used in the attack graph generation methods. Wang et al. [8, 9] proposed an attack graph-based probabilistic metric model to quantify the overall security of network system. Hunag et al.[10] provided an approach to attack graph distillation. They transformed an dependency attack graph into Boolean formula and assigned cost metrics to attack variables in the formula based on the severity metrics. Then, they applied Minimum-Cost SAT Solving(MCSS) to find the most critical path in terms of the least cost incurred by the attacker to deploy multi-step attacks, leading to certain crucial assets in the network. The method can distill critical attack graph surface from the full attack graphs generated for moderate-sized enterprise network in only several minutes. However, this method has a possibility of leaving out the import attack paths.

Long et al. [11] proposed an approach that combines the advantages of exploit-dependency attack graphs and adjacency matrices, which results in quadratic complexity. But they do not take the probability of exploitation of each attack path into consideration. In order to solve this problem, some researchers tried to use CVSS information to help generate attack graphs. Gallon et al. [12] proposed to combine attack graphs and CVSS framework, in order to add damage and exploitability probability information. They defined a notion of risk for each attack scenario, which is based on quantitative information added to attack graphs. They also proposed a method to add CVSS information in attack graphs. It allows to compute damages and exploitability probability assessments for both hosts and network in attack models. However, this method is very complex and not suitable for large networks. Keramati et al.[13] propose a method that can measure the impact of each shown attack in the attack graph on the security parameters (Confidentiality, Availability and Integrity) of the network. In the proposed approach they have defined some security metrics by combining CVSS framework and attack graph. The security metrics can address the issue that existing approaches donot consider interrelation between vulnerabilities of the network efficiently and help the testers to assess network metrics quantitatively by analyzing attack graphs. However, this method failed to solve the problem of circle path and combination explosion.

We learn from the above methods that some methods can only generate one attack path and cannot meet analyze the network security; some methods are out of date and not applicable for practical scenarios; some have a possibility of leaving out the import attack paths; some do not consider the probability of exploitation of each attack path and some failed to solve the problem of circle path and combination exploitation. This paper proposes a penetration graph generating method that optimize the network topology before generating the penetration graph, which can reduce the redundant information effectively and provide reference for testers.

III. Generation Algorithm Of Penetration Graph In Penetration Testing

Before introducing the generation algorithm of penetration graph in penetration testing, some elements in network should be defined abstractly, including node, vulnerability information, reachability matrix and atomic attack.

A. Related definitions

- Definition 1 (Node): Node is an element containing potential security problem in the network. A node can be a server, router or a hardware firewall, which contains a lot of information. However, when constructing the attack graph, we just need to know the service provided by the node. The service can be Ftp service, mail service or other similar services. We can use a quadruple to represent a node as follows:

  $$\mathrm{NODE} ::=< ID, cve\_num, service, protocol, port, subnet > \quad (1)$$

  ID is an unique identifier for the node in the whole network; cve_num is an unique identifier for each vulnerability organized by CVE; service refers to the service in which vulnerability exists; protocol refers to the protocol used by the service, usually the TCP/IP protocol; port refers to the port used by the service. For example, HTTP service corresponds to port 80 and FTP service corresponds to port 21; subnet refers to the subnet segment of the node.
- Definition 2 (Vulnerability information): Vulnerability information refers to the defect

information on the node that can be exploited as follows:

$$VUL\_INFO ::=< cve\_num, cvss\_score, range, con\_privilege > \quad (2)$$

cve_num is the same as definition 1; cvss_score is designed by Common Vulnerability Scoring System(CVSS) to evaluate weakness involved in a vulnerability and potential impact, including: cvss-v2 base score, impact subscore and exploitability subscore. We use cvss-v2 base score here, whose range is 0-10. There are four levels of risk: low, medium, high and critical. 0.1-3.9 is low, 4-6.9 is medium, 7-8.9 is high and 9-10 is critical. According to attack range of the vulnerability, they can be divided into local access, adjacent access and network. Local means that the vulnerability is only exploitable with local access, and testers must have physical access to the vulnerable system or access to a local (shell) account. Adjacent means that the vulnerability is exploitable with adjacent network access, and testers must have access to either the broadcast or collision domain of the vulnerable software. Network means that the vulnerability is exploitable with network access and the vulnerability is remotely exploitable; con_privilege is the privilege of testers after exploiting the vulnerability, and only privilege elevation vulnerability can change its value.

