Computer Science Review 26 (2017) 1–16

Contents lists available at ScienceDirect

Computer Science Review

journal homepage: www.elsevier.com/locate/cosrev

Survey

A survey on the usability and practical applications of Graphical Security Models

Jin B. Hong [a,*], Dong Seong Kim [a], Chun-Jen Chung [b], Dijiang Huang [b]

[a] Department of Computer Science and Software Engineering, University of Canterbury, New Zealand
[b] Department of Computer Science, Arizona State University, USA

HIGHLIGHTS

- Describe the development of security models and their current functions.
- Classify the applications of existing security models.
- Compare the scalability of security models with respect to the complexity analysis.
- List the use of security metrics, available tools and network domains applicable.
- Discuss the future directions of security models.

ARTICLE INFO

Article history:
Received 1 July 2016
Received in revised form 1 September 2017
Accepted 1 September 2017
Available online 24 October 2017

Keywords:
Attack graphs
Attack trees
Security analysis
Security metrics
Security models

ABSTRACT

This paper presents and discusses the current state of Graphical Security Models (GrSM), in terms of four GrSM phases: (i) generation, (ii) representation, (iii) evaluation, and (iv) modification. Although many studies focused on improving the usability, efficiency, and functionality of GrSMs (e.g., by using various model types and evaluation techniques), the networked system is evolving with many hosts and frequently changing topologies (e.g., Cloud, SDN, IoT etc.). To investigate the usability of GrSMs, this survey summarizes the characteristics of past research studies in terms of their development and computational complexity analysis, and specify their applications in terms of security metrics, availability of tools and their applicable domains. We also discuss the practical issues of modeling security, differences of GrSMs and their usability for future networks that are large and dynamic.

Contents

* Corresponding author.
E-mail addresses: jho102@uclive.ac.nz (J.B. Hong), dongseong.kim@canterbury.ac.nz (D.S. Kim), chung-jen.chung@asu.edu (C.-J. Chung), dijiang@asu.edu (D. Huang).

https://doi.org/10.1016/j.cosrev.2017.09.001
1574-0137/

1. Introduction

It is of paramount importance to understand how secure a networked system is to prevent damages caused by cyber attacks. A widely adopted method is to use a Graphical Security Model (GrSM), (e.g., an Attack Graph (AG) [1] or an Attack Tree (AT) [2]) to assess the security of a given networked system and to recommend security hardenings. AGs are one of the basic structures for most graph-based GrSMs, and ATs are one for most tree-based GrSMs. To address the limitations of AGs and ATs, new GrSMs are proposed (e.g., Attack Defense Tree [3]) to enhance the security assessment capabilities, and Hierarchical Attack Representation Models [4] using hierarchy features to improve the scalability. As a result, there is a wide range of different GrSMs available today [5], but they do not necessarily provide the same security analysis nor functions. Consequently, a diverse family of GrSMs exists today that creates a difficulty for users to determine the most suitable one for the security analysis of their networked systems.

To cope with various features of GrSMs, surveys are conducted to reflect their efficiencies in many aspects (e.g., generation, evaluation, and their complexities). However, the focuses of those surveys have changed over the years due to various reasons, such as a development of networks (e.g., static networks to dynamic networks such as cloud networks), and also the availability of tools (e.g., NetSPA [6], MulVAL [7] and NAVIGATOR [8]). Lippmann and Ingols [9] focused on the scalability of GrSMs when generating them, where prior to this survey there were no GrSMs that have analyzed more than 20 hosts in a networked system. The survey conducted by Khaitan and Raheja [10] on the same topic (i.e., scalability of GrSMs) reported the poor scalability as well. On the other hand, Alhomidi and Reed [11] conducted a survey based on the representation of graph-based GrSMs, where structure of an AG can be represented differently based on different features implemented (e.g., clustering (MPAG) and non-clustering (AG)), but there is no standard model to cover all these features. Kordy et al. [5] conducted a survey based on directed acyclic graphs

Table 1
Contribution of available surveys on security models.

	[9]	[10]	[11]	[5]	Our work
Development of GrSMs	**No**	**No**	**No**	**No**	Yes
Details of Arms	Yes	Yes	Yes	Yes	Yes
Complexity Analysis	Yes	**No**	**No**	Yes	Yes
Usable Metrics	**No**	**No**	**No**	Yes	Yes
Tools Availability	Yes	**No**	Yes	Yes	Yes
Applicable Domains	**No**	**No**	**No**	Yes	Yes

(DAGs), which compared mainly the structures of GrSMs and the availability of tools. However, these surveys did not consider the efficiencies of generating, representing, evaluating and modifying the GrSMs, which are essential attributes of using the GrSMs for modern networked systems that can easily scale to hundreds and thousands of hosts with a highly dynamic nature, such as the Cloud computing [12].

The purpose of this survey is to identify the usefulness of GrSMs in a context of modern networked systems that are typically very large and highly dynamic. We describe the usefulness of GrSMs on the basis of: (i) efficiency, (ii) application of metrics, and (iii) availability of tools. The efficiency is described by the scalability and modifiability of GrSMs, which can be detailed in their phases. The application of metrics distinguishes which types of security metrics can be used, and we categorize them into security-oriented (e.g., risk analysis), mathematical (e.g., a probability of an attack success), or financial impact (e.g., return on investment). The availability of tools describes how the user may access the GrSM in a form of tools.

The contributions of this survey are to overview the evolution of GrSMs and identify the future direction of the research. More specifically, they are:

- Classifying the applications of existing GrSMs;
- To compare the performance of GrSMs in terms of their complexities in the GrSM lifecycle;
- To investigate the use of security metrics, available tools and various system domains applicable;
- To identify and discuss the current and future directions of the GrSM research.

The major differences of this survey paper compared to the previously published ones are summarized in Table 1.

The rest of the paper is organized as follows. Section 2 describes the existing GrSMs and their classifications. Section 3 presents the complexity analysis of GrSMs. Applications of GrSMs in various attack/defense scenarios are described in Section 4. Discussion of the current and future research directions of GrSMs are shown in Section 5, and we conclude the survey in Section 6.

2. Classification of GrSMs

GrSMs can mainly be categorized into two major types: Graph-based (such as AGs [1,13,14], MPAGs [15], Topological Vulnerability Analysis [16], Logical Attack Graphs [7]), and Tree-based (such as ATs [17], Defense Trees [18], Attack Response Trees [19], and Attack Countermeasure Trees [20]). Many GrSMs have been developed for various applications with different GrSMs having different features for security analysis, such as incorporating countermeasures, enumerating all possible attack scenarios, and hierarchical representation to improve the scalability. We explore these in detail in Section 4. The scope of GrSMs application is not limited to conventional network systems only, but also any network systems that exchange data (e.g., supervisory control and data acquisition (SCADA) systems [21], voting systems [22], Smart Power Grids [23], banking systems [24], airline systems [25]).

To describe the efficiency of the GrSM in detail, we consider the phases of the GrSM: (i) preprocessing, (ii) generation, (iii) representation, (iv) evaluation, and (v) modification [26]. The efficiency of the GrSM phases is of paramount importance, because a deficiency in any phase may cause a disruption of using the GrSM in practice. The GrSM phases, the flow of security information from the networked system in the lifecycle of security models, are shown in Fig. 1, which was first proposed by Hong [27]. In the preprocessing phase, security information is gathered. The generation phase uses the gathered security information and generates the GrSM. The representation phase visualizes and stores the GrSM. The evaluation phase assesses the security of the networked system with given input security metrics. The modification phase captures the change in the networked system and updates the GrSM accordingly. In this survey, we consider the efficiencies of all GrSM phases, where the previous survey papers have only focused on the generation and representation.

In this survey, we focus on non-state space and hierarchical models that are most widely used for security analysis. State space models (e.g., stochastic Petri Net [28] and stochastic Reward Net [29]) are well known and can be adopted to analyze security, but they suffer from the scalability problem due to an exponential growth of the number of states.

Fig. 2 shows the evolution of GrSMs since the 1990s. An original version dating up to 2014 was shown in [27], which has been extended to include the most recent GrSMs up to 2017. The different types of lines are used to distinguish from one another for readability (i.e., they all have the same meaning). It shows that many variations of the AGs and ATs are introduced, and many of the recent security models combine various features of ones previously introduced. Fig. 3 shows the categorization of GrSMs, showing the number of GrSM categories developed as of today.

2.1. Tree-based GrSMs

Tree-based GrSMs, such as an *Attack Tree* (AT) [2,30], are widely used to model the network security. There are number of different tree-based GrSMs providing different functionalities and features, such as:

- Functions to model countermeasures:
  1. Defense Tree (DT) [18]
  2. Protection Tree (PT) [31]
  3. Attack Countermeasure Tree (ACT) [20,32]
  4. Attack Defense Tree (ADT) [33]
- Features to perform probabilistic analysis
  1. OWA Tree (OWAT) [34]
- Features to consider all possible attack scenarios
  1. Attack Response Tree (ART) [19]
- Taking into account metrics other than security
  1. Attack Fault Tree (AFT) [35]

There are eight major categories of tree-based GrSMs with significant structural differences and methods of analysis: (i) AT, (ii) DT, (iii) OWAT, (iv) PT, (v) ART, (vi) ACT, (vii) ADT, and (viii) AFT. Other tree-based GrSMs are variants of these models with additional features in their applications, but there are no significant structural changes [17,24,36–42]. A keynote for tree-based GrSMs is that, depending on the construction method, the visualization (i.e., the structure) can vary but the network properties are kept the same as long as there is no information loss (e.g., using heuristic methods) [43].

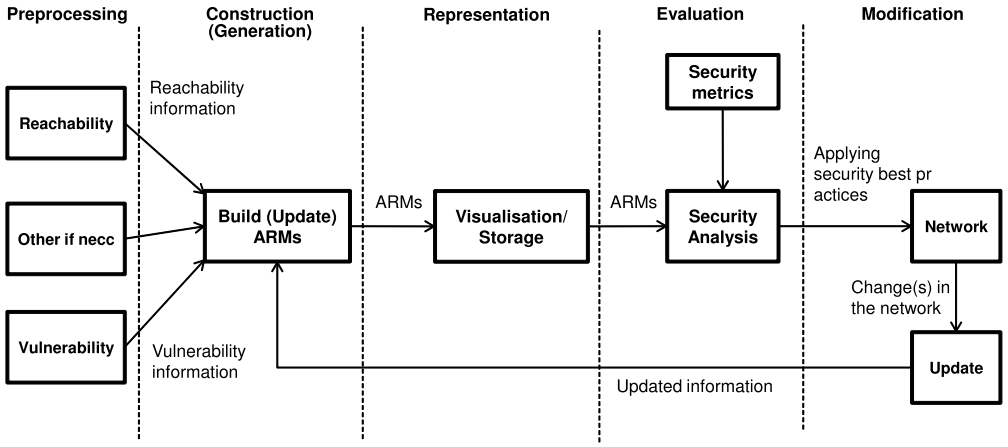

Fig. 1. GrSM phases.

