Volume 2, Issue 8, August 2012 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

Some Algorithms for Generating Attack Graph

Tito Waluyo Purboyo* and Kuspriyanto
School of Electrical Engineering and Informatics
Institute Teknologi Bandung
Indonesia

Abstract— Attack graph can provide clues for the network defender on how an attacker exploits the vulnerability on the network to achieve goals. System administrators use attack graph to determine how vulnerable their systems and to determine what security measures are used to maintain their systems. In a network of large and complex organizations, securing a network is a very challenging task. Attack graphs are very important in the effort to secure the network, because it can directly indicate the presence of vulnerabilities in network and how attackers use the vulnerabilities to implement an effective attack. In this paper, we will describe some very good algorithms can be used to generate the attack graph.

Keywords— Generating Attack Graph; Network Security; Vulnerability Analysis; Attack Graph Generation Algorithm.

I. INTRODUCTION

Since the number of host in network continues to grow, it becomes increasingly more important to automate the process of evaluating their vulnerability to attack. When evaluating network security, rarely enough to consider the vulnerability of isolated existence. Large networks typically contain [1] multiple platforms and software packages, and employ multiple modes of connectivity. It is undeniable that the network has a security hole that could be spared from the observation, even by experienced administrators.

The rapid growth of the internet affects the economic, political, cultural and many aspects of society. The deeper and wild internet applications, the clearer and more complex computer and network security issues are. Hackers and viruses can find more ways [2] to launch an attack with respect to the development of network technology.

When analysing the network security of a company, it is important [3] to consider the multi-stage, multi-host attacks. A determined attacker is not possible to stop the machine he first compromise, but can be expected to try to penetrate more deeply into the network by jumping from one computer to another. For this reason, configuring a secure enterprise network is a daunting task for humans. There are many potential interactions between multiple hosts and components in the network, so that the configuration of the machine will affect the security of others in the network.

Currently [5], in the field of network vulnerability analysis, there was an effective scanner to scan a single host or multiple vulnerabilities in the observed network. However, these tools only check the security hole from the isolated perspective. In order to objectively analyse network vulnerabilities, an analysis tool should be able to automatically make a systematic attack scenarios based on the vulnerability of the target network, service network, host connection and access authorization. Since the attack graph is suitable for simulating attack scenarios, network vulnerability analysis and the establishment of a defense mechanism, more and more attention is paid to them.

As an important aspect of network security, computer security evaluation through the analysis of the computer network is very important and can protect us from being attacked. Attack graphs can provide a view of the security evaluation. In [6] describes a method for generating attack graphs for network security analysis.

As computer networks continue to grow in size and complexity, assessing their susceptibility to attack is a real challenge. The combination of exploits is the typical means whereby an attacker breaks into a network. Attack graphs can describe the transition from the initial state to the target state, caused by the action of the attacker, and support the visual analysis [7].

The organization of the rest of this article is as follows. Section 2 introduces the attack graph generation. The algorithms for generating attack graph are given in Section 3. We give a conclusion in Section 4.

II. ATTACK GRAPH GENERATION

As a traditional method, the vulnerability scanning cannot directly reflect the complex attack routes on the network, so the concept of attack graphs is introduced. After analysing the host computer, the device link connection and the characteristics of attack, the next step is build a model of network security status. Attack graph is one of the important tools for security analysis [8].

Attack graph plays an important role in network security, as it directly shows the existence of vulnerabilities in network and how attackers use the vulnerability to execute an effective attack, the analysis on the attack graph or the simulation of dynamic attacks through attack graph can help us find out the vulnerabilities in network easily, and take the appropriate security measures, to strengthen the security of the network [9]. Figure 1 shows the network graph generation architecture proposed by Zhong et. al. in [9].

