Hindawi Publishing Corporation
Journal of Computer Networks and Communications
Volume 2014, Article ID 818957, 13 pages
http://dx.doi.org/10.1155/2014/818957

*Review Article*

# Use of Attack Graphs in Security Systems

**Vivek Shandilya,[1] Chris B. Simmons,[2] and Sajjan Shiva[1]**

[1] *Department of Computer Science, University of Memphis, Memphis, TN 38152, USA*
[2] *School of Computing and Informatics, Lipscomb University, Nashville, TN 37204, USA*

Correspondence should be addressed to Vivek Shandilya; vmshndly@memphis.edu

Received 22 June 2014; Revised 29 September 2014; Accepted 29 September 2014; Published 20 October 2014

Academic Editor: Tzonelih Hwang

Attack graphs have been used to model the vulnerabilities of the systems and their potential exploits. The successful exploits leading to the partial/total failure of the systems are subject of keen security interest. Considerable effort has been expended in exhaustive modeling, analyses, detection, and mitigation of attacks. One prominent methodology involves constructing attack graphs of the pertinent system for analysis and response strategies. This not only gives the simplified representation of the system, but also allows prioritizing the security properties whose violations are of greater concern, for both detection and repair. We present a survey and critical study of state-of-the-art technologies in attack graph generation and use in security system. Based on our research, we identify the potential, challenges, and direction of the current research in using attack graphs.

## 1. Introduction

Today, most of the computational systems are becoming more complex and engage their respective environments. Since the environments are neither fully controllable nor predictable, this engagement exposes the system behavior into nondeterministic spaces. This is true as much of the simple embedded systems deployed in difficult environments as any complex system providing services over the Internet. These systems can be modeled to be in a different state depending on the values of their defining variables at each instant. An execution/run is a sequence of such states. The execution of the system leading to undesired state with harmful results is referred to as a failure scenario. A failure scenario can be defined as a sequence that violates some correctness property defined over the system. The set of all possible failure scenarios is called a failure scenario graph or *scenario graph* of a system [1]. If the cause of the failure is not a benign internal fault of the system but a malicious action of an attacker, then such a scenario graph is called an *attack graph*. Each path in the attack graph leads to an undesirable state, such as one representing an intruder gaining administrator access of a file server. Initially this was to be drawn by the red team manually, which is impractical for systems with more states. Currently, there are tools to generate them with the inputs from the system and its environment.

As computational systems, simple and complex, are becoming more engaged with their environments, the system dynamics are invariably entering nondeterministic spaces. The modeling of the systems also has to account for the uncertainties and the need for the continuity of operations faced by the system [2]. The real time response strategy is essential to reducing the possibility of down time. Thus, the models have to be effective in detecting and patching the built-in security vulnerabilities in design, as well as providing the bolt-on security during runtime to design runtime monitors and actuators.

Lipmann and Ingols [3] presented a survey on attack graphs nine years ago, wherein they found most of the attempts for using attack graphs were not scalable and have many limitations. The systems reviewed were not able to consider more than 20 nodes and could not be of practical interest. The automated attack graph generation and visualization were not yet advanced. Currently, there has been much progress and many of the limitations pointed out in [3] have been overcome. This is another survey of general models of network security, titled "toward optimal multiobjective models of network security: survey" [4]. This has some relevant material. In the light of many of these important diverse developments we here present a survey and critical study of the state of art today in applying attack graphs to address security problems in network systems. We identify the following challenges in the use attack graphs for practical security systems today:

(i) sufficient representation of system parameters/behaviors in the model,

(ii) attack graph generation,

(iii) graph analyses for and formulation of security properties and violation detection,

(iv) visualization,

(v) recommendation and implementation of response strategy where we present here the recent innovations in both methodologies and technologies that have brought improvement accomplishments of these tasks.

We categorize the works based on their main contribution either in the methodology or in the technology in the use of attack graphs. In this study we observe the challenges that have been overcome, as well as those yet to be addressed. This provides the reader with potential directions for future work.

The major contributions of our work are summarized as follows.

(1) We present the role of attack graphs in the security systems and identify the central problems associated with its use.

(2) We present an extensive up-to-date survey highlighting the state-of-the-art use of attack graphs in security systems.

(3) We present a classification and critique of the works in the past decade based on their main contribution to either methodology or technology in using attack graphs.

(4) Finally we summarize the capabilities of attack graph research and present the challenges ahead.

## 2. Background

The complexity of computational systems is managed by creating models with various degrees of detail and ensuring the desired properties of those models. The interesting behaviors generated by the models are analyzed to detect problems and locate associated fixes. Thus, as systems evolve so do their modeling and analyses. In this work, we are interested in the modeling which leads to attack graphs. Let us identify the premise of such modeling.

For illustration purposes, we provide an intuitive example of a system and its attack graph. There are machine 0, machine 1, and machine 2, which contain a user's work station, a web server, and a database server, respectively. The firewall allows http and ssh requests from machine 0 across to machine 1. During the normal operation, the user makes an http request to server 1, which goes through the firewall. Server 1 accesses database server running on server 2 to retrieve the required data and communicates back to machine 0 through http. If the user attempts to access machine 2 directly, then the firewall blocks the communication. This holds true if there is a request such as an ssh request from machine 0 to machine 2, then it is considered an anomalous behavior which is blocked by the firewall. Moreover, the database on server 2 would have private data of users other than the one at machine 0. Instead a command injection attack is successfully launched on server 1 to compromise it. Then with the help of a compromised shell on machine 1, a SQL injection attack is launched on the database at machine 2. This being successful the restricted data is siphoned to server 1 and then to machine 0. This scenario is depicted with the attack graph in Figures 1 and 2.

This representation is to provide the reader with a logical aspect of an attack. The probability and cost/weight associated with each transition/edge between the states/vertices could be added. The belief/trust factor can be accounted and updated with each transition. Similarly, many other pieces of information can be added to the attack graph.

*2.1. Theoretical Foundations.* The complex systems operating in the Internet with multiple threats cannot possibly fix each of their security vulnerabilities. Thus, the systems and their behavior are modeled such that the important correctness properties can be prioritized. Idika [5] proposed the characterization of the attack graph-based security metrics to improve hardening with given budget constraints. Therefore, the limited resources allotted for security must be appropriately expended to locate and patch vulnerabilities to avoid violating the high priority properties. The modeling needs to facilitate a natural representation of infinite runs, as most systems, such as operating systems and servers operating on the Internet which run continuously. In such systems, a successful exploitation of vulnerabilities resulting in a property violation may lead to an undesirable state. A successful attack will lead the system to infinitely repeating undesired states. This is logically expressed using linear temporal logic (LTL), while systemically modeled by nondeterministic Büchi automaton or its derivative, a type of $\omega$-automaton which is an NFA taking inputs of infinite strings [1]. The power of modeling a system which generates, analyzes, and produces recommendation strategies for security measures and interactions with the adverse environment lies in effectiveness of the finite automaton it is built upon. During our survey we came across the nondeterministic Büchi automaton [1] for modeling systems to generate their respective attack graphs. For review purposes, we present the background definition of the nondeterministic Büchi automaton. A nondeterministic Büchi automaton is a $\omega$-automaton defined as a 5-tuple $A = (Q, \sigma, \Delta, Q_0, Acc)$ where