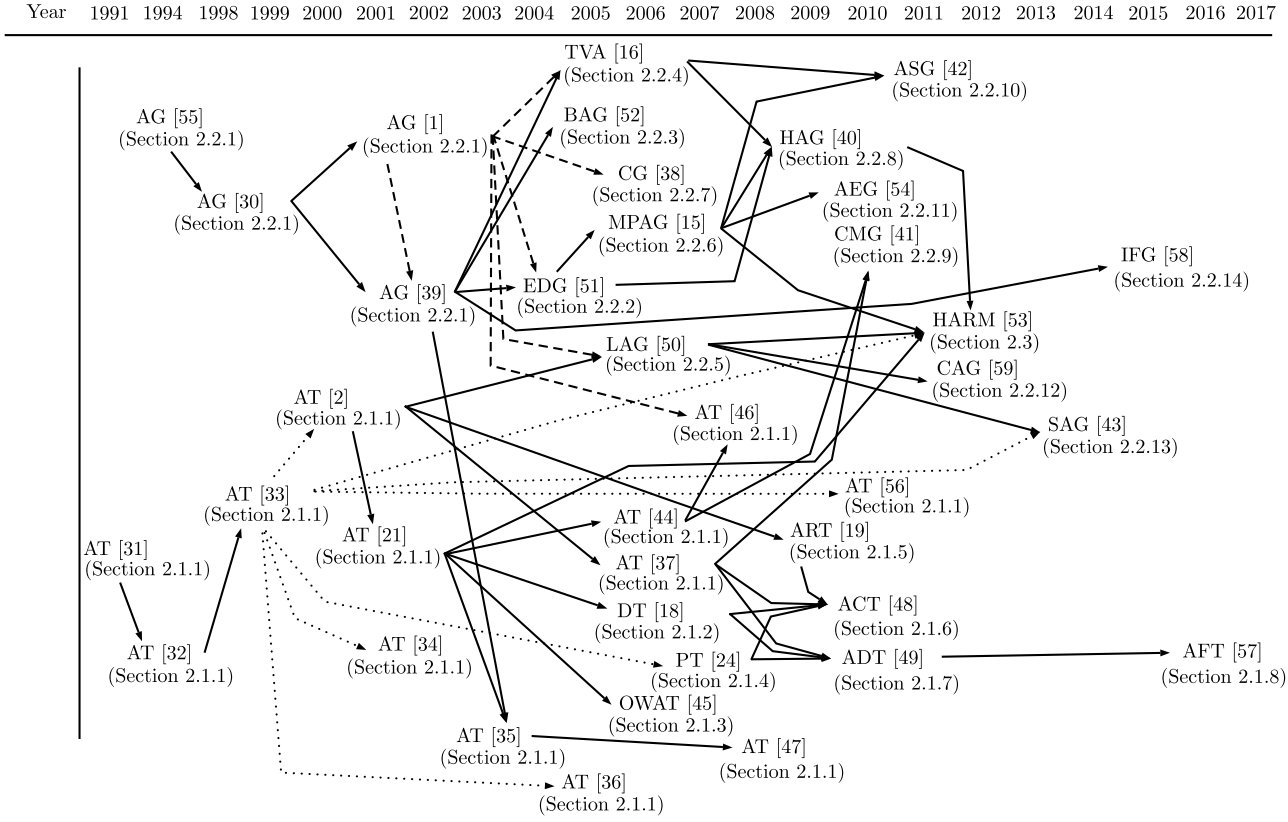

Fig. 2. Timeline of GrSM evolution.

2.1.1. Attack tree (AT)

Weiss [44] proposed to use Threat Logic Tree to evaluate the system security, which was the first graphical attack modeling technique. Amoroso [45] proposed to use Threat Tree for system reliability engineering to security. Nowadays, these models are categorized under AT. The AT consists of a goal node (often called the root node), where the attack is successful if the condition of the node can be satisfied. If the goal cannot be satisfied directly (e.g., complexity or abstract), it is then decomposed into sub-goals. Sub-goals are joined by various logical gates (e.g., *AND, OR, k-of-AND, sequential-AND*) to specify its prerequisites. The AT does not explicitly model countermeasure events (e.g., detection and mitigation). The formalism of the AT is given by Schneier [2,30], and Mauw et al. [43] provided a foundation of the AT, formalized the AT structure, attributes, transformations and projection. Audinot and Pinchinat [46] validated the soundness of ATs, and this problem is further addressed in [47] where the correctness of an AT with respect to the analyzed system has been studied. Further, the use of Sequential-AND gates was formalized by Jhawar et al. [48]. In addition, Horne et al. [49] developed a new semantics for an AT with sequential operator, based on linear logic. A Vulnerability Tree (VT) [50] is very similar to an AT, where the root node is expressed

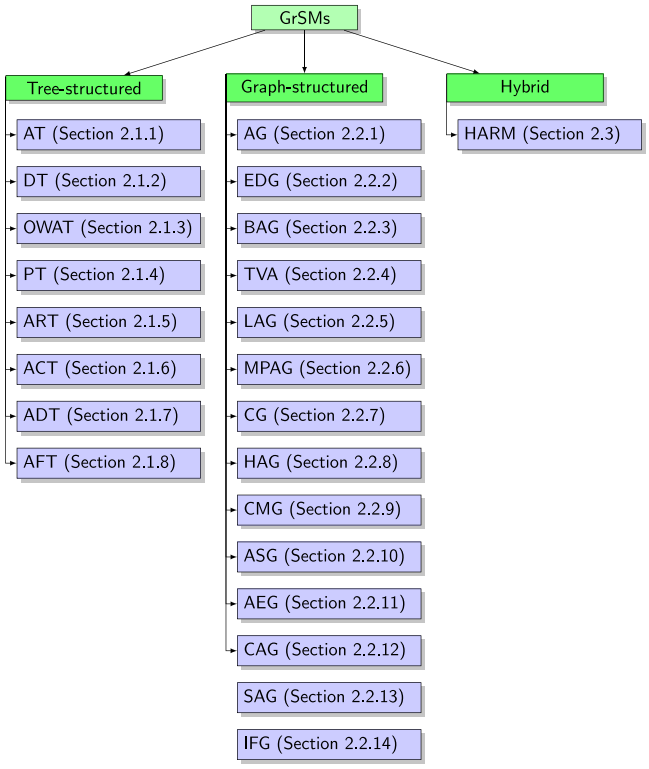

Fig. 3. Categories of GrSMs.

as a vulnerability instead of an attacker's goal. Modeling attacks explicitly assume there are vulnerabilities in the system, hence vulnerabilities can be replaced by attacks with the same structure. Typically, ATs lacked means to generate the model automatically. However, Vigo et al. [51] and Ivanova et al. [52] have presented automatic generation of ATs in recent advances.

2.1.2. Defense tree (DT)

The DT models countermeasure events in addition to the leaf nodes (i.e., attack events) of the AT [18]. Hence, the formalism extends the AT, where countermeasure events are also modeled as leaf nodes and attached to the AT (i.e., initial attack conditions and their countermeasures). Also, multiple countermeasure event nodes can be typically attached to a leaf attack event node. Some attack event nodes in the DT do not have a countermeasure event, because the expression of the DT in the example is a decomposition of each event and they are aggregated in an *AND* gate, whereas in [18], authors have decomposed events and connected directly to their parent node.

2.1.3. OWA tree (OWAT)

OWAT is an extension to the AT where it incorporates the use of probabilistic security analysis based on Ordered Weighted Averaging (OWA) operators [34]. The conventional *AND* and *OR* gates cannot provide functionalities of partial satisfactory conditions in the AT (i.e., all or any), where OWA operators take these into account and use assigned weights to satisfy partial conditions.

2.1.4. Protection tree (PT)

PT nodes represent protection (countermeasure) events only, in contrast to AT where attack (or vulnerability) events are modeled [31]. Every event in the AT is replaced with a protection event. Initially, all leave nodes (i.e., initial attack nodes) are assigned with a countermeasure event. If intermediate attack nodes are not covered by a protection (countermeasure), then countermeasures are added to the PT. This process is repeated until the root node is covered by a countermeasure.

2.1.5. Attack response tree (ART)

The ART is an extension of ATs that incorporate possible countermeasures as well as considering intrusion detection uncertainties [19]. The ART is constructed based on all possible attack scenarios, which if naively computed can suffer from a scalability problem, but authors mention the ART is designed offline by experts on each computing asset. The ART model is built based on the *attack consequences*, which scopes the all possible attack scenarios. A node in the ART represents an attack event, which can be decomposed into a number of sub-attack events. The ART can have a response event (countermeasure) at any node. However, when evaluating the ART, the state of each event is considered (i.e., the combinations possible for every gate). As a result, the number of states grows exponentially with respect to the number of leaf events in the ART.

2.1.6. Attack countermeasure tree (ACT)

The ACT features the characteristics of the DT, where countermeasures are placed not only at leaf nodes, but also at the intermediate nodes [32]. The ACT models both attack and countermeasure events. In contrast to the DT, the ACT categorizes the countermeasure into detection and mitigation separately. Thus, providing the functionality to evaluate the effectiveness of detection and mitigation in detail. The ACT is similar to the ART, but countermeasure events are expressed as a non-state space model (i.e., states are not considered in the ACT).

2.1.7. Attack defense tree (ADT)

The ADT extends the AT with an addition of defense nodes that appears at any level of the tree, which enables the user to capture the interactions between an attacker and a defender [33]. In comparison to the ACT, nodes in ADT are either an attack or a defense and forms a countermeasure relationship in between. A node represents goals and they can be further decomposed into sub-goals. Further work that incorporates game theory has been done by Aslanyan et al. [53], and Fraile et al. [54] conducted a case study to investigate the usefulness of ADT.

2.1.8. Attack fault tree (AFT)

The AFT is proposed by Kumar et al. [35], which incorporates the capabilities of ATs and Fault Trees (FTs) to capture both security and safety aspects of the system. Although the underlying formalism is very similar to the AT, the widened capabilities allow the user to investigate both security and safety aspects using a single model, which other GrSMs are mostly incapable to do so.

2.2. Graph-based GrSMs

Graph-based GrSMs utilize the graph structure to represent attack scenarios. The most common term to describe this type of structure is AGs, which has been first introduced by Phillips and Swiler [55] in 1998 and has extensively been used ever since. An AG is a *directed acyclic graph* (DAG) that represents the interdependencies among vulnerabilities and actions that lead to the violation of the security policy of a network. AG is able to show the possible attack paths from attackers to target hosts. Attack paths are composed of a set of aspects of the network, such as vulnerabilities, network connectivity and configuration, current conditions and exploits of network states, as well as the relationship among all of them. Several variant structures of AG exist in the literature for different purposes; Exploit Dependency Graph

(EDG) [56], Bayesian Attack Graph (BAG) [57], Topological Vulnerability Analysis (TVA) [16], Logical Attack Graph (LAG) [58], Multiple Prerequisite Attack Graph (MPAG) [15], Compromise Graph (CG) [59], Hierarchical Attack Graph (HAG) [60], Countermeasure Graph (CMG) [61], Attack Scenario Graph (ASG) [62], Attack Execution Graph(AEG) [63], Conservative Attack Graph (CAG) [64], Security Argument Graph (SAG) [65], and Incremental Flow Graph (IFG) [66]. We describe these graph-based GrSMs in the following subsections.

2.2.1. Attack graph (AG)

Dacier and Deswarte [67] used a Privilege Graph to capture the unsafe protection states, where the idea developed into an AG by Phillips and Swiler [55]. There can be many different representations of an AG [11], but normally an AG consists of nodes and edges. A node may represent states (e.g., host, privilege, exploit or vulnerability), and an edge is a directed transition from pre-condition to post-condition when an event of the state (e.g., an exploit) has been executed. Sheyner et al. [1] presented an automated generation of the AGs, and Ammann et al. [68] presented a more scalable solution of generating the AGs compared to Sheyner by introducing the explicit assumption of monotonicity. The monotonicity assumes the attacker never backtrack its attack steps. Given the assumption, they can reduce the complexity of the AG generation from exponential to polynomial computational complexity. As reported by Lippmann and Ingols [9], Ammann achieved $O(N^6)$ computational complexity.