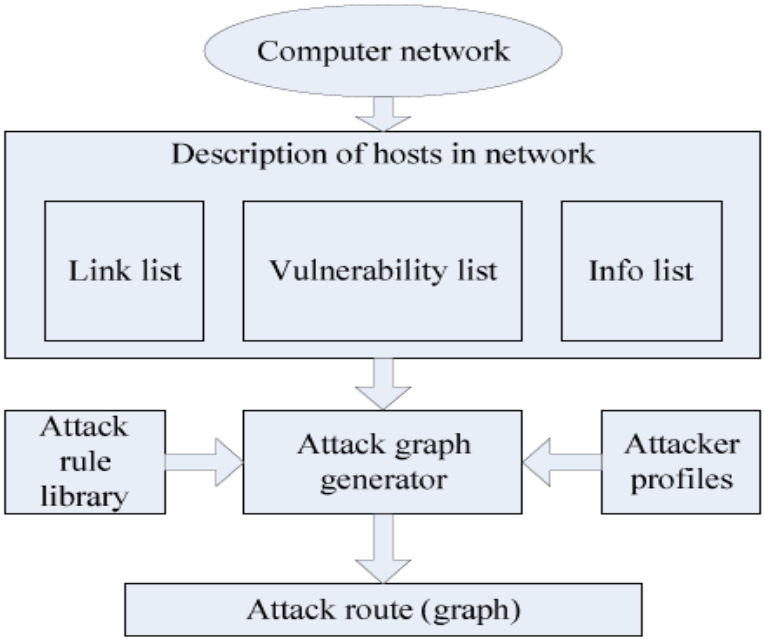

Fig. 1 Architecture of network graph generation [9]

Attack graph revealed weaknesses in the system for administrators and help them to decide which security measures should be effective to deploy. Compare with other models of attack, attack graphs are more practical. It was studied extensively and gets many results [10].

Attack graphs are a valuable tool for network defenders, illustrating paths an attacker can use to gain access to the targeted network. Defenders can then focusing their efforts on patching the vulnerabilities and configuration errors that allow the attackers to access the network [11].

Attack graph, which depicts an attack scenario, is an essential tool for analysing network security vulnerabilities. In the attack graph, a path is a sequence of attacks that an attacker can use to disrupt the network. A full attack graph shows all possible sequences of attacker's actions that eventually led to the desired network resources [12].

Attack graphs are used to determine whether a goal states can be achieved by an attacker tries to penetrate the computer networks from initial starting states. For this use, these graphs are the graphs with starting point represent an attacker at a network being evaluated. Nodes and edges describe an attacker's actions and changes in the network caused by this action. The action usually involves exploitation or exploits which use vulnerabilities in software or protocol. The purpose of this action for the attacker is to get privileges to one or more host target. The target can be a user's computer, router, firewall, or other network components. Many actions that compromise the hosts and use it as a stepping stone may be needed in large attack graph to reach the target host. A full attack graph will display all possible sequences of attacker's action which ultimately leads to the desired level of privilege on the target [13].

III. ALGORITHMS FOR ATTACK GRAPH GENERATION

Zhong et. al. in [9] adopt a positive, breadth-first algorithm for generating attack graph of network. The algorithm has basic principles as follow:

1. Save the node which represents an attacker's host into an empty Node_queue.
2. If the pointer of Node_queue not null, the host pointed to by the pointer will be considered as a host attacker to exploit this. From the host that is directly connected to the host of attacker, we can find the host which can be attacked by an attacker. If they are found, and they are not in Node_queue, we put each of them in the Node_queue. That means an attack has occurred from the attacker's host to a new host. At the same time, the pointer will point to other elements in the next Node_queue next to attacker's host.
3. Continue step (2), until Node_queue pointer is null, which means no more elements in the Node_queue. At this point, the algorithm will end. A detailed explanation of the algorithm is shown in Fig. 2.


```
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;  }
```

Fig. 2 Algorithm to generate attack graph [9]

Zhang et. al. [10] proposed algorithm as shown in Figure 3. When generating an attack graph for indicated network, the target of hosts should be determined first. Second, through scan tools and analysis, fill up above data structures of components, host, the connectivity and the exploit. Then, start from the target, find the possible attack chains. Finally, construct the attack graph with these attack chains.

Figure 3 shows the basic flow of the algorithm. The attack graph is constructed by numbers of attack chains. Since we only find the attacks which obtain a root privilege on the target hosts, the generation of attack graph is effective and simple. The recursive transfers make the algorithm easy to implement.

The hosts inside a network are divided into three types, the target hosts which are the final goals of the attackers, the middleman hosts which are the middle hosts imposed by the attackers, and the puppet hosts which are the hosts hold by the attackers already before the attack launched.