(i) $Q$ is a finite set of states of $A$,

(ii) $\sigma$ is a finite set of symbols called *alphabet of* $A$,

(iii) $\Delta$ is the transition function: $Q \times \sigma \rightarrow \mathbf{P}(Q)$, where $\mathbf{P}(Q)$ is the powerset of $Q$,

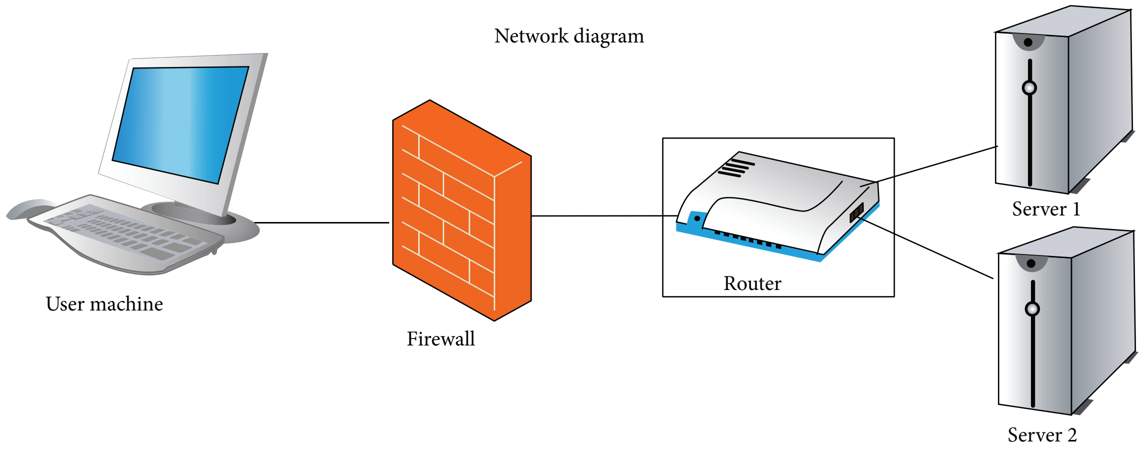

Figure 1: Attack graph example: system.

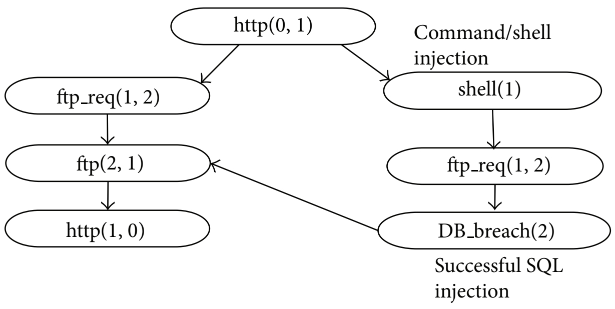

Figure 2: Attack graph example: graph.

(iv) $Q_0$ is a subset of $Q$, called the *set of initial states*,

(v) *Acc* is the *acceptance condition*, a subset of $Q^\omega$ (since the runs are infinite sequences, the infinite set of infinite sequences of states in $Q$, denoted by $Q^\omega$, will have the subset of some infinite sequences that satisfy the acceptance condition).

The run is defined by any infinite sequences $\rho = (q_0, q_1, q_2, \ldots)$ such that,

(i) $q_0 \in Q_0$,

(ii) $q_1 \in \Delta(q_0, a_1)$, where $a_1$ is the first element in the input string,

(iii) $q_i \in \Delta(q_i - 1, a_i)$, $\forall i$ with $0 \le i$.

The input is accepted only if at least one of the possible runs belongs to the *acceptance condition Acc*. In the expressiveness of *Acc* lies the power of the Büchi automaton to model the attack graphs. The violations of the correctness properties can be specified with *Acc*, and thus the systems are modeled to generate the attack graphs. Either directly the above automaton or a variation of it is used to build the different attack graph, generation, and analysis systems as we explore throughout this work.

*2.2. Practical Motivation.* There are mainly two practical objectives for the modeling. The first is the effective management of inputs and outputs of the attack graphs. The system parameters being the inputs must be effectively represented in the model resulting in the graph. The analyses of the model for different security properties, detecting violations, should find the effective responses, as outputs. The second is the ability to generate these graphs autonomously and efficiently with scalability for larger systems. There have been many attempts to achieve these objectives with various degrees of success. The handling of the state-space explosion while generating the graph, analyzing the graphs for security properties and their violations, the precision of evaluating the exact path efficiently, making practically implementable recommendation to mitigate the attacks are the main challenges that provide the practical motivation. The factors that distinguish each of the efforts to achieve these objectives are the following.

(1) What parameters of the system are used to construct the attack graph?

(2) What type of graph is constructed (graphs, trees, nets and nondeterminism involved, etc.) and what formal models they are based on?

(3) How are the graphs generated?

(4) What efforts (methods and tools used) are done for visualization?

(5) What are the properties that can be analyzed?

(6) What recommendations are obtained from the graph to secure the system?

(7) What methods were used to evaluate the work?

We study each of the considered works in the light of the aforementioned questions.

*2.3. Classification Scheme.* We present a classification of the works in the past decade, in their attempts to address the mentioned challenges. Various researches focus on developing new technologies while others focus on new methodologies. Figure 3 shows the topical classification tree. In the following section, we analyze the works and classify them based on their main contribution, while we also make note of any progress they make on the way, in other categories, as shown in Figure 3. We present the detailed analysis of prominent works in each category and a brief one of the others.

## 3. Review and Classification

In this section we present critical review of the works as shown in Figure 3. We classify them based on their main contribution to one of the four prominent problems we identify in the methodologies and technologies associated with the attack graphs. In the reviews, based on the prominence of the work, we present a general summary and identify the main contribution and how they address *the seven questions* we formulated in Section 2.2. When some works contribute to more than one aspect, we highlight the contributions under multiple categories.

### *3.1. Methodologies*

*3.1.1. Formal Modeling of Attack Graphs.* Mehta et al. [6] present an adaptation of Büchi automaton to model the system's behavior. The NFA they present is a 5-tuple with the atomic propositions defined over the systems as their accepting conditions. The security properties of the system are formulated as the atomic propositions over the system. The attack graph generated is then ranked, by assigning a rank to each state based on their reachability probability of an attacker in a random simulation by an adaptation of Google's PageRank algorithm for the web pages. They provide a full algorithm to rank a multistage cyber attack against a network of computers. They describe the procedure to construct attack graph and analyze it for security properties. The main contribution is in proposing a simpler automaton to model the system's behavior to generate attack graphs and algorithms to rank the states in them to aid analyses and visualization.

This is mainly a theoretical work.

(1) They propose the model and procedure to construct and rank attack graphs but do not specify the parameters of any specific system as example. The generality of their model allows application to diverse systems.