Another way to produce an AG is to use a model checker [69]. In this type of attack graph, each node represents an entire state of a network and an edge represents the state transitions between the states. Ritchey and Ammann are the first to use the SMV model checker to analyze network vulnerabilities [70]. Sheyner et al. [1,14] used a modified model checker NuSMV to generate the AG. However, the scalability problem still exists for a very large sized enterprise systems with these AG generation techniques [9]. There are ways to improve the scalability (e.g., SysML-Sec AG [71] to reduce the complexity for complex AGs), while others have developed AG variants to restructure the underlying mechanisms (as shown in the following subsections).

2.2.2. Exploit dependency graph (EDG)

Noel et al. [56,69,72] proposed EDG to model vulnerabilities and possible exploits in a networked system. There are three types of nodes in the EDG: (i) Privilege node, (ii) pre-condition node, and (iii) post-condition node. A privilege node represents the level of privilege access in the networked system (e.g., user or root privilege), a pre-condition node represents the vulnerabilities, and a post-condition node represents the exploits. Edges represent the state transition in either pre-condition to post-condition, or from a post-condition to a privilege.

2.2.3. Bayesian attack graph (BAG)

BAG is first proposed by Liu and Man [57], which models potential attack paths with the Bayesian network. Conditional probability tables were assigned to edges to utilize Bayesian inference methods. Frigault and Wang [73,74] proposed improvements over [57] with model structures and functionalities. The structure of the BAG does not differ from the structure of the AG, but the AG is treated as a Bayesian network with probabilistic assignments. Hence, the complexity and functionalities depend on the AG.

2.2.4. Topological vulnerability analysis (TVA)

TVA is a framework of network security analysis that improves the functionalities and scalability of the AG [16]. The idea initially proposed in [75], but the description and structure of the framework were fully described in [16]. They used dependency graph, which is similar to EDG that uses pre-conditions and post-conditions. TVA tool builds the dependency 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. Next, the model builds the forward dependency graph backwards 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. Noel et al. [76] 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 improves scalability.

2.2.5. Logical attack graph (LAG)

In LAG, a node represents a logical statement, where the edge represents the causality relations between network configurations and the attacker's potential privileges [58]. That is, the graph represents the cause and effect relationship of attacker's potential actions in the system. The LAG is goal-oriented (i.e., a specified goal to generate the graph), and is represented using two types of nodes: derivative nodes and fact nodes. The fact nodes may have two types: a primitive fact node and a 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 LAG can be generated using MulVAL [7]. Another logical model is discussed in [77]. 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.

2.2.6. Multiple prerequisite attack graph (MPAG)

Ingols et al. [15] proposed MPAG to model the security situation in a networked system. There are three types of nodes in the MPAG: (i) state, (ii) prerequisite, and (iii) vulnerability, and the edge represents the transition between these nodes. The state node represents the attacker's level of access on a given host. The prerequisite node represents a reachability group or a credential. The vulnerability node represents a specific vulnerability on a particular port. The MPAG can be generated using a tool NetSPA [15], and its successor GARNET [78] and NAVIGATOR [8] also adopted the use of MPAG. Lee et al. [79] introduced a similar model that also used three types of node: (i) security condition node, (ii) exploit node, and (iii) penetrated condition node, and the edge represented the transitions between these nodes.

2.2.7. Compromise graph (CG)

McQueen et al. [59] introduced CG based on directed graphs, where nodes represent the attack state, and each edge represents a successful compromise. The value of each edge is an estimate of the time required to make the transition, where the weight is related to the difficulty of exploiting the vulnerabilities associated with that edge. A time-to-compromise (TTC) metric is modeled for each edge with different probability distribution functions. Leversage and Byres adopted a similar approach called state-time estimation algorithm (STEA) [80]. They combine a slightly modified TTC calculation approach with a decomposition of the attack according to the architectural areas of the targeted system. However, the CG only consists of attack states, the model lacks features to capture pre and post-conditions (i.e., vulnerabilities).

2.2.8. Hierarchical attack graph (HAG)

Xie et al. [60] proposed two-layered AG, modeling network hosts and vulnerabilities onto two different layers of hierarchy. In the upper layer, network hosts are mapped based on their reachability, and edges in the upper layer specified lower layer models of vulnerabilities. Hong and Kim [81,82] proposed hierarchical GrSM (HARM) with two layers modeling network hosts and vulnerabilities, respectively. Then, an AG is used in both the upper and the lower layers to generate the HAG. The difference between the HAG and two-layered AG is the lower layer models, where two-layered AG has lower layer models referenced in the edges and HAG has lower layer models referenced in the nodes. Hong and Kim extend the use of HARM to incorporate other types of GrSM into different layers, which is further detailed in Section 2.3.

2.2.9. Countermeasure graph (CMG)

Baca and Petersen [61] proposed CMG, which is an extension from the AT that has a DAG-based structure for identification and prioritization of countermeasures. Specification of actors, goals, attacks and countermeasures are required to generate the CMG. When the representing events are related, edges are drawn between goals and actors, actors and attack as well as between attacks and countermeasures. Priorities are assigned to goals, actors, attacks and countermeasures according to the rules of hierarchical cumulative voting [83]. However, there is no study on how scalable the CMG is. Multiple goal nodes are generated and each goal node maintains a list actor nodes that consists of multiple attacks and countermeasures, which in a large sized networked system it can quickly get very complex.

2.2.10. Attack scenario graph (ASG)

ASG combines AG and EDG to enhance the situation awareness [62]. Two major solutions were proposed: (i) mechanism to evaluate the likelihood of each attack pattern, and (ii) address the scalability issue in the evaluation phase. The dependencies between network components, services and infrastructure are captured, and probabilistic features are added to the AG. However, there are no significant structural changes compared to the EDG, because ASG is a successor of EDG and AG that inherited the structural properties with extended features.

2.2.11. Attack execution graph (AEG)

AEG represents potential attack steps against the system and specifies an adversary profile [63]. AEG has five types of nodes: (i) Attack Step, (ii) Access, (iii) Knowledge, (iv) Skill, and (v) Goal. *Attack Step* nodes represent intermediate step of an attack, *Access* nodes represent access state of the attacker, *Knowledge* nodes represent compromised data or information, *Skill* nodes represent exploits, and *Goal* nodes are targets of the attacker. AEG has similar properties as MPAG, with an additional intermediate step of an attack and specification of compromised data or information. However, the generation method requires manual input of attacks and adversaries information from the user.

2.2.12. Conservative attack graph (CAG)

CAG is proposed by Zhuang et al. [64], which is an extension to the LAG 2.2.5 that enables the modeling of both known and unknown vulnerabilities, as well as to capture how an attacker might move through the system to gain specific privileges. It applies stochastic analysis to evaluate the security of the network, as it models both gaining and losing knowledge and privileges that invalidate the monotonicity assumption of most previous work on graph-based GrSMs. For its application, CAG is used to evaluate and deploy moving target defense system [84].

2.2.13. Security argument graph (SAG)

SAG is proposed by Tippenhauer et al. [65], and it is generated in three steps: (i) capture information about network components and their interactions that may affect the security goal as a Goal (G) graph, (ii) detailed system information (e.g., network topology, or device configuration) that are collected and integrated with the *G*-graph to generate a Goal, System (GS) graph, and (iii) possible attacker actions and capabilities are integrated with the *GS* graph to generate a Goal, System, Attacker (GSA) graph. The transition between these steps utilize *Extension Templates*, which is an automated generation procedure with a given input of a definition of security goal (e.g., in relation to Confidentiality, Integrity, or Availability), system information (e.g., network topology) and attacker actions and capabilities. CyberSAGE [65] utilizes SAG as its underlying security model.

2.2.14. Incremental flow graph (IFG)

IFG is proposed by Dhawan et al. [66], which specifically work on Software-defined Network (SDN), as it needs to collect forwarded OpenFlow control messages. IFG is constructed with the collected OpenFlow control messages that define the topology of the SDN. As it runs in real time, the IFG gets updated when there are new OpenFlow control messages. At the same time, all malicious messages are observed. The tool proposed using the IFG named Sphinx works by; (a) monitoring all controller communication and identify relevant messages to construct IFG, (b) analysis of OpenFlow messages and generation of IFG, and (c) verification of current metadata for malicious activities.

2.3. Hybrid GrSMs (HARM)

Hybrid GrSMs use both graph and tree-based GrSMs. Hong and Kim [81,82] used a Hierarchical GrSM (HARM) with both AG and AT in two different layers that modeled network topology and vulnerabilities respectively. Functionalities of the hybrid GrSMs are dependent on the model used. For example, if an AG is used in both layers of the HARM, then it can provide attack sequence information, whereas the HARM with AT in both layers cannot. The formalism of the HARM can be found in [4].

3. Complexity analysis of GrSMs

One of the goals of this paper is to compare the performance of existing GrSMs. In this section, we compare the performance of the GrSMs presented in Section 2 based on their analysis methods in terms of their efficiency in their lifecycle.

As the number of network hosts grows in a dynamic networked system (e.g., Cloud networks), it is crucial for the GrSMs to be scalable and modifiable to cope with changes and update the security information. Hence, the term efficiency of GrSMs in this paper is based on their scalability and modifiability, which can be detailed in their phases of GrSM lifecycle. We compare the complexities in each phase of GrSMs, which is shown in Table 2. Here, the term *complexity* refers to the theoretical performance with respect to time and computational resources required. Although the actual performance differs from the theoretical performance, it gives the user a solid guideline prior to selecting the GrSM. If there are multiple references to the same GrSM, then the most scalable method is described.

The complexity of the generation phase describes the algorithm used, and the complexity of the representation phase is based on the size of the GrSM (i.e., size complexity). We define the complexity of the evaluation phase based on the proposed security analysis method (e.g., computing all possible attack scenarios, or clustering and pruning algorithms). The complexity of the modification phase is based on the computational algorithm to update the

Table 2
Complexity of GrSM phases.

GrSM	Generation	Representation	Evaluation	Modification
AT 2.1.1	D	D	D	U
DT 2.1.2	D	D	D	U
OWAT 2.1.3	D	D	D	U
PT 2.1.4	D	D	D	U
ART 2.1.5	E	E	H	U
ADT 2.1.7	D	D	D	U
ACT 2.1.6	D	D	D	U
AFT 2.1.8	D	D	D	U
AG 2.2.1	S	S	H	U
EDG 2.2.2	S	S	H	U
BAG 2.2.3	S	S	H	U
TVA 2.2.4	S	S	H	U
LAG 2.2.5	S	S	H	U
MPAG 2.2.6	S	S	H	U
CG 2.2.7	U	U	U	U
HAG 2.2.8	S	S	S	E
CMG 2.2.9	D	D	D	U
ASG 2.2.10	S	S	H	U
AEG 2.2.11	U	U	H	U
CAG 2.2.12	S	S	H	U
SAG 2.2.13	U	U	U	U
IFG 2.2.14	U	U	U	U

Key: D: Dependent E: Exponential H: Heuristic S: Scalable U: Unknown

GrSM. Table 2 shows the complexities of different GrSMs, and their values are described as the following: (i) *Dependent* (D) is when the complexity of the evaluation phase is relative to the size of the GrSM (e.g., exponential number of events in the GrSM will result in an exponential complexity in the evaluation phase). (ii) *Exponential* (E) is when the computation of the GrSM phase has an exponential complexity. (iii) *Heuristic* (H) is when a heuristic method is used to evaluate the GrSM in more scalable manner. (iv) *Scalable* (S) is when the algorithm can evaluate the GrSM in a polynomial or linear time. (v) *Unknown* (U) is when the GrSM phase has not been analyzed for the performance. These values are populated based on their complexity analysis given (based on their work or their predecessors that can be found in Fig. 2).