The hosts are described as a 6-tuple (HostID, HostType, AccPriv, Sw, Servs, Vuls). HostID is a unique host identifier, usually be the host name or IP address. HostType is the host type we discussed above. There are three types of hosts, the target hosts, the middleman hosts, and the puppet hosts. It can be expressed as (target, middleman, puppet). Though we cannot ascertain the goals of attackers, we know what we need to protect. The hosts we need to protect will be selected as the target hosts. All the hosts locate at our internal network will be the middleman hosts. Because the target hosts may be used as a middleman host by attackers. Those hosts locate at external network and have the privilege to access our internal network are possible puppet hosts. We suppose that attackers have the root privilege on the puppet hosts. AccPriv means the access privilege for the unknown users. For a host, there are three kinds of access privileges, (root, user, access). Sw is a list of software operating on the host, including the operating system type and version. Servs is the list of services opened on the host. Vuls is the list of vulnerabilities on the host. The CVE and bugtrap standards are considered.

The network connectivity is described as a 3-tuple (SrcID, DstID, Connect). SrcID is the source host. DstID is the destination host. Connect is the connection relations between the source host and the destination host. The connection is mainly the protocol, port, access privilege between these two hosts.

The exploit depicts the process that attackers use the vulnerabilities or services on the destination host to obtain additional privilege or services, and these additional things are beneficial to the attack. The exploit is described as a 3-tuple (Exploit, Precondition, Post condition). Exploit is the exploit name. Precondition is the precondition of an exploit. Post condition is the post condition of an exploit. It is the host state after the exploit is executed. The post condition may be the precondition of succeeding exploit.

The precondition and post condition can be formalized as follows. Precondition (SrcPriv, DstPriv, Connect, Vuls). SrcPriv is the lowest privilege of attacker on the source host launch an attack. DstPriv is the lowest privilege of attacker on the destination host attacked at present. Post condition (PostPriv, PostConn, PostVuls). PostPriv is the acquired privilege on the destination host attacked. PostConn is the connectivity affected by the attack. PostVuls is the new vulnerabilities after the attack.


```
/*
*Network attack graph generation algorithm description
*Algorithm: network attack graph generation
*Input: network attack graph model: hostInfo, connectInfo, exploitInfo
*Output: attack graph
*/

// condition judgement, recursive function
attack _condition(precondition)
{
  postcondition = check(exploitInfo, precondition);
  if(postcondition.accpriv == root)
    return true;
  precondition = postcondition;
  attack_condition(precondition);
}
// generate attack chain, recursive function
attack_chain(host)
{
  host.hosts[] = obtain_hosts(connectInfo,host);
  while(subhost = get_host(host.hosts[]))
  {
    Precondition = subhost.accpriv||subhost.sw||host.accpriv||host.vuls||host.servs;
    if(attack_condition(precondition))
      attack_chains.add(host,subhost,exploit);  // add attack chain
    if(subhost.hosttype == puppet)
      return;
    host = subhost;
    attack_chain(host);
  }
}

// main function
void main()
{
  sys_init();  // initialize system
  targets[] = obtain_targets(hostInfo);  // obtain targets
  while(target = get_target(targets[]))  // for each target
  {
    attack_chain(target);  // create attack chain
  }
  attack_graph(attack_chains);  // create attack graph
  return;
}
```

Fig. 3 Network attack graph generation algorithm [10]

Bhattacharya et. al. [4] proposed an approach which addresses the scalability issue of the attack graph generation through a generic attack path detection algorithm (see Fig. 4). This will reduce the generation of redundancy in attack graph, thus facilitating security management of an enterprise network.

Once the cycles are removed, the GenerateAttackPath algorithm (refer figure 4) is executed. The algorithm in Fig. 4 uses stack and queue as simple basic data structures. The exploit sets that belong to a single attack path and exploit sets that are applied at a particular level of attack graph are accounted by the stack and queues. The queues, for each level of stack are identified as queuei where i belong to stack pointer sp. The basic loop of the proposed algorithm [figure 4, line number 3-4] visits each exploit in the forward reachable attack graph and identifies all of its predecessor exploit (along each edge of the attack graph) and put them in the queue for future reference.

Xie et. al. [12] construct their model as a two-tier framework, whose lower tier describes all of the attack scenarios between each pair of hosts in detail, and the upper tier describes the direct access relationships between each pair of hosts. The computational cost will decrease because there is no need to generate a global complete attack graph.


```
Algorithm GenerateAttackPath (Graph g, Goal)
Input: Directed Acyclic Forward Reachable Graph
Output: Generated attack Paths
(1)  BEGIN
(2)  Initialize stack pointer sp=0.
(3)  Do
       3.1. Find all exploit set that satisty the Goal.
       3.2. Enqueue Queue_sp with the chosen exploit set.
       3.3. Choose one of the exploit set from Queue_sp.
       3.4. Push it at Stack_sp and dqueue from Queue_sp.
       3.5. Update the Goal with chosen exploit set and go to step 3.1.
       3.6. Update stack pointer as sp=sp+1.
(4)  While (all preconditions of exploit set belongs to initial condition)
(5)  Read the stack from the top and get the attack path.
(6)  Do
       1.1. If (Queue_sp is empty)
              1.1.1.   Delete Queue_sp.
              1.1.2.   Delete the stack top exploit set and update sp=sp-1.
              1.1.3.   If (sp==-1)
                     1.1.3.1.   Go to Step 9.
              1.1.4.   Endif
       1.2. Endif
(19) While (not found a non empty Queue_sp)
(20) Go to Step 3.3.
(21) END
```

