Attack Graphs Representations

Mohammed A. Alhomidi and Martin J. Reed
School of Computer Science and Electronic Engineering
University of Essex
Colchester, United Kingdom
Email: {malhom, mjreed}@essex.ac.uk

Abstract— **Attack graphs have been widely used to represent and analyze security attacks. More specifically, they show all ways of how an attacker violets a security policy. Most attack graphs are constructed from nodes (vertices) and edges (arcs). Since there are so many research papers, each has a different representation of attack graphs. This paper discuses attack graph representations in terms of its nodes and edges interpretations. Other factors are addressed such the attack graph constructions and scalability.**

Keywords:attack graph; security; attack state; exploits; vulnerability

I. Introduction

Security attacks have always been a challenge to organizations and governments. In fact, security and network administrators face a critical job in making sure that their networks are secure against all types of attacks. To do such challenging tasks, they have to identify all vulnerabilities in their networks so that attackers can never exploit them and violet their security policies. Attack graphs can be very useful for representing vulnerabilities and exploits. Also, they can be used to carry out a series of analyses to aid this task such as showing the number of attack paths, calculating the loss value, and computing the likelihoods of successful attacks. Many research papers have investigated attack graphs for different objectives: first of all, generating attack graph is the process of constructing a graph based on some initial information including hosts, vulnerabilities, network connectivity and services running on hosts. Second, hardening networks depends on representing all hosts in the network as well as the relationships between these hosts. For example, security and network administrators can study how many ways a machine may be compromised so they act upon it and protect this machine. In addition, detecting potential intrusions can be easy to determine using attack graphs [1, 2]. To illustrate this, attack graphs represent likely attack steps so that attack graph edges are matched to IDS alerts. Last, carrying out a cost-benefit analysis is applicable to attack graphs. For instance, the attack graph presents the least cost for deploying a set of countermeasures in a given network. Furthermore, improving overall risk of a network by analyzing attack graph states especially evaluating the success probabilities moving from an attack state to another [3].

II. Represetation of attack graphs

A. *Graph-Based Attack Graph:*

The node represents a possible attack state, but an edge represents an exploit. An *exploit* is a piece of codes or commands that take advantage of vulnerabilities [4]. A *state* is either a specific machine attributes or a set of network attributes. These attributes may include hosts (machines), the connectivity of hosts, services running on hosts, access control information on hosts. For instance, the most early research papers that discussed the concept of attack graphs are [5, 6]. They presented a graph-based model for generating attack graphs. This approach represents nodes as network states and edges as possible state transitions. The inputs of this model include attack templates, configuration files and attacker profiles (e.g., access to a specific computer resource). The edges of the attack graph are weighted with costs or probabilities of success, while the nodes are encoded in the attack templates. This model showed how an attack graph can be analyzed to find out the shortest attack path with the highest probability to success. These approaches also helped to group and analyze similar nodes which have almost the same attributes and configurations. However, they suffered from scalability problems such as generating a graph with a lot of states. An alternative model is addressed in [7] where the node represents an exploit, the edge represents either a condition or exploit or privileges gained after the exploit. These research papers stated that the approach can find all attack paths where each attack path consists of an attack scenario and aimed to analyze the security of network. An alternative model [8] discussed a probability based approach to generate attack graphs. The paper developed an attack model using a threshold and key states to make the graph scalable. It introduced prior probability, match probability, and transitions probability. To define them in sequence, the prior probability means the likelihood of the attacker reaching the ultimate goal. The match probability describes the matched pre-conditions. The transition probability represents the likelihood of the attacker reaching a state from the initial state. In particular, the attack model consisted of network resources, attack scenarios, and attack rules. The attack model algorithm begins with the attacker's host as an initial state. Then, it matches the initial state with all pre-conditions rules in the rule library. If any rule is matched, a new state with the corresponding edges is added to the attack graph. After the model has generated the attack graph, all attack paths can be presented by a backward search.

The output of the attack graph is managed by a threshold and key states to keep the attack graph scalable. In short, this model shows how to use attack probabilities as knowledge base to construct attack graphs.