3.1. Tree-based GrSMs

Many of the tree-based GrSMs are generated manually by security experts, or using an algorithm with an exponential complexity [85]. A naive approach for automated generation is to populate all possible attack scenarios and group them into a single AT branch. Another method is to construct the AT using a decomposition method [17], but it is difficult to automate due to various requirements, such as defining a policy for decomposing a goal into sub-goals and locating independent components in the networked system, which becomes similar to the Manual method. Dawkins et al. [40] used min-cuts to populate AT branches, but computing min-cuts is an NP-hard problem [86]. Recent work on the automated generation of ATs [51] by automatically inferring from a process algebraic specification in a syntax-directed fashion. Their claim is that although the upper bound is an exponential complexity, the ability of SAT solvers to handle millions of variables make this approach feasible in this domain. More research work is done towards the automatic generation of tree-based GrSMs (such as in [85]), but it seems more efforts into scalable automatic generation methods for tree-based GrSMs are needed.

3.1.1. AT:

For ATs that can be expressed in a logical form, the analysis of an AT is to solve the logical expression of the AT with input security metrics (e.g., the probability of an attack success, impact value, cost of an attack/defense). ATs are combinatorial models, which do not enumerate all possible system states to obtain a solution [87]. Hence, the AT evaluation is dependent on the number of events (i.e., the number of nodes in the AT). Consequently, other tree-based GrSMs that use similar generation and evaluation methods to AT have the same properties and complexities. On the other hand, some ATs may not be expressed logically (e.g., incorporating cost evaluation cannot be interpreted logically [3]). However, even if the AT cannot be interpreted logically, the evaluation process involves some calculation associated with all event nodes in the AT.

Modification of the AT (in an event of an update in the networked system) has not been studied. If we assume there are no repeated events in the AT, then update events may occur to some AT nodes only and an addition of other correlated branches. If the observed update event in the network affects AT nodes near the root, a larger proportion of the AT is affected and requires more changes. If we consider repeated events, then any update event could affect the whole AT components depending on their correlations. A better layout of the AT is required to efficiently update the AT.

3.1.2. DT:

The properties are the same as the AT with an addition of defense nodes. However, the countermeasure nodes in the DT are manually created by security experts. As a result, the analysis approach using the DT is not automated.

3.1.3. OWAT:

The properties are the same to the AT.

3.1.4. PT:

Edge defines an algorithm for PT construction in [31]. The algorithm uses a decomposition method to construct the AT, but the author also described building the AT using all possible attack scenarios. However, the author only recommends using this method if the size of the networked system is small, because it suffers from a scalability problem. Using the PT, the same evaluation can be conducted as the DT, and the same problem exists with the AT in an event of updates in the networked system.

3.1.5. ART:

ART suffers from a scalability problem because a Markovian decision model is used in the evaluation phase, which generates an exponential number of states. Authors used two state-compaction techniques to improve the scalability; (i) Most Likely State (MLS) approximation and (ii) employing an online anytime algorithm called *Envelope*. These heuristic methods showed time improvement over an exhaustive search, but the recovery cost comparison clearly shows that it does not always achieve the optimal solution. The problem of reconstruction in tree-based GrSMs applies to the ART as same as other models and this is not covered in their work.

3.1.6. ADT:

The properties are the same to the AT with an addition of defense nodes. Authors have plans for automated generation method and also an extension to directed acyclic graphs.

3.1.7. ACT:

The construction of the ACT is based on using mincuts (as mentioned, the computation of mincuts is an NP-hard problem). The evaluation of the ACT extends the DT evaluation (e.g., using ROI and ROA) using probability parameters, cost, impact, Birnbaum importance measure and risk. The selection of an optimal countermeasure for an ACT is proposed in [20], which has a polynomial (cubic) computational complexity. The reconstruction or an update is not captured in their work.

3.1.8. AFT:

Authors do not specify how the AFT is generated, visualized, evaluated and updated explicitly. However, as the underlying mechanisms of AFT are similar to the AT, we could suggest they exhibit similar computational complexities.

Overall, the table shows that the generation of tree-based GrSMs suffers from a scalability problem due to inefficient research in automated generation methods (i.e., mostly *D*, *E* or *U*). The evaluation phase relies on the number of events in the tree-based GrSM, such that the exponential number of events created from the generation phase causes an exponential complexity. To make an evaluation of tree-based GrSMs more scalable, one must reduce the number of events created during the generation phase. Hong et al. [88] proposed Logic Reduction Techniques to improve the generation of tree-based GrSMs, which can be used for various tree-based GrSMs to reduce the number of events in the generation and representation phases.

3.2. Graph-based GrSMs

Graph-based GrSMs are mostly generated and represented in a polynomial complexity, but the evaluation phase still suffers from a scalability problem due to the state space explosion when exploring all sets of possible paths with a given graph. Most graph-based GrSMs adopted heuristic methods, and still there is no algorithm that can evaluate all possible states in a polynomial complexity. Also, there is a very limited number of research done on modifiability of graph-based GrSMs.

3.2.1. AG:

Initially, a manual generation method was used for AG, and automated generation method was proposed by Sheyner et al. [1] that had an exponential complexity. Ammann et al. [68] proposed more scalable generation of AG based on a monotonicity assumption, where the generation algorithm scaled at N^6 (N is the number of hosts in the networked system) [9].

Evaluation of graph-based GrSMs suffers from a scalability problem because exploring all possible set of paths with a given graph is an NP-hard problem [86]. Jha et al. [89] proposed a greedy algorithm to solve this problem, which allows a security administrator to analyze the attack graph and deploy security patches in a more cost-effective way. Since then, more heuristic methods are adopted for analyzing graph-based GrSMs widely. Wang et al. [90] proposed a near-optimal approximation algorithm that can evaluate the security with a linear complexity with respect to the size of the graph.

3.2.2. EDG:

EDG can be generated by exploring the correlations between exploits and conditions via forward and backward traversal [56]. With redundant cycles removed with monotonicity assumption, the EDG can be generated in a polynomial complexity. However, evaluation of the EDG yields an exponential complexity if one is to explore all possible states of an attack. To address this issue, a heuristic method is applied to consider only the minimal-impact scenarios that are dependent on the attack goal expression.

3.2.3. BAG:

BAG extended AG with adopting probabilistic security analysis using Bayesian networks. Hence, generating them is similar to generating an AG. However, the evaluation of Bayesian networks has an exponential complexity. To improve the efficiency of evaluating the BAG, other algorithms or dynamic Bayesian networks are adopted [73,74].

3.2.4. TVA:

TVA conducts a security analysis based on attack paths generated from the exploit dependency [16]. Hence, the scalability of TVA is equivalent to the AG. In [91] they described the TVA in the use of attack prevention, detection and response systems, and in [92] they extended the TVA with scalable AG generation algorithm. Theoretical performance analysis has a polynomial (quadratic) complexity and it can achieve a linear complexity in each domain of hosts with clustering, which they showed via an experiment. Noel et al. [76] extended TVA features for visualization, optimal tradeoffs between safety and availability, and scalable algorithms for larger networked systems.

3.2.5. LAG:

LAG can be generated in a polynomial (quadratic) complexity, and this is shown via complexity analysis and experimental results [58]. LAG is generated using a tool *MulVAL* [7], which uses the reasoning system with Datalog tuples and rules, where Datalog is a syntactic subset of Prolog. The evaluation for LAG can be addressed using clustering and pruning techniques [93].

3.2.6. MPAG:

Complexity analysis showed the generation of MPAG has a polynomial (quadratic) time complexity, and the experimental results showed in a practical scenario, the performance is almost linear [15]. Also, using graph simplification improved the scalability of evaluating the security. MPAG can be generated using NetSPA [6], and additional features, such as zero-day exploits, client-side attacks and countermeasures have been developed [94].

3.2.7. CG:

CG is designed for analyzing the risk in SCADA system [59]. The representation shown in the example indicates the polynomial complexity for the representation phase. However, there was no complexity analysis or performance measures given for generation and evaluation phases.

3.2.8. HAG:

Two-layer HAG can be generated in a polynomial complexity [60,81]. Xie et al. [60] used matrix multiplication for probabilistic security evaluation that can be computed in a polynomial computational complexity. However, this method cannot capture the sequential attack paths information. Hong and Kim [81] used an exhaustive search method that had an exponential computational complexity. Further heuristic methods were used to improve the scalability of the evaluation phase, scaling almost linearly [26,95,96].

3.2.9. CMG:

CMG is an extension of ATs, with the relationship between nodes extending to many-to-many, and mechanism for prioritization and countermeasures [61]. One can suspect the representation of the CMG to be polynomial based on the examples given, but there is no description of the generation method, and the evaluation only considered cost analysis based on a voting scheme.

3.2.10. ASG:

ASG is generated using both AG and dependency graph (DG) [62]. The evaluation phase consists of security analysis of new alerts, and this is solved using algorithms to update the index. A formal analysis shows an exponential complexity with respect to the number of vulnerability exploits, but the algorithm restricts the lookup table with only the most recent records to keep the complexity low. This is shown via experimental results that the index size is kept almost constant with a growing number of vulnerability exploits.

Table 3
Security metrics used.

	M1	M2	M3	M4	M5	M6	M7
AT 2.1.1	[22,39]	[98,99]	[17,36,100]	[21,25,39]	[48]		[101]
DT 2.1.2						[18,102]	
OWAT 2.1.3	[34]		[34]				
PT 2.1.4	[24,31]		[24,31]				
ART 2.1.5			[19]				
ADT 2.1.7	[103]	[103]	[3,33,103]	[3,33]			
ACT 2.1.6	[20,32,104]		[20,32,104]	[20,32,104]	[20,32,104]	[20,32,104]	
AFT 2.1.8	[35]		[35]	[35]	[35]		
AG 2.2.1	[105–107]	[13,55,108,109]	[110]	[111,112]	[89,113–115]	[111]	[116,117]
EDG 2.2.2			[56,118]		[56]		
BAG 2.2.3	[73,74,119,120]		[120]	[120]			[121]
TVA 2.2.4	[122]		[123–126]	[111]		[111]	[127]
LAG 2.2.5	[77]		[128]	[129]	[7]		
MPAG 2.2.6			[15,78]	[15]			[78]
CG 2.2.7		[59]					
HAG 2.2.8	[60]			[82]			
CMG 2.2.9			[61]				
ASG 2.2.10	[62]	[62]					[62]
AEG 2.2.11					[63,97]	[63,97]	[63,97]
CAG 2.2.12	[64,84]						
SAG 2.2.13	[65,130]						
IFG 2.2.14							[66]

Key: M1: Probability of Attack/ Protection Success, M2: Mean Time to Failure/ Compromise, M3: Attack/ Protection Cost, M4: Impact/Risk Analysis, M5: Attack Path Analysis, M6: Return on Investment/ Attack, M7: Intrusion Detection and Prevention

3.2.11. AEG:

The generation of AEG requires a manual input of attacks and adversaries from system and adversary experts [63]. Although generating an executable model is automated, there is no detail of its performance given. AEG captures all possible attack scenarios and the authors state the generation method is similar to AGs. Evaluation of AEG requires exploration of all possible attack scenarios using the breadth-first-search method in a tree structure from a given state s until the distance reaches N steps from s (s is a goal state reachable and N is an integer greater than 1) [97]. Pruning is applied to remove non-maximal attractive branches.