Fig. 4 GenerateAttackPath algorithm [4]

Firstly, Xie et. al. [12] define the sub-attack graphs formally. A sub-attack graph or subAG is a tuple

$$subAG = (S, \Sigma, \delta, s_0, S_f, W),$$

where

S is a set of networks states,
Σ is the library of attack rules,
$\delta: S \times \Sigma \rightarrow S$ represents of the state transition rules,
s_0 is the initial state ($s_0 \in S$),
S_f is a set of target states ($S_f \subseteq S$), and
W is the weight of state transitions.

Each state transition in this model has a corresponding weight. Weight is the overall evaluation of the probability of successful transition, detectable probability, the cost of attack, the impact of attack, etc.

The host access graph or HCG is a tuple

$$HCG = (H, C, h_0, H_f, \tau)$$

where

H is a set of hosts in the network,
C is a set of sub-attack graphs,
h_0 is the host of attacker,
H_f is a set of host of target, and
$\tau: H \times C \rightarrow H$ is a set of relations of privilege transition between hosts.

The network attack model they build consists of attack states, attackers and attack rules. In order to describe the process of attack accurately, attack states in their model is a quintuple, (Prvl, Svcs, Conn, Trust, Obj). In this tuple, Prvl is the level of privilege the attacker would obtain. Svcs, Conn and Trust represent the information the attacker has obtained of services, connections and trusts respectively. Obj stores some additional information, e.g., the username and password, which consists of three parts, object name, attribute name and attribute value. The attacker is represented as a quadruple

$$Att = (Goal, CurHost, Info, Cap)$$

Where

Goal is the attack target,
CurHost is the current host,
Info is the information obtained by the attacker.

Cap describes intruder's capabilities. Cap is arranged by four elements i.e. Source IP, target IP, Privilege may be obtained, and ID of the corresponding sub-attack graph. The attack rules describe the transitions between attack states. Each attack rule is represented by a quintuple, (ID, Name, Para, Precond, Postcond). In this tuple, ID and Name represent the identity and the name of the attack rule, Para represents the parameter set of the rule, Precond represents the precondition to match the rule, and Postcond is attack result.

In realistic penetration, attackers would only choose the attack rules that may strengthen his capabilities. We take this consideration when generating sub-attack graphs. Then, a breadth-first-search generation algorithm for sub-attack graphs is shown in Fig. 5.


```
GenerateSubAG(Hs, Ht, Rl)
Input: Hs(source host), Ht(target host), R(attack rules)
Output: subAG
(1)   SubAG.states ← 0
(2)   SubAG.edges ← 0
(3)   SubAG.states.addtail(Hs)
(4)   State(Hs).processed=false
(5)   cur_state ← SubAG.states.firststate
(6)   WHILE cur_state IS VALID
(7)     FOR EACH r IN R
(8)       IF MATCHING(r.Precond, cur_state)=true
(9)         AND attacker's capabilities is strengthened
(10)          new_state ← cur_state + r.Postcond
(11)          IF new_state IS NOT IN SubAG.states
(12)            SubAG.states.addtail(new_state)
(13)            state(new_state).processed=false
(14)          END IF
(15)      SubAG.edges.addedge(cur_state, new_state)
(16)      END IF
(17)    END FOR
(18)    state(cur_state).processed=true
(19)    cur_state ← SubAG.states.nextstate
(20)  END WHILE
```

Fig. 5 Generation of a Sub-Attack Graph [12]