B. *State Enumeration Attack Graph:*

The node represents an entire state of a network. On the other hand, the edge represents the state transitions. As an illustration, [1, 2, 9] proposed a symbolic model checking technique to automatically generate attack graphs. This attack graph generation has been later called a *state enumeration graph* [10]. The inputs of this model are the vulnerabilities on network hosts, connectivity between network hosts and initial capabilities of the attacker. These are considered as network states and the exploits as transitions from one state to another. In practice, the model checker is given the goal state of the attack. Then, it produces the attack graph. Last, it checks if a sequence of exploits can lead to the goal state of attack. Therefore, the model checker revels what vulnerabilities must be fixed or patched. Even though these approaches represented attack graphs, they faced redundant attack paths [11]. Therefore, the model checker performs lots of unnecessary computations for the redundant paths. Analyzing attack graphs in [2, 9] showed how to find a minimum set of atomic attacks that must be removed to stop the attacker from achieving his/her goal. This brought up the problem of finding all possible paths of the attack as well as the problem of growing exponentially in size.

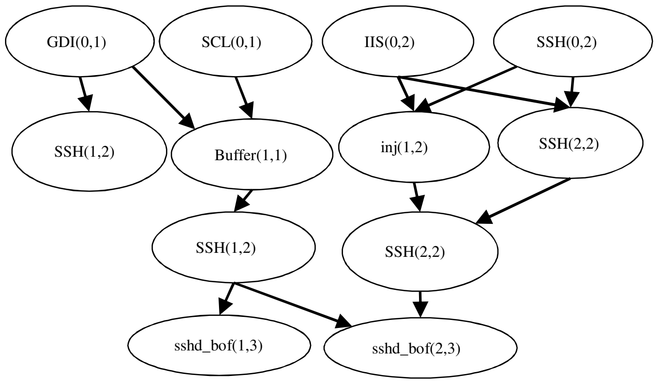

Figure 1. Example of a state enumeration attack graph

Figure 1 gives an example of a state enumeration attack graph. This example shows that the attacker starts from machine 0 and continues exploiting until machine 3. It is seen in the attack graph that there are some redundant states and this can easily lead to a state explosion.

C. *Coordinated Attack Graph:*

The node represents a system state and the edge corresponds to an action. For example, [12] proposed a model called as a *coordinated attack graph.* The concept is to group several attacks to accomplish a common malicious attack. This approach differs from the state enumeration attack graphs [10] by considering a group of attacker as a single attacker. The attack is based on concurrent action attack when the attacker could only reach the goal if properly coordinate with other actions such as a distributed denial of service (DDoS) and then transitioned into some states. To represent this attack model is to have a joint attack space and the concurrent actions can move the attack to other states using the technique of finite state machine (FSM). Despite the unique representation of visualizing coordinated attacks, it still suffered the same problem of growing in size as in state enumeration attack graphs. To improve attack graph generations using model checking, a host-centric model checking proposed [13]. This approach employed a host-centric state representation as well as the assumption of monotonicity. The node in this model represents a network state specified by network attribute values such as hosts, topology and configurations, while the edge represents state transitions. This model can generate all possible paths to reach the attacker's goal. On the other hand, the model checking technique in [2, 9] does not have the capabilities to test specific properties against the network model. Unlike the state-enumeration attack graph, the host-centric attack graph is generated in polynomial time.

D. *Dependency Attack Graph:*

The node represents a condition state of system settings (privileges), the edge represents a causal relation between those conditions. For example, [14] introduced the monotonic assumption which means that the precondition of a given exploit is never invalidated by the success of another exploit. This model improved some of the scalability problems of the model checking technique and presented almost the same data contained in state enumeration graphs but in a dependency attack graph. The dependency attack graph reduced the complexity of generating attack graphs from exponential to polynomial in the number of hosts. The approach modeled all attack paths starting from an initial state. Then, the exploits are transferred further depending on pre-conditions and post-conditions. The attack graph generated using two searches that the first links all exploits from the initial state to the attacker goal, then it searched backward to remove irrelevant state attributes. In contrast, [10, 15, 16] provided *an exploit dependency attack graph.* The nodes and edges are represented as the same as the dependency attack graphs but with no conditions. The exploit attack graph tends to have unlabeled edges. The models starts with an initial state(s) and goal state(s) in which the initial states are exploits requiring some post-conditions and the goal states are exploits requiring some pre-conditions. These models showed how the attacker can combine several individual vulnerabilities to completely attack the network. The authors [10, 15, 16] reported that their framework is computationally scalable.