3.2.12. CAG:

CAG is an extension to the LAG, which inherits the same properties for all phases. CAG extends LAG by featuring the ability to model both known and unknown vulnerabilities, so the number of vulnerability nodes is increased. However, it is a fixed size so the generation and representation of CAG are still scalable. In the evaluation, however, CAG uses state enumeration which has a scalability problem due to state space explosion. In order to avoid this problem, CAG assumes the existence of unknown vulnerabilities without enumerating every one of them that reduces the state model. Breaking this assumption would yield an exponential computational complexity.

3.2.13. SAG:

SAG is automatically generated using a CyberSAGE tool [65], with a given manual input of the networked system information, attacker actions and capabilities. This is still being developed, so the functionalities of SAG are limited as described as the following: (i) The graph generation algorithm iteratively process all vertices in the graph, where a vertex is matched with a set of templates and extended with subcomponents. There is no performance analysis given (analytic or experiment), but the algorithm suggests a polynomial time complexity of $O(TV)$, where T is the number of templates and V is the number of vertices. (ii) There is no description of how to evaluate the SAG or which security evaluations are conducted.

3.2.14. IFG:

IFG is generated based on collected command messages of OpenFlow in the SDN. However, authors did not evaluate the computational complexity (or time complexity) to collect and generated the IFG. Given the algorithm in their work [66], the message processing takes $O(N)$ for N number of messages received for validation of reachability in the SDN. However, we still cannot determine the rate of OpenFlow command messages that are generated and used to generate the IFG. Evaluation is based on observing malicious traffic in the SDN, and the algorithm suggests it would be $O(M)$ for M bytes observed in the given flow of packets.

4. Application of GrSMs

4.1. Security metrics

The security metrics describe the different measures that can be collected using the GrSMs (e.g., what can I measure with different GrSMs?). We have enlisted and categorized most widely used metrics in security analysis, and this is summarized in Table 3. Since a lot of research related to GrSMs exist, we have only listed a few references for each category. The purpose of this table is to show which metrics can be used with different GrSMs, so listing all possible references here is unnecessary.

We have sorted various metrics into seven categories that are most widely used in practice. Note that some metrics are similar (or the same), but uses a different name (e.g., *mean time to security failure* and *mean time to system failure*). These metrics are put into the same category in this survey (e.g., *mean time to failure/compromise*) as their computational processes are in the same domain (i.e., computing a metric with respect to time). We can see that the most widely used metrics are *probability of attack/protection success* and *attack/protection cost* across 13 different GrSMs, followed by impact/risk analysis across 9 different GrSMs. Also, AT and ACT cover the most metric categories for tree-based GrSMs with five different metric categories, where AG covers the most of the metric categories for graph-based GrSMs with all metric categories available. More recent GrSMs (e.g., HAT, SAG) lack in applicable metrics available.

Table 4
Availability of tools.

Tool name	GrSM used	Source	Purpose
SeaMonster	AT 2.1.1	[131]	Research
AttackTree+	AT 2.1.1	[132]	Commercial
SeculTree	AT 2.1.1	[133]	Commercial
NuSMV	AG 2.2.1	[1]	Research
RedSeal	AG 2.2.1	[134]	Commercial
Skybox	AG 2.2.1	[135]	Commercial
TVA	EDG 2.2.4	[16]	Research
MulVAL	LAG 2.2.5	[7]	Research
Cauldron	AG 2.2.1	[136]	Commercial
NetSPA	MPAG 2.2.6	[6]	Research
GARNET	MPAG 2.2.6	[78]	Research
NAVIGATOR	MPAG 2.2.6	[8]	Research
ADVISE	AEG 2.2.11	[97]	Research
CyberSAGE	SAG 2.2.13	[65]	Research
Sphinx	IFG 2.2.14	[66]	Research
Safelite	HARM 2.3	[137]	Research
Firemon	MPAG 2.2.6	[138]	Commercial
CyGraph	AG 2.2.1	[139]	Research
ADTool 2.0	ADT 2.1.7	[140]	Research
ATSyRa	AT 2.1.1	[85]	Research
Attack Navigator	AT 2.1.1	[141]	Research

4.2. Availability of GrSM tools

There is a wide range of tools available to use different GrSMs. In this section, we enlist tools of GrSMs reported and available to the public (i.e., not just the model, but an GrSM that comes in the package) as shown in Table 4. For a list of tools including prototypes that are not yet available to the public can be found in [5]. We observe that there are many tools based on earlier versions of GrSMs (such as an AG and an AT), and a larger portion of them are tree-based GrSMs. However, recent tools incorporate newer GrSM for their underlying model (e.g., ADTool 2.0 using ADT), providing potential users with the benefits of utilizing the latest methods to assess and evaluate the security of systems.

4.3. Applicable GrSMs domains

In this section, we enlist the application of GrSMs in various domains. Most studies used a networked system as their security analysis domain (e.g., security analysis of enterprise systems). In this section, we explore a broader field of GrSM applications. The list of domains that GrSMs have application is shown in Table 5. Again, because there is a lot of research conducted for GrSMs, we only enlist a few examples of each GrSM applications in each domain.

ATs are most widely used in various domains of GrSM applications. Other GrSMs are used in different domains but they are mostly focused on networked system domain only. *E-commerce* and *Network Monitoring System* are two domains modeled by various GrSMs, where other domains are mainly modeled using ATs.

5. Discussion

Although the practice of using GrSMs for the security assessment has settled for more than a decade, there are still issues that current researchers must face and resolve, especially with a rapid evolution in networking technologies. These issues need to be addressed in order to provide efficient and effective security hardening techniques for modern networked systems, which are often large and dynamic. In this section, we describe some of the issues and open problems.

5.1. Practical issues of modeling security

There are a few practical issues of modeling security, especially in a heterogeneous environment where there are a large number of network components (e.g., hosts). Firstly, scalability is a big issue when modeling a large sized networked system due to state space explosion [9,15,58,69,81,82]. Table 2 shows that still not all phases of GrSMs are scalable, especially the modification phase. Although various solutions are proposed, they use heuristic methods that only consider a subset of all possible attack scenarios. This brings us the question: how can we analyze all possible attack scenarios in a scalable manner? We still need more scalable security modeling solutions to solve this question.

Secondly, GrSMs lack reusability. Various security metrics and their analysis methods are proposed but they cannot be transferred between GrSMs due to the structural difference of the latter. As a result, users must choose their GrSM concisely for their need by understanding the capabilities of their chosen GrSM. Table 3 shows various security metrics that different GrSMs can compute, but only the AG is capable of computing all categories of metrics listed. There are some recent advances in unifying the usability of GrSMs, such as work done by Dong et al. [151], where they collate and prepare security input data for the security assessment. However, it is still a challenge how to unify such data for the diverse family of GrSMs.

Thirdly, there is a lack of tools availability. Table 4 shows that with over 20 different GrSMs reviewed in this survey, only 10 GrSMs have tools available (not including prototypes) and only three GrSMs (i.e., AGs, ATs and MPAGs) have commercial tools. The lack of tools availability significantly reduces the scope choices the user can do for their security assessments.

5.2. Differences between tree-based and graph-based GrSMs

The purpose of network security decides which GrSM should be used (e.g., specified assets to be protected, or consider overall the security of the network), but there is a lack of comparison between graph-based and tree-based GrSMs to distinguish different features between them. Important factors are the capabilities to present attack scenarios and their implementation complexities (e.g., required procedures to generate such GrSMs). This survey provides some insight towards distinguishing the difference between those two types of GrSMs, but a closer comparison is needed to fully understand their capabilities and advantages/disadvantages. We further discuss some of their differences in terms of attack scenarios and implementations below.

5.2.1. Attack scenarios

The capabilities to capture and represent attacks are dependent on the ability to express various attack scenarios, various enumerations, abstraction and scalability. Attack scenarios can be captured and presented as attack paths, which are logical sequences of vulnerabilities exploited and hosts compromised in the network. The attack paths information is not always required in a security analysis, such as considering an overall network security, where only the most critical vulnerabilities in the systems could be analyzed. In this case, a tree-based GrSM may be preferred because there is no need to consider sequences of attacks. However, if the ability to capture attack paths is critical (e.g., to characterize attacks), then a graph-based GrSM may be preferred.

Enumerating graph-based GrSMs captures all possible attack paths that can be useful for discovering the criticality of individual attack path. In comparison, enumerating the AT is based on their logical connections (e.g., according to their logical gates), which only capture how the goal (or a sub-goal) may be satisfied based on the combination of events. Unless logical connections that capture

Table 5
GrSMs application in various domains.

Application domain	GrSM	Source	Analysis approach
E-commerce system	AG 2.2.1	[142]	Intrusion detection and response
	PT 2.1.4	[24,31]	Protection probability and implementation cost
	AT 2.1.1	[17,143]	Cost of attack and impact analysis
Network monitoring system	AG 2.2.1	[116,117]	Learn attack strategies from intrusion alerts
	TVA 2.2.4	[126]	Correlating, hypothesizing and predicting alerts
	BAG 2.2.3	[119]	Placement of intrusion detectors
	AT 2.1.1	[99]	Detection rate for trojan horse
Electric power grid	SAG 2.2.13	[65,130]	Probabilistic analysis of availability and confidentiality
Online game system	CMG 2.2.9	[61]	Countermeasure selection via attack cost analysis
Software development	AG 2.2.1	[144,145]	Vulnerability elimination during software development
SCADA system	AT 2.1.1	[21,146]	Risk assessment
Phased-mission system	AT 2.1.1	[147]	Reliability assessment
Electronic copyright management	AT 2.1.1	[36,37]	Attack cost, risk, skill, accessibility, and impact
Mobile and ad hoc network	AT 2.1.1	[98]	Assessment of packet drop rates
Airline system	AT 2.1.1	[25]	Qualitative risk analysis
E-voting system	AT 2.1.1	[22]	Probabilistic analysis
RFID system	AT 2.1.1	[148]	Qualitative confidentiality, integrity, availability, and risk
Cyber-physical system	EDG 2.2.2	[149,150]	Analysis description of denial of sleep attack
Pipeline system	AFT 2.1.8	[35]	As-is and what-if security and safety analysis

the sequential information (such as $S-AND$ gates) are used for tree-based GrSMs, it cannot directly capture the sequence of attack paths. On the other hand, tree-based GrSMs do not suffer the state space explosion when enumerating events, as it is only dependent on the number of events modeled. Therefore, a scalable generation of tree-based GrSMs results in scalable evaluation as well.

Abstraction and scalability are dependent on the structure of GrSMs, and these are covered in Section 3. Table 2 clearly shows that although generating and representing GrSMs are scalable (especially for graph-based GrSMs), there are still needs for scalable evaluation and modification of GrSMs.