The algorithm takes the state of source host as the initial state, and then compares this state with the preconditions rule in the library of attack rules. If any precondition rule is satisfied, then a new state and the corresponding edge are added to the sub graph. After completing one process of comparing state, the algorithm continues to process the next state in queue. After the generation of a sub graph, all of the attack paths could be found easily by backward searching. And then, we get the minimum set of attack paths by analyzing the covering relation of attack paths. An analyzing algorithm is shown in Fig. 6.

```
FindMinimalAttackPathSet(PATHs)
Input: PATHs(the sets of all attack paths)
Output: MAPs(the minimum attack path set)
(1)   MAPs ← Ø
(2)   Paths ← PATHs
(3)   WHILE Paths IS NOT Ø
(4)     minlen ←  {the shortest length of attack path in Paths}
(5)     Minpath ← {find one path whose length is minlen}
(6)     MAPs ← MAPs + minpath
(7)     Paths ← Paths-{the attack paths which cover minpath}
(8)   END WHILE
```

Fig. 6 Finding Minimum Attack Path Set [12]

While sub-attack graphs contain the desired privilege, resources may be compromised and the corresponding attack paths, the host access graph could be built by traversing and checking all the sub graphs. A realization algorithm is shown in Fig. 7.

```
GenerateHCG(C, H, Conn)
Input: C(the set of sub-attack graphs), H(the set of hosts), Conn(the set of network
connection relations)
Output: HCG
(1)   HCG ← Ø
(2)   HCG.states.add(H.hosts)
(3)   FOR EACH SubAG IN C
(4)     Get Hs, Ht, and priviliges may obtain
(5)     HCG.edges.addedge(Hs, Ht)
(6)     edge(Hs, Ht).setattribute(privilege, ID(subAG))
(7)   END FOR
```

Fig. 7 Generation of Host Access Graph [12]

If a sub-attack graph is an attack graph between two hosts and deals with these services, vulnerabilities and the network connections of the target host then the process of generation become very fast, and the sub graph size is always small. After completing the works of generation of all relevant sub graphs, we would get the host access graph with these sub-attack graphs and the network connections between all hosts. The generation algorithm reads each sub attack graph

as input, getting the privilege transition relation, and then constructs the host access graph with all these transition relations. With the host access graph and all sub attack graphs, we can get all attack paths to target hosts in the network.

Synthesizing the attacker's starting point and object [2], host information and network topology information, the based-graph description represents the threat to security of information system, and it is called attack graph. To analyse the security of network based on the analysis of network security incidents and the action of attacker, they make some assumptions:

1. Attackers have the strong ability to attack i.e. attackers knows well the vulnerability in system so than attackers have the ability to attack these vulnerabilities.
2. Attackers are sophisticated in other words attackers will not execute an attack to obtain the previous privilege.

We use the network states to present the model information of network described in [2]. This system use a forward-search, breadth-first and depth-limited (attack steps limited) attack route producing algorithm to find the attack routes, then utilize the tools Graphviz to generate attack graph. The algorithm to produce attack route is described in Fig. 8.

```
Algorithm: NAG_Generate(M)
Input: Model information of network for security M
Output: Attack route clue
(1)  Set up the initial network state—init_state;
(2)  Add init_state into state_queue;
(3)  while (state_queue not empty && depth<Max_depth)
(4)    cur_state ← Get_State(state_queue);
(5)    If (M specified attack object && attack object have achieved)
(6)        continue;
(7)    set up host_queue that include hosts these have link protocols with the host attack
       launched;
(8)    while (host_queue not empty)
(9)      Host ← Get_Host(host_queue);
(10)     Set up protocol_queue these could be used access other host by the host launched attack;
(11)     while(protocol_queue not empty)
(12)       protocol ← Get_Protocol(protocol_queue);
(13)         set up the attack_rule_queue corresponding with protocol;
(14)         while( attack_rule_queue not empty)
(15)           attack_rule ← Get_Attack_Rule(attack_rule_queue);
(16)           attack the host according to attack_rule;
(17)           if (attack successfully && attacker's privilege escalated on the host)
(18)                 generate new network state new_state;
(19)           if (new_state don't appeared before it)
(20)                 add new_state into state_queue;
(21)           output attack route clue;
```

Fig. 8 Algorithm for Generate Network Attack Graph [2]

More detail explanation about algorithm in Fig. 8 can be explored and studied in [2].

IV. CONCLUSIONS

In this paper, we discuss some algorithms for generating attack graph. Many related research should be done in the future, the results from network scanning software should be used in the process of generating attack graph. Algorithm to generate attack graph should be made more efficient in order to attack graph creation process faster and attack graph complexity is reduced. Methods to analyze attack graph must be developed and studied further.