E. *Full and Predictive Attack Graph:*

The node represents hosts, but the edge represents vulnerabilities. To illustrate, [17] proposed a model called *a full attack graph.* The full attack graph starts at the attacker host and then the graph visualizes all possible sequences of attacker actions. It is generated based on either a breadth-first search or depth-first search. In practice, the algorithm begins by having a node representing the attacker's starting location as the root node. Then, the algorithm checks for the reachability of nodes from the root node using a queue. As soon as, a node (host) is reachable, an attempt is made to compromise this host using all available vulnerabilities. Once the vulnerability is exploited and the targeted host is not contained on a node on the path from the attack graph root to the current node, the exploit is added to the attack graph. Whereas this approach works for all real networks, it generates redundant paths. Another graph attack model proposed by [17] called *a predictive attack graph.* The predictive is a graph that predicts the effect of eliminating the edges (vulnerabilities) nodes from the predictive attack graph. To remove the edges is to patch the vulnerabilities and having firewall rules. This process is referred to as Dynamic Pruning and is used to remove redundant paths. Adding firewall rules has some effects on the process of generating the predictive attack graph because it takes the attacker's actions and split them into two stages: first, a direct-comprise stage compromises the hosts that are reachable through the firewall; second, an indirect-compromise stage uses the directly compromised hosts to attack the rest of the vulnerable hosts behind the firewall. The predictive attack graph is as the same as the full attack graph, but with no redundant paths. The drawback of the predictive attack graph is that it grows exponentially with the size of network topology (hosts). This is a result of adding firewall rules. Another alternative approach to the predictive attack graphs is called *a node-predictive attack graph* [17]. This approach offers to overcome the problem of growing exponentially for the predictive attack graph. The node still represents either hosts or groups of hosts. The authors used a technique called Dynamic Host Collapse (DHC) that puts the nodes in host groups based on their similar vulnerabilities. If there are two hosts that could be attacked on the same level from all hosts which the attacker has already attacked, then the two hosts should be placed in one group Thus, the attacker can easily compromise a host group by attacking one single host in the same group. In other words, the group of hosts is considered as a single host. Although the node-predictive attack graph proved to run faster than the predictive attack graph, it only provides a few details about each single host in each group of hosts. Therefore, it is very challenging to protect each single host.

F. Host-Compromised Attack Graph:

While the node represents the network state, the edge represents the application of an exploit. For instance, [18] discussed the concept of *host-compromised attack graph.* The authors took advantage of data produced by a penetration test to construct the host-compromised attack graphs. This model actually was proposed to avoid the scalability problem of growing exponentially in the size of network. In practice, the model generates an *access graph* with a node for each host in the network. Then, it adds directed edges between hosts to show that there is a trust relationship between these hosts. The model only takes into account the highest level of access or more powerful attacks. Any host may exploit other trusted hosts to gain access into these hosts. Next, the model finds the maximal obtainable level of access between all hosts in the network using each host's known exploits. Now, constructing access graphs helps an analyst to determine hosts which have weak access and harden their access. This model grows polynomial with the number of hosts in the network. All attack graphs discussed in [17, 18] do not consider any goals for the attacker. An alternative model of [18], introduced in [19] has the node as the network state, but the edge as the attacker's action. To model the network, the authors used a forward-search, breadth-first and depth-limited algorithms to produce attack graphs. The authors also wanted to find all attack paths (routes) as well as the all minimal attack routes. A minimal attack route is when the attack completely relies on a previous attack on the attack route. In their experiment, the attacker begins at the initial state with the highest privileges on a given machine. Next, the attacker continues compromising hosts until having root privileges on the target machine. Another similar model discussed in [20] represents the nodes as network states and the edges as the attacker action. This research paper addressed the problem of having too many attack paths while generating attack graphs. Two ideas to solve this problem are presented: the first one is to limit the number of attack steps. The second one is to calculate probabilities of attack paths. A breadth-first search algorithm is used to generate the attack graphs. The model identifies the potential attack sequence based on the causal relationships between these attack actions and finds the advanced attack strategies of attackers.