5.2.2. Implementation complexities

The complexity of implementing GrSMs depends on the procedure of gathering information from various sources (e.g., network reachability, vulnerabilities). To construct a typical GrSM considering attacks and possible countermeasures, one needs information on machine configurations (e.g., active services), vulnerabilities, attacks and countermeasures at the host level, and network configurations (e.g., reachability) and firewall settings (e.g., filtering rules) at the network level. An approach is to generate databases for information required at the host level, and combine with information from the network level to generate the GrSM. For example, *MulVAL* [7] uses various network tools available in public to populate its databases (e.g., using a vulnerability scanner *NESSUS* [152]). Using these tools can be time-consuming, and they also require a transformation of raw data for a more suitable format. Moreover, these records must be synchronized in an event of an update.

5.3. Usability of GrSMs

GrSMs are widely used to analyze security in various domains (as shown in Section 4), and they can evaluate many different aspects of security (as shown in Section 3). However, some GrSMs can analyze the security more efficiently depending on various factors (e.g., the structure of the networked system, security metrics used), as well as ability to capture different security information (e.g., attack paths). For example, tree-based GrSMs can efficiently perform security analysis (assuming the tree structure is optimized), but it cannot capture the sequential information without the use of specialized features (e.g., using sequential-*AND* gates). However, the generation of graph-based GrSMs is scalable, achieving the polynomial time complexity, where the generation of tree-based GrSMs still has a state space explosion when we consider an automated generation of a large sized networked system. That is, the user must make a decision on which type of GrSMs will be best suited for the security analysis, considering various features of GrSMs and how suitable the GrSM is for the intention of the user.

5.4. Open issues

Firstly, most GrSMs do not incorporate how countermeasure selection may affect the system performance. There are studies considering both security and performance, but these are not captured in GrSMs. As a result, users may know the effective countermeasures, but they cannot be sure how much performance in their system will change when countermeasures are implemented.

Secondly, different GrSMs have different features to enhance the security analysis. However, these features are specifically made for each GrSMs, such that different capabilities of different GrSMs cannot be shared. This becomes cumbersome from the user's point of view. For example, if the user has used an AG to evaluate the risk of the system but now wants to evaluate the risk of the network based on different reachability groups, the user must reconstruct the AG for each reachability group. However, this could have been done without reconstruction if the user used MPAG 2.2.6 [94] from the beginning of his/her security analysis, which has a feature to capture different reachability groups.

Lastly, networked systems are growing in size with dynamic properties (e.g., growing number of mobile devices), but there is a lack of solutions to deal with the scalability and adaptability. This is summarized in Table 2, where the automated performance is in exponential complexity or even done by a manual analysis for different phases of GrSMs. Graph-based GrSMs can be generated in polynomial complexity, but the evaluation phase has an exponential complexity to cover all set of attack paths or uses heuristic methods. However, many heuristic methods are proposed that addresses the scalability issues in the evaluation phase nowadays. Tree-based GrSMs can evaluate the security in a scalable manner with respect to polynomial size complexity, but there is a lack of efficient generation algorithms for tree-based GrSMs. As a result, we still need more robust methods of graph-based GrSM evaluation and tree-based generation methods, as well as research into how to capture changes in the networked system efficiently in GrSMs.

6. Conclusion

Numerous GrSMs are proposed in the past to enhance the security assessment of networked systems in an efficient and effective way. However, the diversity of those models brought users with

a difficulty of comprehending their wide scope of functionalities, such that users cannot decide which models provide functions and methods required for their needs. In this survey, we have compared and evaluated various aspects of security models in terms of GrSM phases, security metrics, available tools and various GrSM application domains. As a result, this paper can provide an insight for users to determine the most suitable GrSM to effectively address their security concerns. Lastly, this survey identified key problems of GrSMs as future work, such as the practical issues (e.g., adaptability), the difference between the tree and graph-based models, and the usability of GrSMs (e.g., interoperability).

Acknowledgment

This research was partially supported by the NATO Science for Peace & Security Multi-Year Project (MD.SFPP 984425), NSF SaTC CNS 1528099 research grant, and by Grant NPRP 8-531-1-111 from Qatar National Research Fund (QNRF). The statements made herein are solely the responsibility of the authors.

References

[1] O. Sheyner, J. Haines, S. Jha, R. Lippmann, J. Wing, Automated generation and analysis of attack graphs, in: Proc. of IEEE Symposium on Security and Privacy (S&P 2002), IEEE, 2002, pp. 273–284. http://dx.doi.org/10.1109/SECPRI.2002.1004377.

[2] B. Schneier, Secrets and lies: Digital security in a networked world, John Wiley and Sons Inc., 2000.

[3] B. Kordy, S. Mauw, S. Radomirovic, P. Schweitzer, Attackdefense trees, J. log. comput. 24 (2014) 55–87. http://dx.doi.org/10.1093/logcom/exs029. http://logcom.oxfordjournals.org/content/early/2012/06/21/logcom.exs029.abstract.

[4] J. Hong, D. Kim, Towards scalable security analysis using multi-layered security models, J. Netw. Comput. Appl. 75 (2016) 156–168. http://dx.doi.org/10.1016/j.jnca.2016.08.024. http://www.sciencedirect.com/science/article/pii/S1084804516301928.

[5] B. Kordy, L. Pietre-Cambacedes, P. Schweitzer, DAG-based attack and defense modeling: Don't miss the forest for the attack trees, Comput. Sci. Rev. 1314 (2014) 1–38. http://dx.doi.org/10.1016/j.cosrev.2014.07.001. http://www.sciencedirect.com/science/article/pii/S1574013714000100.

[6] M. Artz, NetSPA : A network security planning architecture, Massachusetts Institute of Technology, 2002.

[7] X. Ou, S. Govindavajhala, A. Appel, MulVAL: A logic-based network security analyzer, in: Proc. of the 14th USENIX Security Symposium (USENIX Security 2005), 2005, pp. 113–128.

[8] M. Chu, K. Ingols, R. Lippmann, S. Webster, S. Boyer, Visualizing attack graphs, reachability, and trust relationships with NAVIGATOR, in: Proc. of the 7th International Symposium on Visualization for Cyber Security (VizSec 2010), VizSec '10, ACM, New York, NY, USA, 2010, pp. 22–33. http://dx.doi.org/10.1145/1850795.1850798. http://doi.acm.org/10.1145/1850795.1850798.

[9] R. Lippmann, K. Ingols, An annotated review of past papers on attack graphs, technical report, MIT Lincoln Lab, 2005.

[10] S. Khaitan, S. Raheja, Finding optimal attack path using attack graphs: A survey, Int. J. Soft Comput. Eng. 1 (2011) 2231–2307.

[11] M. Alhomidi, M. Reed, Attack graphs representations, in: Proc. of Computer Science and Electronic Engineering Conference (CEEC 2012), 2012, pp. 83–88. http://dx.doi.org/10.1109/CEEC.2012.6375383.

[12] P. Mell, T. Grance, SP 800-145. The NIST Definition of Cloud Computing, NIST, Gaithersburg, MD, United States, 2011.

[13] L. Swiler, C. Phillips, D. Ellis, S. Chakerian, Computer-attack graph generation tool, in: Proc. of DARPA Information Survivability Conference Exposition II, (DISCEX 2001), Vol. 2, 2001, pp. 307–321. http://dx.doi.org/10.1109/DISCEX.2001.932182.

[14] O. Sheyner, Scenario Graphs and Attack Graphs, Ph.D. thesis, Carnegie Mellon University, Pittsburgh, PA, USA, 2004.

[15] K. Ingols, R. Lippmann, K. Piwowarski, Practical attack graph generation for network defense, in: Proc. of the 22nd Annual Computer Security Applications Conference (ACSAC 2006), IEEE, 2006, pp. 121–130. http://dx.doi.org/10.1109/ACSAC.2006.39.

[16] S. Jajodia, S. Noel, B. OBerry, Topological analysis of network attack vulnerability, in: V. Kumar, J. Srivastava, A. Lazarevic (Eds.), Managing Cyber Threats, in: Massive Computing, Vol. 5, Springer US, 2005, pp. 247–266. http://dx.doi.org/10.1007/0-387-24230-9.9.

[17] V. Saini, Q. Duan, V. Paruchuri, Threat modeling using attack trees, J. Comput. Sci. Coll. 23 (2008) 124–131. http://dl.acm.org/citation.cfm?id=1352079.1352100.

[18] S. Bistarelli, F. Fioravanti, P. Peretti, Defense trees for economic evaluation of security investments, in: Proc. of the First International Conference on Availability, Reliability and Security (ARES 2006), 2006, pp. 337–350. http://dx.doi.org/10.1109/ARES.2006.46.

[19] S. Zonouz, H. Khurana, W. Sanders, T. Yardley, RRE: A game-theoretic intrusion response and recovery engine, in: Proc. of IEEE/IFIP International Conference on Dependable Systems Networks (DSN 2009), 2009, pp. 439–448. http://dx.doi.org/10.1109/DSN.2009.5270307.

[20] A. Roy, D. Kim, K. Trivedi, Scalable optimal countermeasure selection using implicit enumeration on attack countermeasure trees, in: Proc. of the 42nd Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN 2012), IEEE Computer Society, Los Alamitos, CA, USA, 2012, pp. 1–12 http://doi.ieeecomputersociety.org/10.1109/DSN.2012.6263940.

[21] A. Moore, R. Ellison, R. Linger, Attack Modeling for Information Security and Survivability, Carnegie Mellon University, Software Engineering Institute, 2001.

[22] A. Buldas, T. Magi, Practical security analysis of e-voting systems, in: A. Miyaji, H. Kikuchi, K. Rannenberg (Eds.), Advances in Information and Computer Security, Lecture Notes in Computer Science, Vol. 4752, Springer Berlin Heidelberg, 2007, pp. 320–335. http://dx.doi.org/10.1007/978-3-540-75651-4.22.

[23] A. Hahn, M. Govindarasu, Smart Grid Cybersecurity Exposure Analysis and Evalution Framework, in: Proc. of IEEE Power and Energy Society General Meeting (PES GM 2010), 2010, pp. 1–6. http://dx.doi.org/10.1109/PES.2010.5590152.

[24] K. Edge, R. Raines, M. Grimaila, R. Baldwin, R. Bennington, C. Reuter, The Use of Attack and Protection Trees to Analyze Security for an Online Banking System, in: Proc. of the 40th Annual Hawaii International Conference on System Sciences (HICSS 2007), 2007, pp. 144–151.

[25] D. Balzarotti, M. Monga, S. Sicari, Assessing the Risk of Using Vulnerable Components, in: D. Gollmann, F. Massacci, A. Yautsiukhin (Eds.), Quality of Protection, Advances in Information Security, Vol. 23, Springer US, 2006, pp. 65–77. http://dx.doi.org/10.1007/978-0-387-36584-8.6.

[26] J. Hong, D. Kim, Scalable security model generation and analysis using k-importance measure, in: Proc. of the 9th International Conference on Security and Privacy in Communication Networks (SecureComm 2013), Springer International Publishing, Cham, 2013, pp. 270–287. http://dx.doi.org/10.1007/978-3-319-04283-1.17.

[27] J. Hong, Scalable and Adaptable Security Modelling and Analysis, Ph.D. Thesis, University of Canterbury, 2015.

[28] J. Peterson, Petri net Theory and The Modeling of Systems, Prentice Hall PTR, Upper Saddle River, NJ, USA, 1981.