(2) The graphs constructed are directed graphs with nodes having in-degree and out-degree as important parameters. Each node also has an edge to itself.

(3) There is no specific mention of methods to generate graphs, but usage of any off-the-shelf model checkers for the purpose is proposed.

(4) With the help of ranks associated with the nodes, the graph's visualization is improved with better comprehensibility.

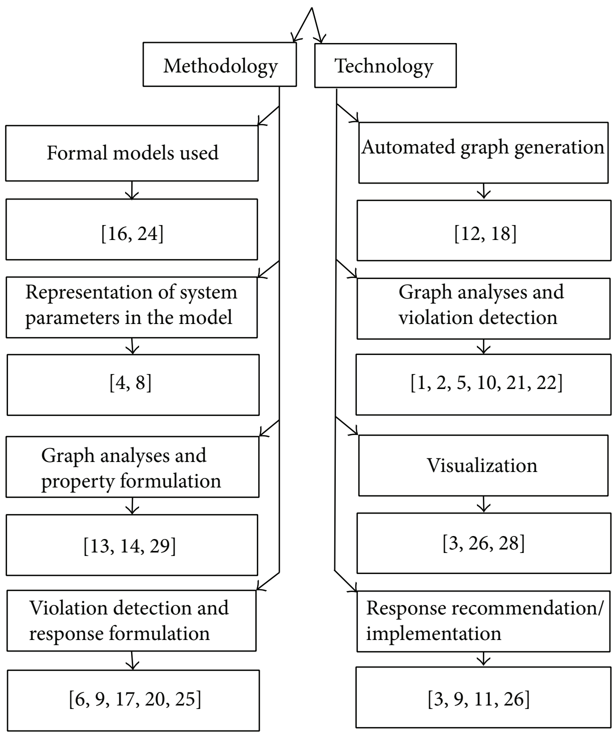
Methodology
Technology
Formal models used
[16, 24]
Representation of system parameters in the model
[4, 8]
Graph analyses and property formulation
[13, 14, 29]
Violation detection and response formulation
[6, 9, 17, 20, 25]
Automated graph generation
[12, 18]
Graph analyses and violation detection
[1, 2, 5, 10, 21, 22]
Visualization
[3, 26, 28]
Response recommendation/ implementation
[3, 9, 11, 26]

Figure 3: Classification.

(5) The atomic propositions defined over the system as the model's accepting conditions are proposed to be used to formulate the violations of the security properties. Each path is a successful violation of a security property.

(6) With their simple example they obtain the recommendation as which path must contain an intrusion detection system, to detect the intruder's attempt to gain root privilege on a target host.

(7) They have a simple network attack example to show the effectiveness of their approach and claim scalability of their methods.

Wang et al. [7] proposed the use of attack graphs and the hidden Markov model to explore the probabilistic nature between system observations and actual states. The development of this technique utilizes a middle-ware approach using a slightly modified dependency attack graph representing network assets and vulnerabilities, where the hidden Markov model captures the uncertainties of the observations to produce potential attacks and security hardening from defense candidates. Wang et al. [7] proposed system that lacks the ability to generate the graphs and incorporate the results into the system. Although mention is made of graph generation being out of scope, it serves as a starting point to ensure the observations are appropriate without administrator input, considering various starting states of an attack.

(1) The parameters used to construct these attack graphs are the network assets and vulnerabilities from the NVD.

(2) Dependency attack graphs are created using a modified multiple prerequisite graph from Ingols et al. [8], with three additional labels: solid observation (i.e., physical components and network assets), soft observation (i.e., network traffic and software applications), and dark observation (i.e., system vulnerabilities).

(3) The attack graphs are not automatically generated in this work but created manually and fed into MatLab for analysis. Wang et al. [7] use the Graphviz software to illustrate the state transitions with the key observations as the method to generate a visual representation.

(4) No particular effort is expended towards visualization of the networks. Since the suggested procedure to generate the attack graph is from tools like MulVal the visualization methods associated to them, as discussed in later works, can be applied in conjunction with this work too.

(5) The properties that are analyzed are the scores from the CVEs and the potential states.

(6) The recommendations obtained are potential attacks and security hardening measures a defender can take from candidate countermeasures.

(7) Dependency attack graphs are manually generated and the HMM-based attack graphs are calculated in the HMM toolbox in MATLAB. The methods used to evaluate the work involve manually generating a dependency attack graph with specified vertices and edges as HMM-based elements and parameters. Incoming node edges are given preconditions, whereas outgoing node edges are assigned consequences. The HMM-based attack graph is computed inside a MATLAB environment using the HMM toolbox.

*3.1.2. Representation of System Parameters in the Model.* Hong and Kim [9] propose a two-layer graph model to capture the vulnerability information of the network and provide comparative complexity analysis of security analysis in their model with that in the traditional attack graphs. They claim their model is more amenable for distributed generation and analysis and thus is more efficient. The two-layered graphs they construct have vulnerability information of the individual hosts in the lower graph while the topological information of the network is in the upper layer. They expect this to be helpful in dynamic learning of the changes in the network. This is theoretical work with no implementation. The main contribution is to gather the network's topological parameters in one layer of the graph and the vulnerability information of the individual host in another, aiding convenient analysis.

(1) The two-layer attack graph model is proposed. Individually both of the graphs make up the full graph. The layer with topological information can be nonacyclic depending on the network structure. The layer with the vulnerabilities of hosts has a directed tree structure leading to the highest valued asset with the vulnerability/prime goal of the attacker.

(2) The lower layer has the vulnerability information collating all the different (potential) processes of security interest. The lower layer of the graphs will have a root node in the upper layer which represent a single host. The edges in upper layer graph represent the network edges between the hosts.

(3) They suggest that both layers can be generated from the network using distributed processing. They claim that the distributed processing is more tenable than for traditional attack graphs as the joining of partial solutions does not have high processing overhead.

(4) The two-layered design is argued to be more promising with better visualization.

(5) No specific security property analysis is presented. But the graph traversal which is the computational workhorse of any such analysis is shown to be having algorithmic advantages in time compared to the traditional attack graphs.

(6) No security recommendations are derived from the graphs constructed.

(7) No implementation or evaluation of the work is presented. They have an analytic comparison of the computational complexity of the analysis of the attack graphs in their work with other prominent contemporary works. They plan an experimental evaluation in their future research.

Cole [10] presents a multistep attack detection via Bayesian modeling under model parameter uncertainty. The work mainly addresses the two shortcomings of the traditional IDS. The traditional IDS cannot reason across multiple attacks and does not consider the uncertainties in the parameter representation. They identify the problem to be computationally complex and thus provide heuristic solutions.

*3.1.3. Graph Analyses and Security Property Formulation.* Kotenko and Stephashkin [11] utilized attack graphs and security metrics to evaluate the security factor of malefactor's action. All objects of general attack graphs are divided into two groups, one being base (elementary) and the other being combined objects. The former being host and attack actions and the latter being objects of type route, threat, and graph. The evaluation of the security level is measured in two forms, qualitative methodologies of risk assessment and quantitative computation using Bayesian networks. The complexity of generating the attack graph is determined by malefactor's actions, the host in an analyzed network, and the quantity of vulnerabilities using Bayesian networks.