G. Topological Vulnerablitiy Attack Graph:

The node represents a condition, the edge represents an exploit. For example, [21] proposed a model called *a topological vulnerability attack graph*. The topological attack graph representation has been later used to develop a tool called *Topological Vulnerability Analysis (TVA).* The authors [21] aimed to solve the problem of analyzing how the attacker can use low-level vulnerabilities to meet the attacker's goal. Their solution is to model networks in terms of their security conditions, model attacker's exploits as transitions between these security conditions and compute combinations of atomic exploits that lead to a given network resource. In particular, the model builds the dependency attack graph through a two-step process. The first process is to have the set of all exploits that are successfully used by the attacker. Then, the attack graph starts with an initial condition exploit. To move from a state to another a pre-condition and post-condition are matched. Consequently, the graph shows a forward dependency from the initial condition to the attacker's goal. Next, the model builds the forward dependency graph backward starting from the attacker goal exploit to the initial exploit. This results in finding all relevant exploits which are not included in the forward dependency graph. In other words, the model visualizes all possible attack paths. [22] added some improvements over the TVA tool by populating TVA models through automated agent-based network discovery, asset management, and vulnerability reporting technology. The authors aimed to avoid active network based vulnerability scanning (e.g., Nessus) in TVA. They also aimed to provide a high-level abstraction model that reduces complexity and improve scalability. [23] stated some other benefits of the TVA modeling. The authors [23] discussed how TVA gives better understanding of how individual and combined vulnerabilities affect the network security. The authors managed to transform raw security data into a model of all possible attack paths into a network.

H. Host-Based Network Attack Graph:

A node represents a host(s), an edge represents an exploit. As an illustration, [24] proposed a model referred to as a *host-based network attack graph*. The model is simply composed of three parts: first, attack profiles that are taken from a security analyst. Second, a description of hosts in the network that include the connectivity between hosts in the network, the vulnerabilities of each host, and some other information of each host. Last, an attack rule library containing all existing attack rules which are used to generate attack graphs. To explain how the model can exploit hosts is to have attackers use vulnerabilities in compromising hosts. Particularly, the algorithm of generating the attack graph starts by exploiting a host which has vulnerabilities and then, it uses this host's resource to attack another host until the attacker's goal is reached. The algorithm used a positive breadth-first search. The authors stated that the algorithm has better running times and space complexity than algorithms in [14, 19, 21]. A variant model is proposed in [25] and referred to as *a two-layer attack graph*. This research paper aimed to solve the problem of scalability under the assumption of monotonicity and evaluating the overall security of a network. The model still represents the node as a host, but the directed edge represents the access relation between two hosts. The model is composed of a lower-layer which describes all of details scenarios between each host-pair. The upper-layer presents all the direct access relations between each host-pair. The host-pair graph shows the attack sequence directly from one source host to one target host and how the attacker gains a root or user access. This research paper claimed that the model takes polynomial time and has less computational cost than the host-compromised attack graph [18]. Another similar model is discussed in [26] are called *an access attack graph*. Both the two-layer attack graph and the access attack have the same representation of nodes. However, the edges in the access graph are two types: a direct edge that shows the access available between two given hosts based on a trust relationship between these two given hosts or an exploit against a known vulnerability. The other type is an indirect edge that represents the access available between two given hosts based on a chained set of exploits of trust relationships on many hosts. The access graph generation consists of three steps: First, find access by trust is to calculate the direct edges between each host's pair using each host's trust relationships. Second, find access by exploit is to introduce each host's known exploits into model. Last, find access by trans is to calculate indirect edges by examining the chained set of exploits of trust relationships on many hosts. An alternative model presented in [27] addresses the scalability problem in attack graph generation. In particular, the model constructed a two-tier attack graph framework, which included a host access graph and a sub-attack graph. It also applied the assumption of monotonicity. The two-tier framework consisted of a lower tier and upper tier. Both the lower and upper tiers described all the attack scenarios between each pair of hosts and all the directed access relationships between each pair of hosts. The sub-attack graph is built according to attack states, attackers and attack rules. The paper pointed out that the sub-attack graph is used to represents the host's services, vulnerabilities, and connections between two given hosts. The paper concluded that the model is generated in polynomial time.