- Definition 3 (Reachability matrix): Reachability matrix represents the services connectivity of nodes in the network. The whole service connectivity of a network can be expressed by a reachability matrix A. For example, there are N nodes providing M services in the network. The reachability matrix A can be expressed as follows:

$$A = (a_{ij}),\ a_{ij} = \begin{cases} 1, & host\ i\ can\ use\ the\ service\ of\ host\ j \\ 0, & host\ i\ can't\ use\ the\ service\ of\ host\ j \end{cases} \quad (3)$$

And A satisfies that $0 < i \le N-1$, $0 < j \le N-1$, $\sum_{i=0}^{N-1}\sum_{j=0}^{N-1} a_{ij} = M$. For example, if $a_{12} = 1$, it means that node 2 can use the service of node 1. The service can be FTP service or other services. If a node provides two services, then the node will be seen as two nodes, and the services provided by itself can visit each other.

- Definition 4(Atomic attack): Atomic attack means that testers exploit a vulnerability of the node. The goal of an atomic attack is to obtain the new privilege of the target, which can be expressed as follows:

$$ATOM\_ATTACK =::< dst\_ip, port, src\_pri, dst\_pri > \quad (4)$$

dst_ip represents the target node of this attack; port represents the port which has vulnerability; src_pri represents the privilege of testers on the target node before exploiting the vulnerability. Testers must have at least access privilege to initiate an attack; dst_pri represents the privilege of testers after exploiting the vulnerability.

B. The generation algorithm

Based on the above definition, we can construct the flow chart of automatic generation algorithm of penetration graph in penetration testing as Figure 1.

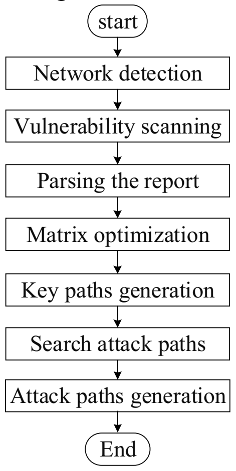

Figure 1. The algorithm of flow chart

We can see from Figure 1 that the algorithm contains seven steps as explained below:

1. Detect the surviving node of the target network, obtain the reachability matrix A of the target network, and divide the target network into the subnet. The division method is shown as: number the nodes of the Matrix A in the sequence from the outside to the inside. The outermost is the subnet 0，followed by the 1st…and so on. Each subnet contains at least one node.
2. Scan the vulnerability of the target network and derive the scanning report.
3. Parse the scanning report and show the atomic attack information. First, get the cve_num, ip address and port of the vulnerabilities from Nessus scanning report; second, search the corresponding cvss_score and range by cve_num in vulnerability database. According to the above definition of cvss_score, we assume that when the range of a vulnerability is network and the cvss_score in the range 7-10, the testers can get the administrator privilege of this node; third, filter the cve_num, ip address and port from the data obtained from the first step to satisfy the assumption; finally, use the data obtained from the third step to generate atomic attack information, the value of dst_ip and port is set to be ip address and port, the value of src_pri is set to be the privilege of testers on the target node; the value of dst_pri is set to be admin. Then we get the atomic attack information from the nesses scanning report.
4. Optimize the reachability matrix A according to the atomic attack information generated in the third step. a_{ij} is the element of the reachability matrix A . if $a_{ij} \neq 0$ and atomic attack does not exist in the node j, then set a_{ij}=0 . The optimization is aimed at simplifying the generation of attack graph, avoiding generating

redundant attack information. Denote the optimized reachability matrix as A_1.