(1) Using the data from the host to construct the attack graph.

(2) Building attack graph by modeling actions of malefactor using information about network configuration, security policy, and available actions from data repository.

(3) The authors proposing a Security Analysis System, with its architecture and other details, using this system, which has an action database including *IF-THEN*

type rules, the actions exploiting vulnerabilities are tracked and an attack graph is generated.

(4) The existence a report generator in their architecture which is used to generate the reports with the details of exploits along the attack paths.

(5) The properties of being able to analyze information from hosts and internal vulnerability repositories. They are implementing the properties by grouping host and firewalls and using firewall rulesets.

(6) Various types of attacks and multistage attack on an FTP Server to gain escalated privileges.

(7) The prediction methodology used to determine the security evaluation being qualitative methodologies of risk assessment and quantitative computation of network security level (on basis of Bayesian networks, possibility theory, and fuzzy sets).

Kijsanayothin and Hewett [12] present an approach to automatic statistic analysis of the attack graphs using logical expressions and conditional preference networks. The approach yields countermeasures for the attack paths based on the preference criteria selected by the administrator. The methodology includes the steps of attack set extraction, dependency identification, countermeasure, and preference generation. The main contribution is the method to enable prioritization of the preventive countermeasures based on preferred cost criteria.

(1) The work does not prescribe any specific set of parameters to construct the attack graphs. They allude to the relevant network attributes like connectivity between source and victim hosts to be the parameters considered.

(2) No specific graph generation methodology is suggested. But the methodology for the analysis of the graphs generated is suggested. It turns out to be applicable independent of the nature of the topology of the graph.

(3) No specific graph generation methodology is considered.

(4) There is no specific direct method proposed for visualization. The creation of analysis graph and conditional preference net for countermeasures from the attack graph itself simplifies the structure containing only the relevant functional attributes for analysis which enhances visualization.

(5) There are properties specified by the preference criteria by the administration.

(6) The graphs considered are set of paths leading to a node representing a state with a security breach. The main analysis of the graph is to get the cost effective breakage of the paths leading to security breach. They show that each of the tasks in their methodology takes a polynomial time in terms of the nodes and preference criteria.

(7) Their example is a three-machine network. They build an attack graph of an attacker working across a firewall from the first machine to gain root privilege in the third machine. They construct the conditional preference net and show the partial ordering and arriving at the required countermeasure.

Xie et al. [13] present a methodology to analyze the networks for vulnerabilities using Bayesian networks. They generate a conditional probability table and emphasize the appropriate identification of the uncertainties to be represented in the Bayesian networks. They also provide sensitivity analysis for their approach.

*3.1.4. Violation Detection and Response Formulation.* Dewri et al. [14] present a methodology for security hardening of a network system based on attack trees. They formulate the problem of associating the hardening efforts with their costs and optimizing the choice of vulnerabilities to be fixed for the optimal security within a given budget as a multiobjective optimization problem. Then use nondominated sorting genetic algorithm-11 to solve it. They present attribute as a propositional instance of an attribute-template, made of salient system parameters. They construct the attack tree from the attributes and the conditions formulated based on the end-goal of the attacker. The main contribution is to propose the methodology to select the security measures to minimize residual damages within the given budget.

(1) They use the parameters of the system, such as generic property of hardware and software configuration of a network, including system vulnerabilities, network and system configuration, access privileges, and connectivity to define the attribute-templates. Using these attribute templates a propositional instance is created to get the attribute.

(2) Using the attributes, the end goal of the attacker, and the conditions of attributes that hold to satisfy a successful launch of attack are used to form a 4-tuple. They refer to this 4-tuple as the attack tree for further analysis.

(3) They prefer the idea that *once a breach happens, its effects remain forever*, to avoid the state explosion while constructing the attack tree. They do not propose any specific technology/tool to generate the attack trees though.

(4) They do not specify any particular visualization methods/technologies of the attack trees.

(5) The properties they analyze are the one they formulated to define the attribute-templates.

(6) They are able to obtain the cost-effective set of security measures to minimize the residual damages.

(7) They provide a simple 4-node example to illustrate the effectiveness of their method with an FTP server, SMTP server and Terminal and a Data server. They consider the ftp and ssh buffer overflow and host attacks (8) based on the CVE between 1999 and 2002.

Wang et al. [15] present an attack graph based security metric. They present the motivation and interpretation of the metric using two factors. The first factor is the individual vulnerabilities at each node determined by the conditions of host machines forming the network. The second factor is the activity status of the attacker. The third factor is the causal relationship between the exploitation of the individual vulnerabilities. They consider both acyclic and cyclic attack graphs. The main contribution is a breadth-first-search algorithm. The algorithm is recommended for calculating the security metric using the probabilities of the vulnerabilities and the attacks.

(1) The processes on different machines, the legal access privileges, the individual vulnerabilities of each process at each host, and the probability of the attacker exploiting each of them are used to generate the graph. These parameters are crucial for their method as it builds on them to calculate the causal relationships between them.

(2) Both the cyclic and acyclic graphs are considered with the discrete probabilities associated with the possibilities of attacks and success of exploitation of vulnerabilities.

(3) No particular graph generation method is prescribed, but whatever method used for this work must be able to evaluate the vulnerabilities and chances of attacks at each distinct node.

(4) No particular effort is suggested for visualization.

(5) The properties which can emerge by the causal relationships between vulnerabilities encoded in attack graphs are analyzed.

(6) A security metric is provided which gives a useful assessment of the security of the system. This in general gives a useful information in designing the load on networks based on the security situation.

(7) A toy example of two machine network accessed across a firewall by another machine used to attack is considered to illustrate the applicability of the procedure.

Huang et al. [16] proposed a distillation method to reduce attack graphs to simpler graphs with only the "most critical" edges and vertices. For reduction, they use a severity metric and a linear depth first search (DFS) formula transformer using Boolean clauses from the attack steps. The Minimum-Cost SAT Solving (MCSS) is used to find the most critical path relative to the least cost to the attacker to exploit the critical assets on the network. Huang et al. [16] provided a linear depth first search (DFS) formula transformer using Boolean clauses from the attack steps. Huang et al. [16] proposed distill method which reduces the attack graph by fifteen percent (15 percent) to the most vital security concerns. Although presenting a technique to reduce the graph is ideal, there lacks an intuitive way to provide a more in-depth analysis. Further, no information is provided to the recommendation of any potential defenses associated to the attack graph.

(1) They do not construct the attack graph but take the attack graph as input in aspiration to reduce the size of the attack graph.

(2) Graphs are minimized using an interactive method to reduce the size of the graph.

(3) Actual attack graph generation is not complete in this work, but Huang et al. mention the critical attack graph surface can be generated from the collected arcs and nodes using the top **k** attack paths.

(4) There are no tools presented or developed for visualization.