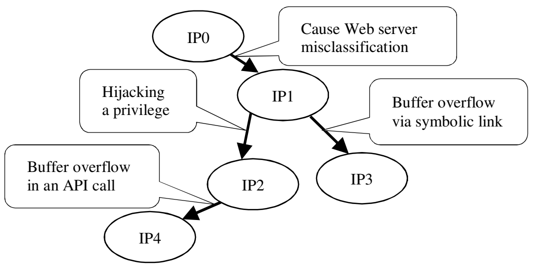
IP0
Cause Web server misclassification
Hijacking a privilege
IP1
Buffer overflow via symbolic link
Buffer overflow in an API call
IP2
IP3
IP4

Figure 2. Example of a host-based network attack graph

I. Multiple-Prerequisites Attack Graph:

The node represents one of three types (a state, prerequisite, or vulnerability); the edge represents the transition between the three types of nodes. For example, [28] introduced a model called as *a multiple-prerequisites attack graph*. The state node represents the attacker's level of access on a given host. In contrast, the prerequisite node represents a reachability group or a credential. Last, the vulnerability node represents a specific vulnerability on a particular port. In practice, a breadth-first technique is used to construct the multiple prerequisite attack graphs. Each node is shown only once in the attack graph if the attacker manages to compromise it. To successfully compromise a node, the attacker must reach at least one host already accessed and also must have obtained all credentials required by the vulnerability. The authors develop a simplification algorithm for the multiple-prerequisite graph to avoid the graph becoming very large and having many cycles. A similar model introduced in [29] which used three types of nodes (square, circle, and diamond). Whereas a square node represents security conditions in a network, a circle node represents exploits which cause new conditions, a diamond node represents penetrated conditions of machine or target resource. However, the edge represents the transitions between the three types of nodes. The model is developed based on "divide and concur" approach which addresses the problem of scalability in attack graphs. In practical, the connected attack graph is divided into multiple sub graphs and then a cut set is applied. In sum, the approach reduced the workload for risk assessment process.

J. Logical Attack Graph:

Whereas the node represents a logical statement, the edge represents the causality relations between network configurations and the attacker's potential privileges. For instance, [30] discussed a model referred to as *a logical attack graph*. It is mainly based on the causality relations between system configurations and the attacker's privileges. In fact, the logical attack graph is a goal-oriented graph that has two types of node: derivative nodes and fact nodes. The fact node has two types: a primitive fact node and derivative fact node. The first type does not require a pre-condition, while the second type requires a pre-condition. All attacks must be expressed in a propositional formula in terms of network configurations parameters. The logical attack graph originally based on a tool called MulVAL [31]. It also presents the attacker goal as the root node focuses on the root causes of the attack. The authors stated that they wanted to prove efficiency as the size of the logical attack graph is always polynomial in the size of the network. Thus, it is possible to obtain all attack scenarios using a simple depth-first search. Another logical model discussed in [32]. While the node represents an asset, the edge represents one of two logical relations: “OR” condition can be enabled by any one of its out-neighbors. An “AND” condition requires all pre-conditions of its out-neighbors to be true. This paper aimed to use dependency attack graphs to compute a numeric value showing the importance of each attack asset to an attacker. To make this, an asset ranking was introduced to precisely model the various likelihood of the attacker gaining privileges. MulVAL was used to generate the dependency attack graphs. Figure 3 illustrates a logical dependency attack graph. The AND node is shaped as an ellipses. The OR node is shaped as a diamond and vulnerability node is shaped as boxes. It’s shown that vulnerabilities in node 1 (Privilege escalation) and node 2 (Symbolic Link) are existed in a PC (IP1). The attacker’s goal is to become a root on IP1. So, the attacker can exploit these vulnerabilities using buffer overflow or enumerate access control list. Then, the attacker moves to node 3 and 4 which requires only one of its pre-conditions to be satisfied because they’re OR type nodes. Next, the attacker moves to the goal node 5 which requires both 3 and 4 to be true because it’s AND type node.

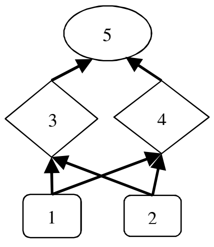

Figure 3. Example of a logical dependency attack graph

K. Goal-Oriented Attack Graph:

The node represents a vulnerability, the edge represents an action. As an example, in [33] the authors claimed that each system is reachable from each other system in a network. They also claimed that a malware can be executed by the existence of all necessary pre-conditions. Then, a serious of exploits can be used to achieve a given goal regardless of which system provided the pre-conditions. Figure 4 illustrates a goal-oriented attack graph where v1 and v2 are the start of the attack whose goal is to exploit v6. In this graph, the sequences of vulnerabilities v1, v3, v6 is an attack path. Also, v2, v4 and v6 and v2, v5, v6 are other attack paths. Even though this model generates highly scalable attack graphs, it solely depends on the ability of intrusion detection systems to detect all exploits. A similar model discussed in [34], both models have the same representation for the nodes. On the other hand, [34] represents the edge as an attack step. The authors actually described a goal-oriented approach to find out all possible steps of the attacker. The attack graph is used to present and capture the network attributes, network components and vulnerability specifications. Therefore, the weakest node can be identified and fixed. For the case of generating attack graphs, this model considers the attack graph as a sequence of goals linked together. In other words, the set of paths showed the attacks which lead to the ultimate goal.