[29] C. Hirel, B. Tuffin, K. Trivedi, SPNP: Stochastic petri nets. version 6.0, in: Computer Performance Evaluation. Modelling Techniques and Tools, Springer, 2000, pp. 354–357.

[30] B. Schneier, Modeling security threats, Dr. Dobb'S Journal 24 (1999).

[31] K. Edge, A Framework for Analyzing and Mitigating the Vulnerabilities of Complex Systems Via Attack and Protection Trees, Ph.D. Thesis, Air Force Institute of Technology, Wright Patterson AFB, OH, USA, AAI3305523, 2007.

[32] A. Roy, D. Kim, K. Trivedi, Cyber security analysis using attack countermeasure trees, in: Proc. of the Sixth Annual Workshop on Cyber Security and Information Intelligence Research (CSIIRW 2010), in: CSIIRW '10, ACM, New York, NY, USA, 2010, pp. 28:1–28:4. http://dx.doi.org/10.1145/1852666.1852698. http://doi.acm.org/10.1145/1852666.1852698.

[33] B. Kordy, S. Mauw, S. Radomirović, P. Schweitzer, Foundations of attack–defense trees, in: P. Degano, S. Etalle, J. Guttman (Eds.), Formal Aspects of Security and Trust, Lecture Notes in Computer Science, Vol. 6561, Springer, 2011, pp. 80–95. http://dx.doi.org/10.1007/978-3-642-19751-2.6.

[34] R. Yager, OWA trees and their role in security modeling using attack trees, Information Sciences 176 (2006) 2933–2959.

[35] R. Kumar, M. Stoelinga, Quantitative security and safety analysis with attack-fault trees, in: Proc. of the 18th IEEE International Symposium on High Assurance Systems Engineering (HASE 2017), 2017, pp. 25–32. http://dx.doi.org/10.1109/HASE.2017.12.

[36] M. Higuero, J. Unzilla, E. Jacob, P. Saiz, M. Aguado, D. Luengo, Application of 'attack trees' in security analysis of digital contents E-commerce protocols with copyright protection, in: Proc. of the 39th Annual International Carnahan Conference on Security Technology (CCST 2005), 2005, pp. 57–60. http://dx.doi.org/10.1109/CCST.2005.1594865.

[37] M. Higuero, J. Unzilla, E. Jacob, P. Saiz, M. Aguado, D. Luengo, Application of 'attack trees' technique to copyright protection protocols using watermarking and definition of a new transactions protocol SecDP (Secure Distribution Protocol), in: V. Roca, F. Rousseau (Eds.), Interactive Multimedia and Next Generation Networks, Lecture Notes in Computer Science, 3311, Springer Berlin Heidelberg, 2004, pp. 264–275. http://dx.doi.org/10.1007/978-3-540-30493-7.24.

[38] I. Fovino, M. Masera, Through the description of attacks: A multidimensional view, in: Proc. of the 25th International Conference on Computer Safety, Reliability, and Security (SAFECOMP 2006), SAFECOMP'06, Springer-Verlag, Berlin, Heidelberg, 2006, pp. 15–28. http://dx.doi.org/10.1007/11875567.2.

[39] I. Ray, N. Poolsapassit, Using attack trees to identify malicious attacks from authorized insiders, in: S. di Vimercati, P. Syverson, D. Gollmann (Eds.), Computer Security ESORICS 2005, Lecture Notes in Computer Science, Vol. 3679, Springer Berlin Heidelberg, 2005, pp. 231–246. http://dx.doi.org/10.1007/11555827.14.

[40] J. Dawkins, J. Hale, A systematic approach to multi-stage network attack analysis, in: Proc. of 2nd IEEE International Information Assurance Workshop (IWIA 2004), 2004, pp. 48–56. http://dx.doi.org/10.1109/IWIA.2004.1288037.

[41] X. Zang, H. Sun, K. Trivedi, A bdd-based algorithm for reliability graph analysis, Department of Electrical Engineering, Duke University, Tech. Rep (2000).

[42] R. Dewri, N. Poolsappasit, I. Ray, D. Whitley, Optimal security hardening using multi-objective optimization on attack tree models of networks, in: Proc. of the 14th ACM Conference on Computer and Communications Security (CCS 2007), ACM, New York, NY, USA, 2007, pp. 204–213. http://dx.doi.org/10.1145/1315245.1315272. http://doi.acm.org/10.1145/1315245.1315272.

[43] S. Mauw, M. Oostdijk, in: Foundations of Attack Trees, Springer Berlin Heidelberg, Berlin, Heidelberg, 2006, pp. 186–198. http://dx.doi.org/10.1007/11734727.17.

[44] J. Weiss, A System Security Engineering Process, in: Proc. of the 14th National Computer Security Conference, Vol. 249, 1991, pp. 571–580.

[45] E.G. Amoroso, in: Fundamentals of Computer Security Technology, Prentice-Hall, Inc., 1994.

[46] M. Audinot, S. Pinchinat, in: On the Soundness of Attack Trees, Springer International Publishing, Cham, 2016, pp. 25–38. http://dx.doi.org/10.1007/978-3-319-46263-9.2.

[47] M. Audinot, S. Pinchinat, B. Kordy, Is my attack tree correct?in: D. Gollmann, S. Foley (Eds.), Computer Security – ESORICS 2017, Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2017, pp. 88–102. http://dx.doi.org/10.1007/978-3-319-66402-6_7.

[48] R. Jhawar, B. Kordy, S. Mauw, S. Radomirović, R. Trujillo-Rasua, attack trees with sequential conjunction, in: Proc. of the 11th IFIP TC International Conference on ICT Systems Security and Privacy Protection (SEC 2015), Springer International Publishing, Cham, 2015, pp. 339–353. http://dx.doi.org/10.1007/978-3-319-18467-8.23.

[49] R. Horne, S. Mauw, A. Tiu, Semantics for specialising attack trees based on linear logic, Fundamenta Informaticae 153 (2017) 57–86. http://dx.doi.org/10.3233/FI-2017-1531.

[50] S. Vidalis, A. Jones, Using vulnerability trees for decision making in threat assessment, in: Proc. of the 2nd European Conference on Information Warfare and Security (ECIW 2003), Academic Conferences Limited, 2003, pp. 329–342.

[51] R. Vigo, F. Nielson, H. Nielson, Automated generation of attack trees, in: Proc. of the 27th IEEE Computer Security Foundations Symposium (CSF 2014), 2014, pp. 337–350. http://dx.doi.org/10.1109/CSF.2014.31.

[52] M. Ivanova, C. Probst, R. Hansen, F. Kammüller, Transforming graphical system models to graphical attack models, in: S. Mauw, B. Kordy, S. Jajodia (Eds.), Proc. of the Second International Workshop on Graphical Models for Security (GraMSec 2015), Springer International Publishing, 2016, pp. 82–96. http://dx.doi.org/10.1007/978-3-319-29968-6.6.

[53] Z. Aslanyan, F. Nielson, D. Parker, Quantitative verification and synthesis of attack-defence scenarios, in: Proc. of the 29th IEEE Computer Security Foundations Symposium (CSF 2016), 2016, pp. 105–119. http://dx.doi.org/10.1109/CSF.2016.15.

[54] M. Fraile, M. Ford, O. Gadyatskaya, R. Kumar, M. Stoelinga, R. Trujillo-Rasua, in: Using Attack-Defense Trees to Analyze Threats and Countermeasures in an ATM: A Case Study, Springer International Publishing, Cham, 2016, pp. 326–334. http://dx.doi.org/10.1007/978-3-319-48393-1.24.

[55] C. Phillips, L. Swiler, A graph-based system for network-vulnerability analysis, in: Proc. of the Workshop on New Security Paradigms (NSPW 1998), in: NSPW '98, ACM, New York, NY, USA, 1998, pp. 71–79. http://dx.doi.org/10.1145/310889.310919.

[56] S. Noel, S. Jajodia, B. O'Berry, M. Jacobs, Efficient minimum-cost network hardening via exploit dependency graphs, in: Proc. of the 19th Annual Computer Security Applications Conference (ACSAC 2003), IEEE, 2003, pp. 86–95. http://dx.doi.org/10.1109/CSAC.2003.1254313.

[57] Y. Liu, H. Man, Network vulnerability assessment using Bayesian networks, in: B.V. Dasarathy (Ed.), Data Mining, Intrusion Detection, Information Assurance, and Data Networks Security 2005, Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series, Vol. 5812, 2005, pp. 61–71. http://dx.doi.org/10.1117/12.604240.

[58] X. Ou, W. Boyer, M. McQueen, A scalable approach to attack graph generation, in: Proc. of the 13th ACM Conference on Computer and Communications Security (CCS 2006), ACM, 2006, pp. 336–345. http://dx.doi.org/10.1145/1180405.1180446. http://doi.acm.org/10.1145/1180405.1180446.

[59] M. McQueen, W. Boyer, M. Flynn, G. Beitel, Quantitative Cyber Risk Reduction Estimation Methodology for a Small SCADA control system, in: Proc. of the 39th Annual Hawaii International Conference on System Science (HICSS 2006), Vol. 9, 2006. http://dx.doi.org/10.1109/HICSS.2006.405. 226–226.

[60] A. Xie, Z. Cai, C. Tang, J. Hu, Z. Chen, Evaluating network security with two-layer attack graphs, in: Proc. of Annual Computer Security Applications Conference (ACSAC 2009), 2009, pp. 127–136. http://dx.doi.org/10.1109/ACSAC.2009.22.

[61] D. Baca, K. Petersen, Prioritizing countermeasures through the countermeasure method for software security (CM-Sec), in: M.A. Babar, M. Vierimaa, M. Oivo (Eds.), Product-Focused Software Process Improvement, in: Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2010, pp. 176–190.

[62] M. Albanese, S. Jajodia, A. Pugliese, V. Subrahmanian, Scalable analysis of attack scenarios, in: V. Atluri, C. Diaz (Eds.), Computer Security – ESORICS 2011, in: Lecture Notes in Computer Science, Vol. 6879, Springer Berlin Heidelberg, 2011, pp. 416–433. http://dx.doi.org/10.1007/978-3-642-23822-2.23.

[63] E. LeMay, W. Unkenholz, D. Parks, C. Muehrcke, K. Keefe, W. Sanders, Adversary-driven state-based system security evaluation, in: Proc. of the 6th International Workshop on Security Measurements and Metrics (MetriSec 2010), in: MetriSec '10, ACM, New York, NY, USA, 2010, pp. 5:1–5:9. http://dx.doi.org/10.1145/1853919.1853926. http://doi.acm.org/10.1145/1853919.1853926.

[64] R. Zhuang, S. Zhang, S. DeLoach, X. Ou, A. Singhal, Simulation-based approaches to studying effectiveness of moving-target network defense, in: Proc. of National Symposium on Moving Target Research (MTD 2012), 2012, pp. 1–12. https://www.nist.gov/publications/simulation-based-approaches-studying-effectiveness-moving-target-network-defense.

[65] N. Tippenhauer, W. Temple, A. Hoa Vu, B. Chen, D. Nicol, Z. Kalbarczyk, W. Sanders, Automatic generation of security argument graphs, in: Proc. of the 20th IEEE Pacific Rim International Symposium on Dependable Computing (PRDC 2014), 2014, pp. 33–42. http://dx.doi.org/10.1109/PRDC.2014.13.