5. Obtain the key path of A_1 and get the transpose matrix of A_1, which can be denoted as A_1^T. Then compute $A_1 - A_1^T$. The elements in A_1 are marked as a_{ij} and the elements in $A_1 - A_1^T$ are marked as b_{ij}. if $a_{ij} = 1$ and $b_{ij} = 0$, it means the firewalls and routers do not filter the connection between node i and node j. For all $b_{ij(i \le j)} \ne 0$, obtain the atomic attack paths in its subnet. These paths are called key paths and nodes in the paths are called key nodes.
6. Search the attack path of A_1 from the testers to the target. Pseudo-code is shown as follows:

Algorithm Attack path search algorithms

```
INPUT: the network matrix
OUTPUT: the attack paths
1.  Nodestack.push(0);
2.  VertexStatus[]={0};
3.  ArcStatus[][]={0};
4.  Paths={};
5.  VertexStatus[0]=1;
6.  while (!Nodestack.empty()) do
7.      int elem= Nodestack.top();
8.      if(elem==N) do
9.          path=Traverse(Nodestack);
10.         Paths.add(path);
11.         VertexStatus[elem]=0;
12.         UpdateArcStatus();
13.         Nodestack.pop();
14.     else    do;
15.         i=0;
16.         while(i<N+1) do
17.             bool b1= (VertexStatus[i]=0);
18.             bool b2=(ArcStatus[elem][i]=0);
19.             bool b3=(Arcs.contain(elem,i));
20.             bool b4=(subnet(elem)<=subnet(i));
21.             if(b1&&b2&&b3&&b4)    do
22.                 VertexStatus[i]=1;
23.                 ArcStatus[elem][i]=1;
24.                 Nodestack.push(i);
25.                 break;
26.             i++;
27.         if(i=N+1) do
28.             VertexStatus[elem]=0;
29.             UpdateArcStaus();
30.             Nodestack.pop();
31.         end while
32. end while
33. return Paths;
```

The network matrix is represented by the adjacent matrix A_1 here. Nodestack represents the work stack of the nodes while VertexStatus[] preserves the status of nodes. When the node i is not in the Nodestack, the value of the VertexStatus[i] is 0，otherwise, it is 1. ArcStatus[][] preserves the status of arc. For example, the arc e has two nodes, node i and node j, when both node i and node j are outside of the Nodestack, ArcStatus[i][j] is 0，otherwise it is 1. Paths preserve the attack paths here. UpdateArcStatus () function is used to update ArcStatus [][] and set the status of the arcs that both nodes of the arcs are outside of the stack to be 0. subnet (i) function returns the subnet of the node i.

7. Combine the attack path generated in the sixth step and the vulnerability information of the node, draw the attack path and generate the penetration testing scheme based on attack graph. We can know from the above analysis that the maximum time complexity of the algorithm is O((n-2)!).

IV. The Prototype System Of PGG

According to the generation algorithm of penetration graph, build the prototype system of penetration graph generation (PGG). We can see the structure of PGG in Figure 2.

The system consists of two parts: the input of the first part is Network addresses of the target network and the output is the reachability matrix after optimization; The input of the second part is the reachability matrix deduced from first part and the output of the second part is the penetration graph. The vulnerability characteristics database stores the characteristics of the vulnerability which are used for vulnerability scanning. The vulnerability database stores the cve_num, cvss_score and range of the vulnerability, which can be used to deduce the atomic attack paths.

First, execute the network detection module to get a rough reachability matrix of the network; then execute the vulnerability scanning module and get the scanning report. Second, optimize the reachability matrix deduced from first step by the scanning report.