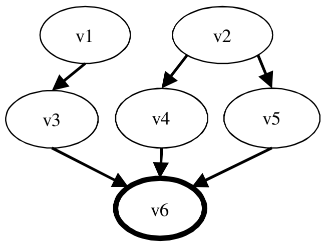

Figure 4. Example of a goal-oriented attack graph

III. CONCLUSIONS

This paper discusses previous works on attack graphs representations. Each model has its own advantages and disadvantages. Some of these approaches were developed to generate scalable attack graphs, while other approaches used the attack graph for other risk analysis such as finding the number of paths and the shortest paths. Some research papers showed that attack graphs have excellent capabilities to carry out risk assessment especially finding out the probabilities of the attacker being successful. Also, they provide attack graph as tools for intrusion detection, network attack identification and link prediction. At the moment, there is no standard model to generate attack graphs and this will remain a challenge for future works to come up with new representation models for attack graphs.

IV. REFERENCES

[1] O. Sheyner and J. Wing, "Tools for generating and analyzing attack graphs," in Proceedings of International Symposium on Formal Methods for Components and Objects, 2004, pp. 344-371.

[2] O. Sheyner, J. Haines, S. Jha, R. Lippmann, and J. M. Wing, "Automated generation and analysis of attack graphs," in In Proceedings of the 2002 IEEE Symposium on Security and Privacy (Oakland 2002), Oakland, CA, 2002.

[3] L. Huiying, "Research on Network Risk Assessment Based on Attack Probability," in Computer Science and Engineering, 2009. WCSE '09. Second International Workshop on, Chine, 2009, pp. 376-381.

[4] M. Bishop, Computer Security Art and Science Addison-Wesley Professional, 2005.

[5] C. Phillips and L. P. Swiler, "A graph-based system for network-vulnerability analysis," in In ACM New Security Paradigms Workshop, 1998, pp. 71-79.

[6] L. P. Swiler, C. Phillips, D. Ellis, and S. Chakerian, "Computer-attack graph generation tool," in In Proceedings of the DARPA Information Survivability Conference and Exposition, 2001, p. 1307.

[7] N. Ghosh and S. K. Ghosh, "An Approach for Security Assessment of Network Configurations Using Attack Graph," in Networks and Communications, 2009. NETCOM '09. First International Conference on, Chennai, India, 2009, pp. 283-288.

[8] X. Anming, Z. Li, H. Jianbin, and C. Zhong, "A Probability-Based Approach to Attack Graphs Generation," in Electronic Commerce and Security, 2009. ISECS '09. Second International Symposium on, 2009, pp. 343-347.

[9] S. Jha, O. Sheyner, and J. Wing, "Two formal analyses of attack graphs," in Computer Security Foundations Workshop, 2002. Proceedings. 15th IEEE, 2002, pp. 49-63.

[10] 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, 2004, pp. 109-118.

[11] R. Lippmann, "An annotated review of past papers on attack graphs," DTIC Document2005.

[12] S. Braynov and M. Jadliwala, "Representation and analysis of coordinated attacks," 2003, pp. 43-51.

[13] R. Hewett and P. Kijsanayothin, "Host-Centric Model Checking for Network Vulnerability Analysis," in Computer Security Applications Conference, 2008. ACSAC 2008. Annual, California, USA, 2008, pp. 225-234.

[14] P. Ammann, D. Wijesekera, and S. Kaushik, "Scalable, graph-based network vulnerability analysis," in Proceedings of the 9th ACM conference on Computer and communications security, Washington, DC, USA, 2002, pp. 217-224.

[15] S. Noel, S. Jajodia, B. O'Berry, and M. Jacobs, "Efficient Minimum-Cost Network Hardening Via Exploit Dependency Graphs," in Proceedings of the 19th Annual Computer Security Applications Conference, Las Vegas, USA, 2003, p. 86.