(5) The use of the Access Complexity from the metric vector provided by CVSS is used to derive the probability of attack success, which Huang et al. [16] use in their MinCost SAT solver.

(6) There is no information provided to the recommendation of any potential defenses associated with the attack graph.

(7) The work was conducted on an enterprise network scenario adopted from real networks, but the scalability of this work was tested on a simulated network. In the enterprise network scenario, there is the use of 5 subnets containing various publicly accessible servers, a database server, user workstations, and so forth. The total number of hosts ranged from 500 to 600 with identical configuration and a grouping method to conduct the attack graph generator.

Homer et al. [17] proposed a model to aggregate vulnerability metrics in an enterprise network to produce quantitative metrics measuring the probability an attack occurs in a given network. They utilize the key concept of d-separation in Bayesian Network inference and design customized algorithms for probabilistic reasoning on attack graphs. The Bayesian Network is a directed acyclic graph (DAG), which does not allow directed cycles, but this work correctly provides reasoning over cycles without a self-referencing effect and accounts for hidden exploit steps. They provide a cumulative metric to represent the effect of all vulnerabilities in a network, which takes into account an attacker attempting all possible paths. Noel et al. [18] present a framework to measure the security risk of networks using attack graphs. They consider both the individual vulnerabilities and also the risk incurred when many of them are exploited in tandem. Their method for quantified analysis of network security risk is by comparing the cost associated with the remedial measures to arrive at the measure which provides highest quantified confidence against residual risks for the given security budget. The main contribution is to evaluate and recommend the most cost effective remedial measure for a given set of vulnerabilities, when exploited individually and in tandem, to reduce the risk.

(1) Live network scans and databases that have the knowledge about properties such as vulnerability

likelihood, impact, severity, and ease of exploitation are used to populate the attack graph parameters. They consider the inter dependencies between the vulnerabilities to evaluate the risk, based on how an attacker would exploit them step by step to achieve a more complex attack goal.

(2) They generate for each of the different network configurations as a response of the expected threat a corresponding residual attack graph incurring different costs. This facilitates to them the analysis of the expected cost of security measures to evaluate the cost/benefit tradeoffs. They represent the risks with the conjunction and disjunction of the exploitation of different vulnerabilities to achieve an attack goal. The actual values are kept general alluding the use of Boolean variables, real numbers, or even probability distributions.

(3) They simulate the attack graphs using Monte Carlo methods, handling the highly complex logical (nonlinear) relationships among the variables involved.

(4) No special effort is made for visualization.

(5) They analyze the different dependencies among the vulnerabilities and the security risk in exploiting the vulnerabilities in combinations. They also analyze the attack graphs corresponding to different potential security measure countering the attack to evaluate the cost/benefit ratio.

(6) Their recommendations are based on return-on-investment analysis of the remedial measures. For each option against the given set of vulnerabilities, for a given budget the measure yielding lowest residual risk is evaluated and recommended.

(7) They use a toy example in [15] and present their calculations to show the effectiveness of their method.

Poolsappasit et al. [19] propose a method for risk assessment from different vulnerability based on metrics defined in Common Vulnerability Scoring System. They quantify the cost-benefit ratio for the security measures. They present a risk management framework using Bayesian attack graphs. They formulate the vulnerabilities and their mutual dependencies to construct the Bayesian attack graph. A dynamic analysis is made so as to use the latest information got in the real time. Formulating a multiobjective optimization problem based on this information to arrive at the optimal set of security measures. A genetic algorithm is used to solve the optimization problem. The main contribution is to formulate the framework for a dynamic analysis of the system, using Bayesian attack graphs and multiobjective optimization during the deployment phase of the network. This work is similar to [14] in regard to the first 5 factors we are investigating as given in Section 2.2. The dynamic analysis helps in addressing the ongoing attacks by considering the minimization of the loss from the present into the future based on the real time inputs from security devices like IDS, and so forth. They have an effective combination of asset identification, system vulnerability and connectivity analysis, and mitigation strategies. They illustrate the effectiveness of their work using a network with two systems and five servers with different security configurations. Their experiment has 13 security controls and considers 12 typical vulnerabilities capable of producing 20 different attack scenarios with different outcomes based on the public CVEs from the standard repositories. Their multiobjective optimization problem is solved effectively with a genetic algorithm which has a satisfactory convergence rate to the neighborhood of the global optimum.

### *3.2. Technologies*

*3.2.1. Automated Graph Generation.* Ou et al. [20] present an automated attack graph generation tool tested on networks with thousands of nodes. To do this they adapt the tool MulVal, a network security analyzer based on logical programming. Their method uses logical dependencies among goals of attacker and configuration information. The main contribution is to avoid the state explosion problem of the method using standard formalism of [1] and provide a scalable automated attack graph (logical attack graph) generation tool.

(1) The configuration details of the network and expected goal of the attacker along with the related cost/benefit are used to construct the logical attack graphs. The idea is to represent the configuration information using datalog tuple and the attack techniques and OS security semantics using datalog rules. The specification of pre- and postconditions for the attack formulates the attacks.

(2) The graphs generated are called logical attack graphs with two types of nodes called derivation and fact nodes. The graph is a directed tree with the root node being the goal of the attacker. The edges represent the dependency relationships. The cycles resulting due to the dependencies, leading to useless information, are removed using standard directed graph depth-first-search, giving a DAG.

(3) The MulVal tools are used to generate graphs. The MulVal reasoning is done using XSB, a Prolog system which evaluates the Datalog interaction rules on the facts about the present system given as input. The evaluating MulVal interaction rules for the graph are done with $O(N^2)$ to $O(N^3)$ number of derivation steps, where $N$ is the number of hosts.

(4) They are able to store the graph in the standard data structures. For visualization they just plot various parameters on linear and logarithmic scales.

(5) They do not explicitly analyze any properties from the graph. But the properties can be expressed by the logical attack graphs in terms of system configurations and known attack goals.

(6) There is no explicit work to generate recommendations.

(7) They generated logical attack graph stored in nearly 1 GB of memory, for a fully connected and partially connected network with 1000 hosts. The complexity being the cube of the number of hosts led to practically viable delays. Their system had a Pentium 4.3.2 GHz with 1 GB of RAM, running XSB version 2.7.1 on Microsoft Windows XP Professional Version 2002 Service Pack 2. They used the *map* template in C++ standard library to achieve their speed in the implementation.

Ingols et al. [21] present practical graph generation with the NetSPA based technology. The effort is to generate for a given network the full graph, predictive graph, and the multiple-prerequisite graph for the given networks. The computation of network reachability, classification of the vulnerabilities, graph construction, and deriving recommendations to improve the network security are achieved.

*3.2.2. Graph Analyses and Violation Detection.* Danforth [22] proposed an efficient rule-based approach for generating attack graphs through aggregating individual attacks into abstract classes, as well as clustering identical machines to reduce visual complexity. Convenient models for threat assessment in networks are proposed based on their efficient approach.

(1) The host related information is used to construct the attack graphs.

(2) The graphs are constructed and clustering is used to minimize the attack graphs for improved visualization.