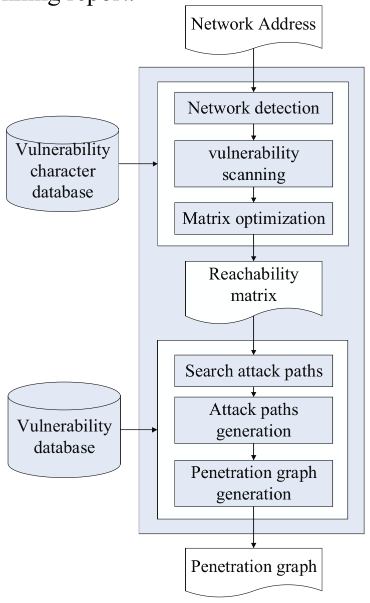

Figure 2. The structure of PGG

Third, execute the search attack paths module and the attack paths generation module. Then execute the penetration graph generation module to map the attack paths into penetration graph.

V. Experiment

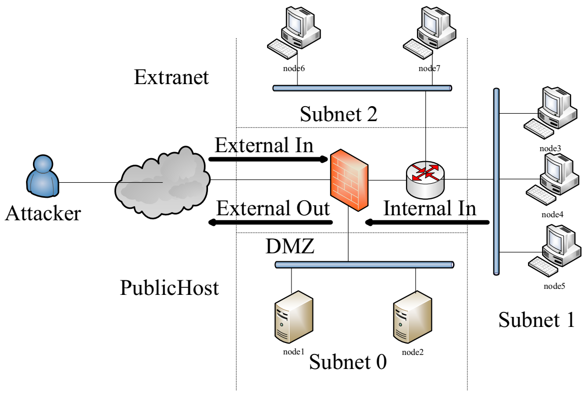

Figure 3. The Experimental topology

Figure 3 shows the network topology of this experiment. As is shown in the figure, the experiment consists of four parts: the external network that the testers stays, NO.0 subnet, NO.1 subnet and NO.2 subnet. The IP address of tester is 172.16.23.45. NO.0 is the DMZ zone of the target subnet node1(Web Server IP:192.168.1.246) and node 2(FTP Server IP:192.168.1.135) are in the NO.0 subnet; node3(open RDP service IP:192.168.2.122), node4(open WWW service IP:192.168.2.15) and node5(open SMB service IP:192.168.2.77) are in NO.1 subnet; node6(open RDP service IP:192.168.4.8) and node7(open MYSQL service IP:192.168.4.15) are in NO.2 subnet. The nodes information are shown in TBALE Ⅱ.

TABLE II. Nodes Information

ID	cve_name	service	protocol	port	subnet
N1-192.168.1.246	CVE-2013-2249	Apache	TCP	443	0
N2-192.168.1.135	CVE-2012-1899	FTP	TCP	21	0
N3-192.168.2.122	CVE-2012-2002	RDP	TCP	3389	1
N4-192.168.2.15	CVE-2012-1823	WWW	TCP	8080	1
N5-192.168.2.77	CVE-2012-0870	SMB	TCP	445	1
N6-192.168.4.8	CVE-2012-0002	RDP	TCP	3389	2
N7-192.168.4.15	CVE-2011-2688	MySQL	TCP	3306	2

Leaving out the process of network detection, we can get the reachability matrix A as follows:

$$A=\begin{array}{c|cccccccc} & 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ \hline 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 2 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 3 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 6 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 7 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{array}$$

As is shown in matrix A, There are 8 nodes in the network. Number the 8 nodes from the outside to the inside, the order from outside to inside is from number 0 to number 7. NO.0 node is the testers in the network, NO.1 and NO.2 nodes corresponds to node1 and node2 in NO.0 subnet; NO.3 to NO.5 nodes corresponds to node3 to node5 in NO.1 subnet; NO.6 and NO.7 nodes corresponds to node6 and node7 in NO.2; node7 is the final penetration target.

Scanning the vulnerability of the target network, we can see the scanning information in TABLE III.