[16] S. Noel, M. Jacobs, K. Pramod, and J. Sushil, "Multiple coordinated views for network attack graphs," in Visualization for Computer Security, 2005. (VizSEC 05). IEEE Workshop on, 2005, pp. 99-106.

[17] R. P. Lippmann, K. W. Ingols, C. Scott, K. Piwowarski, K. Kratkiewicz, M. Artz, R. Cunningham, and M. I. O. T. L. L. LAB., Evaluating and strengthening enterprise network security using attack graphs: Citeseer, 2005.

[18] P. Ammann, J. Pamula, J. Street, and R. Ritchey, "A host-based approach to network attack chaining analysis," 2005.

[19] Z. Tao, H. Ming-Zeng, L. Dong, and S. Liang, "An effective method to generate attack graph," in Machine Learning and Cybernetics, 2005. Proceedings of 2005 International Conference on, 2005, pp. 3926-3931 Vol. 7.

[20] M. Dapeng, Z. Bing, Y. Wu, J. Wenjin, and Y. Yongtian, "A Method for Global Attack Graph Generation," in Networking, Sensing and Control, 2008. ICNSC 2008. IEEE International Conference on, 2008, pp. 236-241.

[21] S. Jajodia, S. Noel, and B. O'Berry, "Topological analysis of network attack vulnerability," Managing Cyber Threats, pp. 247-266, 2005.

[22] S. Noel, M. Elder, S. Jajodia, P. Kalapa, S. O'Hare, and K. Prole, "Advances in Topological Vulnerability Analysis," in Conference For Homeland Security, 2009. CATCH '09. Cybersecurity Applications & Technology, 2009, pp. 124-129.

[23] S. Jajodia and S. Noel, Topological vulnerability analysis: A powerful new approach for network attack prevention, detection, and response: World Scientific Press, 2007.

[24] Z. Shangqin, Y. Danfeng, and L. Chen, "Automatic Generation of Host-Based Network Attack Graph," in Computer Science and Information Engineering, 2009 WRI World Congress on, Los Angeles, USA, 2009, pp. 93-98.

[25] X. Anming, C. Zhuhua, T. Cong, H. Jianbin, and C. Zhong, "Evaluating Network Security With Two-Layer Attack Graphs," in Computer Security Applications Conference, 2009. ACSAC '09. Annual, Honolulu, USA, 2009, pp. 127-136.

[26] X. Xiaochun, Z. Tiange, and Z. Gendu, "Extended Abstract: Access Graph Based Risk Analysis for Network Information System," in Security Technology, 2008. SECTECH '08. International Conference on, 2008, pp. 129-132.

[27] X. Anming, C. Guodong, W. Yonggang, C. Zhong, and H. Jianbin, "A New Method to Generate Attack Graphs," in Secure Software Integration and Reliability Improvement, 2009. SSIRI 2009. Third IEEE International Conference on, 2009, pp. 401-406.

[28] K. Ingols, R. Lippmann, and K. Piwowarski, "Practical Attack Graph Generation for Network Defense," in Proceedings of the 22nd Annual Computer Security Applications Conference, Washington, USA, 2006, pp. 121-130.

[29] L. Jehyun, L. Heejo, and H. P. In, "Scalable attack graph for risk assessment," in Information Networking, 2009. ICOIN 2009. International Conference on, 2009, pp. 1-5.

[30] X. Ou, W. F. Boyer, and M. A. McQueen, "A scalable approach to attack graph generation," presented at the Proceedings of the 13th ACM conference on Computer and communications security, Alexandria, USA, 2006.

[31] X. Ou, S. Govindavajhala, and A. W. Appel, "MulVAL: A logic-based network security analyzer," 2005, pp. 8-8.

[32] R. Sawilla and X. Ou, "Identifying critical attack assets in dependency attack graphs," Computer Security-ESORICS 2008, pp. 18-34, 2008.

[33] S. Nanda and N. Deo, "A highly scalable model for network attack identification and path prediction," in SoutheastCon, 2007. Proceedings. IEEE, Richmond, USA, 2007, pp. 663-668.

[34] L. Xuejiao, F. Chengfang, X. Debao, and X. Hui, "A Goal-Oriented Approach for Modeling and Analyzing Attack Graph," in Information Science and Applications (ICISA), 2010 International Conference on, 2010, pp. 1-8.