(3) The graphs are generated using dot from the Graphviz project.

(4) Danforth [22] uses the expert system Java Expert System Shell (JESS) to generate the attack graph and dot from the Graphviz project is used for graph visualization.

(5) The properties based on the abstract (attack) classes of the networks are being analyzed.

(6) Danforth does not provide direct recommendation but provides what-if scenarios relative to mitigation techniques for an administrator to gain insight when potential vulnerabilities are within the system.

(7) The method used to evaluate this work involves 986 hosts.

Sawilla and Ou [23] proposed AssetRank using dependency graphs representing attacker privileges and vulnerabilities. CVSS parameters from the national vulnerability database (NVD) are used as input into the AssetRank tool. Three experiments are provided using the MulVAL attack-graph tool suite to compute a dependency attack graph. Schuppenies [24] developed both theoretical and technological improvements in Automatic Extraction of Vulnerability Information for Attack Graphs. The work provides a very thorough Bayesian Attack Graph and analysis for dynamic risk management. Cheng et al. [25] proposed a compression technique for representing attack graphs using reference encoding. Simulation is run by first generating a topological graph using BRITE and GT-ITM followed by injecting vulnerability information into the generated graphs. Cheng et al. [25] are able to generate 263 topological graphs with host ranging from 60 to 1023. A Jaccard similarity coefficient is utilized to determine the degree of similarity between vulnerabilities on various hosts.

(1) Host connectivity and known vulnerabilities were used as the parameter input to attack graph construction.

(2) Attack graphs were used to visualize attack paths.

(3) A hierarchical approach was used to generate the attack graph.

(4) There were no specific tools highlighted for construction.

(5) The properties that are analyzed are the similarity between the hosts to determine if the vulnerabilities on various hosts are similar.

(6) No recommendation techniques are used to provide a network administrator information to harden the network.

(7) The methods used to evaluate the work involve the simulation of 263 topological graphs with host ranging from 60 to 1023.

Idika [5] proposed the characterization of attack graph-based security metrics to improve network hardening relative to budget constraints. The security metrics derived within this work are network oriented. An algorithm is presented to aggregate the necessary metrics for analyzing two, or more, attack graphs. Simulated work was conducted using MulVal for attack graph generation, wherein the attack graph-based security engine is used to compute the metrics for attack graph analysis of network security. Abramov et al. [26] present a tool to analyze a network with 10000 simulated nodes with 1000 access control rules. They were able to dynamically do the security analysis to detect the breach of confidentiality, integrity, and availability properties of the network. Their novel contribution is providing the practical technology to evaluate the security of the network as a whole rather than the risk analysis of vulnerabilities of individual hosts.

*3.2.3. Visualization.* Williams et al. [27] present an efficient computation engine that generates attack graphs step-by-step and provide an interactive opportunity to trace the attacker's path. The computation module is written in C++ for speed while the visualization module is implemented in Java. It is targeted to the analyses of trust relationships the attacker exploits at each step. Xie et al. [28] proposed a two-layer attack graph to thwart inside malicious attackers from attacking the network. The upper layer is the host access graph and the lower layer is the host-pair attack graph. Xie et al. proposed this method to be rational using host-central model wherein the attacker has used stepping stones to obtain user or root level access to the network, which would decrease the computation time in computing an attack graph.

(1) They use a host access graph with properties of a simple network, which has five hosts, that is, H0, H1, H2, H3, and H4, where host H0 is the one used by the attacker, and the others compose a network he wants to penetrate.
(2) Host-pair attack and host access graphs are generated.
(3) The graphs are generated using the probability of obtaining user and root privileges through the structure of the graphs in every host-pair attack graph and then redrawn to obtain a host access graph.
(4) Xie et al. use the host access graph and the color matrices in a gray scale format as the method to generate a visual representation.
(5) The properties that are analyzed are the inside hosts to determine if an attack has transpired via user and root level privileges.
(6) The use of prioritized stepping stone recommendation is provided to the network administrator to harden the network.
(7) The methods used to evaluate this work are the adjacency matrix to evaluate the network security.

Chu et al. [29] proposed visualization tool called NAVIGATOR, which is an improved tool from the previous works NetSPA and GARNET, that is designed to depict the current state of the system for future security planning. NAVIGATOR builds upon GARNET, which utilizes the strip tree algorithm to process the grouping of hosts. NAVIGATOR is built using Java with a C++ engine for calculating attack graphs and reachability. The improvement in NAVIGATOR from its predecessor has given it the ability to represent client site and trust relationship exploits, as well as viewing the infrastructure devices and network topology with zooming features for an in-depth analysis. NAVIGATOR provides recommendation sets, as well as proposing scenarios to determine the impact the recommendation may have on the system.

*3.2.4. Response Recommendation/Implementation.* Ingols et al. [8] proposed multiple-requisite graph tool NetSPA, which imports data from multiple sources to autonomously compute reachability for a given host to connect to open ports. NetSPA is able to generate attack graph to analyze 250 actual hosts and 50,000 simulated hosts. NetSPA uses a matrix to determine the reachability of a network using typical information of hosts, IP address, open port number, and protocol. Once the reachability was determined using reverse reachability (i.e., for a given interfaces' exit nodes and walk the chain backwards from them to all other interfaces' entry nodes), the attack graph was constructed, followed by the computed suggested remediation procedures.

(1) They use the parameters from the host interface (ip address, port number, and protocol) to construct attack graphs.
(2) NetSPA constructs attack graphs on a breadth-first technique and uses a method to model reachability using tuples of the form [source IP $\rightarrow$ target IP: portnum/protocol]. The maximum number of nodes in an MP graph is linearly related to the source data.
(3) The graphs are generated by utilizing a collapsed reachability matrix as input. Using the collapsed reachability matrix and providing reachability groups drastically reduce the cost of computing the reachability matrix.
(4) NetSPA uses graphviz to visualize the graphs. Ingols et al. [8] recognize the need to improve the graph visualization aspect of this work.
(5) The properties they are able to analyze are the firewall settings of inbound and outbound traffic. They are implementing the properties by grouping host and firewalls and using firewall rulesets.
(6) NetSPA provides recommendations through the computation of individual prerequisite nodes in a graph to determine the vulnerabilities of concern to prevent the attack from satisfying the prerequisite, along with states the attack cannot reach.
(7) The method used to evaluate this approach involves testing 250 actual hosts and 50,000 simulated hosts. The results show graph generation in a suitable time frame and linearly scales relative to the input.

Huang et al. [16] present a procedure to distill from a full attack graph of the network a smaller critical attack graph. The full scale attack graphs are usually too large to be convenient for finding security flaws efficiently. So, using the critical attack graph with smaller size, the paths form the critical attack graph surface. This though only can detect an approximation of the values calculated with the full attack graph, which turns to be a good trade with considerable speed advantage. This can be a first catch leading to in-depth analysis and thus forming a quick response, as response time can be crucial in real time attack mitigation. This makes the visualization better along with the analysis.