TABLE III. Vulnerabilities Information

cve_name	cvss_score	range	con_privilege
CVE-2013-2249	7.5	Network exploitable	admin
CVE-2012-1899	4.3	Network exploitable	admin
CVE-2012-2002	8.3	Network exploitable	admin
CVE-2012-1823	7.5	Network exploitable	admin
CVE-2012-0870	7.9	Network exploitable	admin
CVE-2011-2688	7.5	Network exploitable	admin

Parsing the scanning report and obtaining the atomic attack information from the scanning report, we can see in TABLE IV

TABLE IV. Atomic Attack Paths Information

dst_ip	port	src_pri	dst_pri
192.168.1.246	443	access	admin
192.168.1.135	21	access	admin
192.168.2.122	3389	access	admin
192.168.2.15	8080	access	admin
192.168.2.77	445	access	admin
192.168.4.8	3389	access	admin
192.168.4.15	3306	access	admin
192.168.4.15	3306	access	admin
192.168.4.15	3306	access	admin

Optimize the matrix A, because every node contains atomic attack information, $A=A_1$ here.

Obtain the key path of A_1 and translate A_1 to its transpose matrix A_1^T :

$$A_1^T=\left\{\begin{matrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \end{matrix}\right\},$$

Then calculate $A_1\text{-}A_1^T$,

$$A_1\text{-}A_1^T=\left\{\begin{matrix} 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & -1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & -1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & -1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & -1 & -1 & -1 & 0 \end{matrix}\right\}$$

Element in $A_1\text{-}A_1^T$, $b_{ij(i\le j)}\neq 0$ means key path, in this experiment the key paths are $b_{01}, b_{02}, b_{13}, b_{25}, b_{47}, b_{57}, b_{67}$.

Searching the attack path of A_1 from the testers to the target, we can get three attack paths here:

node0->node2->node5->node7;

node0-> node1-> node4-> node7;

node0-> node1-> node3-> node6-> node7.

Map the paths and relevant vulnerability information into penetration graph. As is shown in Figure 4, blue parts represent the privilege of penetration tester owned on the node. For example, admin(192.168.1.135) means the testers own administrator privilege on 192.168.1.135; oval parts represent vulnerability information on the node. For example, port 8080(CVE-2012-1823) means the 8080 port of this node has a CVE-2012-1823 vulnerability; tail nodes of dashed arrows represent the service relations between node and testers. For example, Apache(172.16.23.45,192.168.1.246) means the node 192.168.1.246 provide Apache service for the node 172.16.23.45. We can see from the Figure 4，there are three attack paths. In the first path, testers exploit CVE-2013-1889 of 192.168.1.135 on port 21 to get administrator privilege, then exploit CVE-2012-0870 of 192.168.2.77 on port 445 to get privilege, and finally exploit CVE-2011-2688 of 192.168.4.15 on port 3306 to get administrator privilege. In the second path, testers get the administrator privilege of node 192.168.4.15 from the node 172.16.23.45 via the node 192.168.1.246 and the node 192.168.2.15. In the third path, testers firstly get the privilege of node 192.168.1.246 and 192.168.2.122, then exploit CVE-2012-0002 of 192.168.4.18 on port 3389 to get administrator privilege and at last exploit CVE-2011-2688 of 192.168.4.15 on port 3306.