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. Because there is no perfect method for generating attack graph until now, modeling is a solution to the problem of making attack graph. Generating the Attack Rule Library and building model automatically are the directions of our further research.

There are many open problems in the security metrics research area as stated in [15]. In the future works, we will provide the multiple security metrics including our proposed metrics [14, 17, 18] which can be used to evaluate a network security thoroughly. We will develop a network security metrics based on path analysis in the future. We will also try to develop the network security metrics based on attack graph and its relation with Eigen pair [16] and other properties of a graph.

REFERENCES

[1] O. Sheyner and J. Wing, "Tools for Generating and Analyzing Attack Graphs," FMCO 2003, LNCS 3188, pp. 344–371, 2004.

[2] T. Zhang, M. Hu, X. Yun, Y. Zhang, "Attack Graph Generation with Implementation in Network Security," Proceeding of WESEAS 2007.

[3] X. Ou, W. F. Boyer, M. A. McQueen, "A Scalable Approach to Attack Graph Generation," CCS'06, Alexandria, Virginia, USA, October 30–November 3, 2006.

[4] S. Bhattacharya, S. Malhotra, S.K. Ghsoht, "A Scalable Representation towards Attack Graph Generation," Proceedings of the 2008 1st International Conference on Information Technology, Gdansk, Poland, 19-21 May 2008.
[5] D. Man, B. Zhang, W. Yang, W. Jin, Y. Yang, "A Method for Global Attack Graph Generation," Proc. ICNSC 2008, IEEE Intl. Conf. 6-8 April 2008 p. 236 – 241.
[6] T. Zhang, M. Hu, D. Li, L. Sun, "An Effective Method to Generate Attack Graph," Proceedings of the Fourth International Conference on Machine Learning and Cybernetics, Guangzhou, 18-21 August 2005.
[7] Y. Zhao, Z. Wang, X. Zhang, J. Zheng, "An Improved Algorithm for Generation of Attack Graph Based on Virtual Performance Node," 2009 International Conference on Multimedia Information Networking and Security, 2009.
[8] S. Khaitan, S. Raheja, "Finding Optimal Attack Path Using Attack," International Journal of Soft Computing and Engineering (IJSCE), Volume 1, Issue 3, July 2011.
[9] S. Zhong, D. Yan, C. Liu, "Automatic Generation of Host-based Network Attack Graph," World Congress on Computer Science and Information Engineering, 2009.
[10] B. Zhang, K. Lu, X. Pan, Z. Wu, "Reverse Search Based Network Attack Graph Generation," Proceedings of International Conference on Computational Intelligence and Software Engineering (CiSE) 2009, 11-13 Dec. 2009.
[11] K. Ingols, R. Lippmann, K. Piwowarski, "Practical Attack Graph Generation for Network Defense," ACSAC '06. 22nd Annual p. 121 – 130, 2006.
[12] A. Xie, G. Chen, Y. Wang, Z. Chen, J. Hu, "A New Method to Generate Attack Graphs," Proceedings of 2009 Third IEEE International Conference on Secure Software Integration and Reliability Improvement, 2009.
[13] Z. Lufeng, T. Hong, C.Y. Ming, Z.J. Bo, "Network Security Evaluation through Attack Graph Generation," World Academy of Science, Engineering and Technology 54, 2009.
[14] T.W. Purboyo, B. Rahardjo, Kuspriyanto, I.M. Alamsyah, "A New Metrics for Predicting Network Security Level," Journal of Global Research in Computer Science, Volume 3, No. 3, March 2012.
[15] T.W. Purboyo, B. Rahardjo, Kuspriyanto, "Security Metrics: A Brief Survey," 2011 International Conference on Instrumentation, Communication, Information Technology and Biomedical Engineering, Bandung, Indonesia, 8-9 November 2011.
[16] Irawati, T.W. Purboyo, "Developing Computer Program for Computing Eigen pairs of 2x2 Matrices and 3x3 Upper Triangular Matrices Using The Simple Algorithm," Far East Journal of Mathematical Sciences (FJMS), Volume 56, Issue 2, p. 185-200, September 2011.
[17] T.W. Purboyo, Kuspriyanto, "New Non Path Metrics for Evaluating Network Security Based on Vulnerability," International Journal of Computer Science Issues, Volume 9, Issue 4, July 2012.
[18] T.W. Purboyo, Kuspriyanto, "Attack Graph Based Security Metrics: State of The Art," International Journal of Science and Engineering Investigations, Volume 1, Issue 7, August 2012.