## 4. Discussion

The exhaustive critical study of prominent works in section III gives the state of the art in modeling and use of attack graphs in security systems. In this section we present, as of today, what are the capabilities, possible applications, and challenges ahead for the use of attack graphs in security systems. The complexity of state space explosion in attack graphs has deterred its effective use in real world application. Recent works have seen an increase in the system parameters being represented in attack graph modeling leading to diverse properties being analyzed leading to detection of wide range of security violations to provide meaningful recommendations of security measures. Based on the nondeterministic Büchi automaton, attack graphs are generated to model the continuously executing systems, as the violation of security properties represented in the linear temporal logic over the network. Using Hidden Markov Models the graph modeling is made to computationally capture the probability of attacks. Both changing topological information of the network and the vulnerability information of the individual hosts are being represented in the attack graphs as well as ranking the nodes.

Technology is mature to both incrementally learn from the system and dynamically generate the attack graphs up to 1000s of nodes. The generation of and representation in attack graphs can be simulated effectively with tools such as MulVal, as conducted in [5, 20, 23]. Once the technology is determined, the scalable automatic generation of the attack graphs is being achieved up to 1000s of nodes. Moreover, as the process gets more automated the identification of attack paths and subgraphs of interest is more for machine functionality than for human vision. Effective methods are devised to either circumvent or accommodate the increasing state space complexity.

The security properties analyzed are the ones that can be represented in the linear temporal logic over the network. The violation of these properties is detected and the attack paths will be generated. Both the violations of properties on individual hosts and the whole network are being captured with the different methodologies developed. The visualizing of the graphs is done by ranking the nodes, separating the vulnerability and topological information and such methods. Visualization tools, such as NAVIGATOR, provide sufficient capabilities to represent and visualize a network topology. The recommendations are provided based mostly on static analysis. In that case it is the budget that decides the measures that serve the preserving of the higher priority security properties. During the dynamic analysis the recommendations are based on the amount of information gathered and processed until the analysis started, as in [27]. Usually it is the identification of the attack path towards an attacker's goal and the recommendation of the measure to thwart that effort with qualified results. These results are summarized in Table 1.

One recent use of attack graphs has been to develop a metric for zero-day attacks [30]. Since it is important to be able to measure the risks associated with exploiting the unknown vulnerabilities and improve the safety of the network associated with such risks, [30] considers attack graphs to quantify such vulnerabilities. Their main contribution is proposing that they will count the number of vulnerabilities in the network that must be successfully exploited to compromise an asset in the network. This provides an important link in quantifying how susceptible the network is to the known attacks but also to those which are not. They propose a $k$-Zero Day Safety model, to quantify how many successful exploits are needed to compromise an asset. Their model makes three formalized assumptions for asset compromise by exploiting the vulnerabilities:

(1) that their exploitation requires a network connection between the source and destination hosts,

(2) a remotely accessible service on the destination host,

(3) existing privilege on the source host.

The second set of preconditions they consider has

(1) existence of service,

(2) connectivity,

(3) attackers existing privilege,

and with a Post condition of *having the privilege of service*. Thus, they define *a zero-day attack graph as a directed graph composed of both zero-day and known exploits, with edges pointing from preconditions to corresponding exploits and from exploits to their postconditions*.

Since their method depends on using the attack graph to track how many compromises are needed to compromise an attack, it brings the focus on the known network parameters to deal with the unknown threats. They provide algorithms to do so and establish their complexities as being reasonable for effective deployment.

*4.1. Application.* As an example, let us consider a website that contains numerous validation vulnerabilities. This leads to a sql injection attack (SQLIA) where the attacker intends to gain unauthorized access to information within a database. The attacker intends to maximize chances of success by probing the website containing server information with known vulnerabilities to unpatched database servers. The SQL injection attack is normally a multistaged attack requiring knowledge of specific information for success. Given our scenario and the above possibilities and capabilities of attack graphs, we can address security problems due to

(i) the attacks with known goals,

(ii) the affected topology of the network due to attack which grows at a rate less than $N^3$ for an $N$ host network and should be contained within 1000s of nodes,

(iii) the attack exploited by the known vulnerabilities.

*4.2. Challenges.* There are still some important challenges in making the use of these attack graphs in the security systems effective in addressing the present day attacks:

(i) optimum cost effective security measure with *imperfect information* about the network, as, in the modern day, many nodes will keep joining and going out of network constantly,

(ii) *generation of useful domain model* from network topology,

(iii) *effective attack taxonomy* for the attack identification which can be updated so that only the differential changes in the present have to be learned and the rest of the information can be accessed efficiently,

(iv) an *integrated security system*.

These limitations restrict the security systems from effectively addressing the attacks launched through distributed nodes in a network with a dynamic topology.

Table 1: State of the art.

| Features | Prominent works |
|---|---|
| Formal models/methods used | Nondeterministic Buchi automaton, stream automaton, omega languages, linear temporal logic. |
| Parameters represented | Vulnerabilities on hosts, network topology information, cost of transitions, transition probability, quantified attacker rewards, damage/compromise network states. Bayesian learning is used to obtain information of network and hosts. |
| Automatic graph generation | Tools (NetSpa, MulVal, NAVIGATOR, BRITE, GT-ITM, and GARNET). Graphs up to 1000 s of nodes. Directed graphs with and without cycles are generated. |
| Properties analyzed | Properties captured in LTL of attack paths. Dynamic analysis provides a real time evaluation of network. |
| Violations detected | Network paths leading to exploits of individual host, break down of network paths. |
| Visualization | Tools (NAVIGATOR, GARNET, and NetSpa), methodology (separating host-vulnerability info and network-topology info in the graph, ranking of nodes of graph). |
| Recommendations derived | Least expensive and minimum number of cuts to break the attack paths. Identification of the most vulnerable and most affective hosts in the network to secure. |
| Computational complexity | Min cost SAT solving (MCSS) is used to calculate critical paths. Either with randomization like Monte Carlo methods or not, the analysis for the graph with $N$ hosts is between $O(N^2)$ and $O(N^3)$. |

## 5. Conclusion

The modeling of network systems as attack graphs gives opportunities to analyze the system for some security properties. As a methodology, attack graphs scale to model complex systems with useful results. The changing topology and the dynamic nature of the attacks are the current challenges. It has to be used in conjunction with other complimentary methodologies like dynamic response formulation based on game theory, minimizing the real time learning, to get a system powerful enough to be of practical interest. Given the complexity involved in constructing, evaluating, and modifying attack models like attack graphs and attack trees of an enterprise level complex network system, other alternative approaches have been also suggested as in [9]. More creative metrics are necessary to enhance the defender's ability to compare various attack graphs, as described in Idika [5]. We believe this work provides a foundation for delineating the continued challenges in the realm of attack graph modeling, generation, visualization, and analysis to play an effective role in the defense against the evolving real world attacks.

## Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

## References

[1] O. M. Sheyner, *Scenario graphs and attack graphs [Ph.D. thesis]*, School of Computer Science, Computer Science Department, 2004.

[2] J. M. Wing, *Scenario Graphs Applied to Network Security*, Elsevier, 2008.