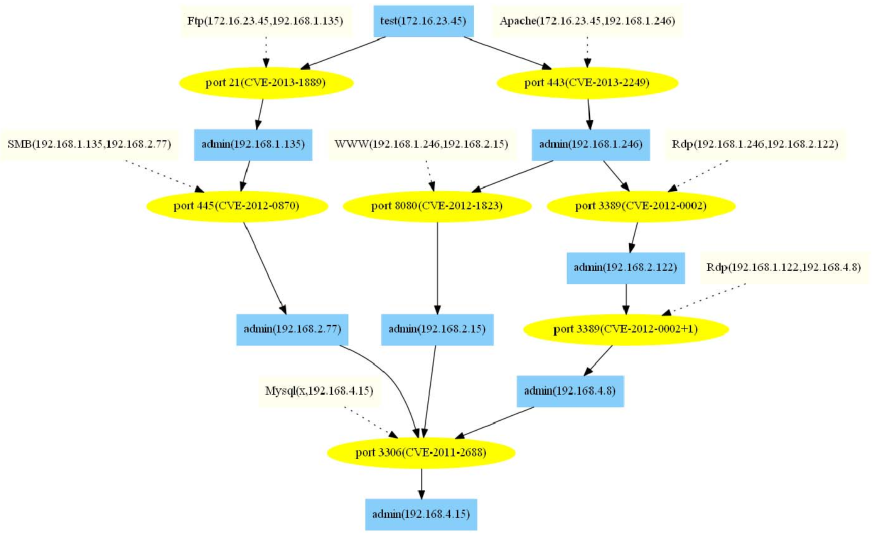

Figure 4. The result of the penetration graph

VI. CONCLUSION

This paper proposes an automatic generation algorithm of penetration graph. First, execute network detection and obtain a rough reachability matrix of the network. Then scan the network and get the scanning report. Second, parse the scanning report and optimize the reachability matrix. Third, deduce the atomic attack information from the report and obtain the key paths from the reachability matrix. Finally, search the attack paths from reachability matrix and map the paths into penetration graph. Experiment shows that the method can generate multi-path correctly and effectively, which can clearly express the structure of network, facilitate the testers' analysis of the target network, and provide reference for executing penetration testing.

ACKNOWLEDGMENT

This work is supported by the National Nature Science Foundation of China under Grant No.61170925，the Co-Funding Project of Beijing Municipal education Commission under Grant No.JD100060630, and the Project of National Ministry under Grant No.A2120110006.

We would like to express gratitude to Prof. Liangshuang Lv, Mr. An Xue and MS. XiaoYan Liang for many helpful discussions. Detailed comments by all of them greatly improve the paper.

REFERENCES

[1] Bechtsoudis, A. and N. Sklavos, Aiming at higher network security through extensive penetration tests. Latin America Transactions, IEEE (Revista IEEE America Latina), 2012. 10(3): p. 1752-1756.

[2] Sheikhi, A., et al. Distributed Generation Penetration Impact on Distribution Networks Loss. in International Conference on Renewable Energies and Power Quality. 2013.

[3] Wilhelm, T., Professional Penetration Testing: Creating and Learning in a Hacking Lab. Vol. 1. 2013: Newnes.

[4] Phillips, C. and L.P. Swiler. A graph-based system for network-vulnerability analysis. in Proceedings of the 1998 workshop on New security paradigms. 1998: ACM.

[5] Idika, N. and B. Bhargava, Extending attack graph-based security metrics and aggregating their application. Dependable and Secure Computing, IEEE Transactions on, 2012. 9(1): p. 75-85.

[6] Ingols, K., et al. Modeling modern network attacks and countermeasures using attack graphs. in Computer Security Applications Conference, 2009. ACSAC'09. Annual. 2009: IEEE.

[7] Alhomidi, M. and M. Reed. Risk assessment and analysis through population-based attack graph modelling. in Internet Security (WorldCIS), 2013 World Congress on. 2013: IEEE.

[8] Wang, L., et al., An attack graph-based probabilistic security metric, in Data and applications security XXII. 2008, Springer. p. 283-296.

[9] Wang, L., A. Singhal and S. Jajodia, Measuring the overall security of network configurations using attack graphs, in Data and applications security XXI. 2007, Springer. p. 98-112.

[10] Huang, H., et al. Distilling critical attack graph surface iteratively through minimum-cost SAT solving. in Proceedings of the 27th Annual Computer Security Applications Conference. 2011: ACM.

[11] Long, T., D. Chen and R. Song. Measure large scale network security using adjacency matrix attack graphs. in Future Information Technology (FutureTech), 2010 5th International Conference on. 2010: IEEE.

[12] Gallon, L. and J. Bascou. Cvss attack graphs. in Signal-Image Technology and Internet-Based Systems (SITIS), 2011 Seventh International Conference on. 2011: IEEE.

[13] Keramati, M., A. Akbari and M. Keramati. CVSS-based security metrics for quantitative analysis of attack graphs. in Computer and Knowledge Engineering (ICCKE), 2013 3th International eConference on. 2013: IEEE.