[3] R. Lipmann and K. Ingols, "An annotated review of past papers on attack graphs," Tech. Rep., Lincoln Laboratory, 2005.

[4] V. Viduto, W. Huang, and C. Maple, "Toward optimal multi-objective models of network security: survey," in *Proceedings of the 17th International Conference on Automation and Computing (ICAC '11)*, pp. 6–11, September 2011.

[5] N. C. Idika, *Characterizing and aggregating attack graph-based security metric [Ph.D. thesis]*, Center for Education and Research, Information Assurance and Security, Purdue University, 2010.

[6] V. Mehta, C. Bartzis, H. Zhu, E. Clarke, and J. Wing, "Ranking attack graphs," in *Proceedings of the 9th International Conference on Recent Advances in Intrusion Detection (RAID '06)*, pp. 127–144, Springer, Berlin, Germany, 2006.

[7] S. Wang, Z. Zhang, and Y. Kadobayashi, "Exploring attack graph for cost-benefit security hardening: a probabilistic approach," *Computers and Security*, vol. 32, pp. 158–169, 2013.

[8] K. Ingols, M. Chu, R. Lippmann, S. Webster, and S. Boyer, "Modeling modern network attacks and countermeasures using attack graphs," in *Proceedings of the 25th Annual Computer Conference Security Applications (ACSAC '09)*, pp. 117–126, December 2009.

[9] J. Hong and D.-S. Kim, *HARMs: Hierarchical Attack Representation Models for Network Security Analysis*, SRI Security Research Institute, Edith Cowan University, Perth, Australia, 2012.

[10] R. Cole, *Multi-step attack detect ion via Bayesian modeling under model parameter uncertainty [Ph.D. thesis]*, College of Information Sciences and Technology, 2013.

[11] I. Kotenko and M. Stephashkin, "Attack graph based evaluation of network security," in *Communications and Multimedia Security*, H. Leitold and E. Markatos, Eds., pp. 216–227, Springer, Berlin, Germany, 2006.

[12] P. Kijsanayothin and R. Hewett, "Analytical approach to attack graph analysis for network security," in *Proceedings of the 5th International Conference on Availability, Reliability, and Security (ARES '10)*, pp. 25–32, February 2010.

[13] P. Xie, J. H. Li, X. Ou, P. Liu, and R. Levy, "Using Bayesian networks for cyber security analysis," in *Proceedings of the 2010 IEEE/IFIP International Conference on Dependable Systems and Networks (DSN '10)*, pp. 211–220, Chicago, Ill, USA, June 2010.

[14] R. Dewri, N. Poolsappasit, I. Ray, and D. Whitley, "Optimal security hardening using multi-objective optimization on attack tree models of networks," in *Proceedings of the 14th ACM Conference on Computer and Communications Security (CCS '07)*, pp. 204–213, ACM, November 2007.

[15] L. Wang, T. Islam, T. Long, A. Singhal, and S. Jajodia, "An attack graph-based probabilistic security metric," in *Proceeedings of the 22nd Annual IFIP WG 11.3 Working Conference on Data and Applications Security*, pp. 283–296, Springer, Berlin, Germany, 2008.

[16] H. Huang, S. Zhang, X. Ou, A. Prakash, and K. Sakallah, "Distilling critical attack graph surface iteratively through minimum-cost sat Solving," in *Proceedings of the 27th Annual Computer Security Applications Conference (ACSAC '11)*, pp. 31–40, New York, NY, USA, December 2011.

[17] J. Homer, S. Zhang, X. Ou et al., "Aggregating vulnerability metrics in enterprise networks using attack graphs," *Journal of Computer Security*, vol. 21, no. 4, pp. 561–597, 2013.

[18] S. Noel, S. Jajodia, L. Wang, and A. Singhal, "Measuring security risk of networks using attack graphs," *International Journal of Next-Generation Computing*, vol. 1, no. 1, 2010.

[19] N. Poolsappasit, R. Dewri, and I. Ray, "Dynamic security risk management using Bayesian attack graphs," *IEEE Transactions on Dependable and Secure Computing*, vol. 9, no. 1, pp. 61–74, 2012.

[20] X. Ou, W. F. Boyer, and M. A. McQueen, "A scalable approach to attack graph generation," in *Proceedings of the 13th ACM Conference on Computer and Communications Security (CCS '06)*, pp. 336–345, ACM, New York, NY, USA, November 2006.

[21] K. Ingols, R. Lippmann, and K. Piwowarski, "Practical attack graph generation for network defense," in *Proceedings of the 22nd Annual Computer Security Applications Conference (ACSAC '06)*, pp. 121–130, Miami Beach, Fla, USA, December 2006.

[22] M. Danforth, *Models for threat assessment in networks [Ph.D. thesis]*, School of Computer Science Computer, Science Department, 2006.

[23] R. E. Sawilla and X. Ou, "Identifying critical attack assets in dependency attack graphs," in *Computer Security—ESORICS 2008*, S. Jajodia and J. Lopez, Eds., vol. 5283 of *Lecture Notes in Computer Science*, pp. 18–34, Springer, Berlin, Germany, 2008.

[24] R. Schuppenies, *Automatic extraction of vulnerability information for attack graphs [M.S. thesis]*, Hasso-Plattner-Institute for IT Systems Engineering, 2009.

[25] P. Cheng, L. Wang, and T. Long, "Compressing attack graphs through reference encoding," in *Proceedings of the 10th IEEE International Conference on Computer and Information Technology (CIT '10)*, pp. 1026–1031, July 2010.

[26] E. S. Abramov, A. V. Andreev, D. V. Mordvin, and O. B. Makarevich, "Corporate networks security evaluation based on attack graphs," in *Proceedings of the 4th International Conference on Security of Information and Networks (SIN '11)*, vol. 11, pp. 29–36, ACM, New York, NY, USA, 2011.

[27] L. Williams, R. Lippmann, and K. Ingols, "An interactive attack graph cascade and reachability display," in *VizSEC 2007: Proceedings of the Workshop on Visualization for Computer Security*, Mathematics and Visualization, pp. 221–236, Springer, Berlin, Germany, 2008.

[28] A. Xie, Z. Cai, C. Tang, J. Hu, and Z. Chen, "Evaluating network security with two-layer attack graphs," in *Proceedings of the 25th Annual Computer Conference Security Applications (ACSAC '09)*, pp. 127–136, December 2009.

[29] M. Chu, K. Ingols, R. Lippmann, S. Webster, and S. Boyer, "Visualizing attack graphs, reachability, and trust relationships with navigator," in *Proceedings of the 7th International Symposium on Visualization for Cyber Security (VizSec '10)*, pp. 22–33, ACM, New York, NY, USA, September 2010.

[30] L. Wang, S. Jajodia, A. Singhal, P. Cheng, and S. Noel, "K-zero day safety: a network security metric for measuring the risk of unknown vulnerabilities," *IEEE Transactions on Dependable and Secure Computing*, vol. 11, no. 1, pp. 30–44, 2014.