[66] M. Dhawan, R. Poddar, K. Mahajan, V. Mann, Sphinx: Detecting security attacks in software-defined networks, in: Proc. of the Network and Distributed System Security Symposium (NDSS 2015), 2015, pp. 1–15.

[67] M. Dacier, Y. Deswarte, Privilege graph: An extension to the typed access matrix model, in: D. Gollmann (Ed.), Computer Security — ESORICS 1994, in: Lecture Notes in Computer Science, Vol. 875, Springer Berlin Heidelberg, 1994, pp. 319–334. http://dx.doi.org/10.1007/3-540-58618-0.72.

[68] P. Ammann, D. Wijesekera, S. Kaushik, Scalable, graph-based network vulnerability analysis, in: Proc. of the 9th ACM Conference on Computer and Communications Security (CCS 2002), ACM, 2002, pp. 217–224.

[69] S. Noel, S. Jajodia, Managing attack graph complexity through visual hierarchical aggregation, in: Proc. of the 2004 ACM Workshop on Visualization and Data Mining for Computer Security (VizSEC 2004), ACM, New York, NY, USA, 2004, pp. 109–118. http://dx.doi.org/10.1145/1029208.1029225.

[70] R. Ritchey, P. Ammann, Using model checking to analyze network vulnerabilities, in: Proc. of IEEE Symposium on Security and Privacy (S&P 2000), 2000, pp. 156–165. http://dx.doi.org/10.1109/SECPRI.2000.848453.

[71] L. Apvrille, Y. Roudier, in: SysML-Sec Attack Graphs: Compact Representations for Complex Attacks, Springer International Publishing, Cham, 2016, pp. 35–49. http://dx.doi.org/10.1007/978-3-319-29968-6.3.

[72] S. Noel, M. Jacobs, P. Kalapa, S. Jajodia, Multiple coordinated views for network attack graphs, in: Proc. of IEEE Workshop on Visualization for Computer Security (VizSEC 2005), 2005, pp. 99–106. http://dx.doi.org/10.1109/VIZSEC.2005.1532071.

[73] M. Frigault, L. Wang, Measuring network security using bayesian network-based attack graphs, in: Proc. of the 32nd Annual IEEE International Computer Software and Applications (COMPSAC 2008), 2008, pp. 698–703. http://dx.doi.org/10.1109/COMPSAC.2008.88.

[74] M. Frigault, L. Wang, A. Singhal, S. Jajodia, Measuring network security using dynamic Bayesian network, in: Proc. of the 4th ACM Workshop on Quality of Protection (QoP 2008), in: QoP '08, ACM, New York, NY, USA, 2008, pp. 23–30. http://dx.doi.org/10.1145/1456362.1456368.

[75] R. Ritchey, B. O"Berry, S. Noel, Representing TCP/IP connectivity for topological analysis of network security, in: Proc. of 18th Annual Computer Security Applications Conference (ACSAC 2002), 2002, pp. 25–31. http://dx.doi.org/10.1109/CSAC.2002.1176275.

[76] S. Noel, M. Elder, S. Jajodia, P. Kalapa, S. O'Hare, K. Prole, Advances in topological vulnerability analysis, in: Proc. of Cybersecurity Applications Technology Conference for Homeland Security (CATCH 2009), 2009, pp. 124–129. http://dx.doi.org/10.1109/CATCH.2009.19.

[77] R. Sawilla, X. Ou, Identifying critical attack assets in dependency attack graphs, in: S. Jajodia, J. Lopez (Eds.), Computer Security - ESORICS 2008, Lecture Notes in Computer Science, Vol. 5283, Springer Berlin / Heidelberg, 2008, pp. 18–34.

[78] L. Williams, R. Lippmann, K. Ingols, GARNET: A graphical attack graph and reachability network evaluation tool, in: J. Goodall, G. Conti, K.-L. Ma (Eds.), Visualization for Computer Security, Lecture Notes in Computer Science, 5210, Springer Berlin Heidelberg, 2008, pp. 44–59. http://dx.doi.org/10.1007/978-3-540-85933-8.5.

[79] J. Lee, H. Lee, H. In, Scalable attack graph for risk assessment, in: Proc. of International Conference on Information Networking (ICOIN 2009), 2009, pp. 1–5.

[80] D. Leversage, E. Byres, Comparing electronic battlefields: Using mean time-to-compromise as a comparative security metric, in: V. Gorodetsky, I. Kotenko, V. Skormin (Eds.), Computer Network Security, Communications in Computer and Information Science, Vol. 1, Springer Berlin Heidelberg, 2007, pp. 213–227. http://dx.doi.org/10.1007/978-3-540-73986-9.18.

[81] J. Hong, D. Kim, HARMs: Hierarchical attack representation models for network security analysis, in: Proc. of the 10th Australian Information Security Management Conference on SECAU Security Congress (SECAU 2012), 2012, pp. 74–81.

[82] J. Hong, D. Kim, in: Performance Analysis of Scalable Attack Representation Models, Springer Berlin Heidelberg, Berlin, Heidelberg, 2013, pp. 330–343. http://dx.doi.org/10.1007/978-3-642-39218-4.25.

[83] P. Berander, M. Svahnberg, Evaluating two ways of calculating priorities in requirements hierarchies–an experiment on hierarchical cumulative voting, J. Syst. Softw. 82 (2009) 836–850. http://dx.doi.org/10.1016/j.jss.2008.11.841. http://www.sciencedirect.com/science/article/pii/S0164121208002665.

[84] S. DeLoach, X. Ou, R. Zhuang, S. Zhang, Model-driven, moving-target defense for enterprise network security, in: N. Bencomo, R. France, B.H.C. Cheng, U. Aßmann (Eds.), Lecture Notes in Computer Science, Springer International Publishing, 2014, pp. 137–161. http://dx.doi.org/10.1007/978-3-319-08915-7.5.

[85] S. Pinchinat, M. Acher, D. Vojtisek, in: ATSyRa: An Integrated Environment for Synthesizing Attack Trees, Springer International Publishing, Cham, 2016, pp. 97–101. http://dx.doi.org/10.1007/978-3-319-29968-6.7.

[86] M. Bruglieri, F. Maffioli, M. Ehrgott, Cardinality constrained minimum cut problems: Complexity and algorithms, Discrete Appl. Math. 137 (2004) 311–341. http://dx.doi.org/10.1016/S0166-218X(03)00358-5.

[87] D. Nicol, W. Sanders, K. Trivedi, Model-based evaluation: From dependability to security, IEEE Transactions on Dependable and Secure Computing (TDSC) 1 (2004) 48–65. http://dx.doi.org/10.1109/TDSC.2004.11.

[88] J. Hong, D. Kim, T. Takaoka, Scalable attack representation model using logic reduction techniques, in: Proc. of the 12th IEEE International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom 2013), 2013, pp. 404–411. http://dx.doi.org/10.1109/TrustCom.2013.51.

[89] S. Jha, O. Sheyner, J. Wing, Two formal analyses of attack graphs, in: Proc. of the 15th IEEE Computer Security Foundations Workshop (CSFW 2002), 2002, pp. 49–63. http://dx.doi.org/10.1109/CSFW.2002.1021806.

[90] L. Wang, M. Albanese, S. Jajodia, Linear-time network hardening, in: Network Hardening, in: SpringerBriefs in Computer Science, Springer International Publishing, 2014, pp. 39–58. http://dx.doi.org/10.1007/978-3-319-04612-9.5.

[91] S. Jajodia, S. Noel, Topological vulnerability analysis: A powerful new approach for network attack prevention, detection, and response, in: Algorithms, Architectures, and Information Systems Security, World Scientific, 2007, pp. 1–20.

[92] S. Jajodia, S. Noel, Topological vulnerability analysis, in: S. Jajodia, P. Liu, V. Swarup, C. Wang (Eds.), Cyber Situational Awareness, Advances in Information Security, Vol. 46, Springer US, 2010, pp. 139–154. http://www.springerlink.com/content/p562083831v2214m/abstract/.

[93] J. Homer, A. Varikuti, X. Ou, M. Mcqueen, Improving attack graph visualization through data reduction and attack grouping, in: Proc. of the 5th International Workshop on Visualization for Computer Security (VizSec 2008), VizSec '08, Springer-Verlag, Berlin, Heidelberg, 2008, pp. 68–79. http://dx.doi.org/10.1007/978-3-540-85933-8.7.

[94] K. Ingols, M. Chu, R. Lippmann, S. Webster, S. Boyer, Modeling modern network attacks and countermeasures using attack graphs, in: Proc. of Annual Computer Security Applications Conference (ACSAC 2009), 2009, pp. 117–126. http://dx.doi.org/10.1109/ACSAC.2009.21.

[95] J. Hong, D. Kim, Scalable security analysis in hierarchical attack representation model using centrality measures, in: Proc. of the 43rd Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshop (DSNW 2013), 2013, pp. 1–8. http://dx.doi.org/10.1109/DSNW.2013.6615507.

[96] J. Hong, D. Kim, A. Haqiq, What vulnerability do we need to patch first?in: Proc. of the 44th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshop (DSNW 2014), 2014, pp. 684–689. http://dx.doi.org/10.1109/DSN.2014.68.

[97] E. LeMay, M. Ford, K. Keefe, W. Sanders, C. Muehrcke, Model-based security metrics using adversary view security evaluation (advise), in: Quantitative Evaluation of Systems (QEST), 2011 Eighth International Conference on, 2011, pp. 191–200. http://dx.doi.org/10.1109/QEST.2011.34.

[98] F. Kargl, A. Klenk, S. Schlott, M.l Weber, Advanced detection of selfish or malicious nodes in ad hoc networks, in: C. Castelluccia, H. Hartenstein, C. Paar, D. Westhoff (Eds.), Security in Ad-Hoc and Sensor Networks, in: Lecture Notes in Computer Science, Vol. 3313, Springer Berlin Heidelberg, 2005, pp. 152–165. http://dx.doi.org/10.1007/978-3-540-30496-8.13.

[99] C. Jin, X. Wang, H. Tan, Dynamic Attack Tree and Its Applications on Trojan Horse Detection, in: Proc. of the Second International Conference on Multimedia and Information Technology (MMIT 2010), Vol. 1, 2010, pp. 56–59. http://dx.doi.org/10.1109/MMIT.2010.12.

[100] R. Dewri, I. Ray, N. Poolsappasit, D. Whitley, Optimal security hardening on attack tree models of networks: A cost-benefit analysis, Int. J. Inf. Secur. 11 (3) (2012) 167–188. http://dx.doi.org/10.1007/s10207-012-0160-y.

[101] D. Grochocki, J. Huh, R. Berthier, R. Bobba, W. Sanders, A. Cardenas, J. Jetcheva, AMI threats, intrusion detection requirements and deployment recommendations, in: Proc. of the 3rd International Conference on Smart Grid Communications (SmartGridComm 2012), 2012, pp. 395–400. http://dx.doi.org/10.1109/SmartGridComm.2012.6486016.

[102] S. Bistarelli, M. DallAglio, P. Peretti, Strategic games on defense trees, in: T. Dimitrakos, F. Martinelli, P. Ryan, S. Schneider (Eds.), Formal Aspects in Security and Trust, in: Lecture Notes in Computer Science, Vol. 4691, Springer Berlin Heidelberg, 2007, pp. 1–15. http://dx.doi.org/10.1007/978-3-540-75227-1.1.

[103] A. Bagnato, B. Kordy, P. Meland, P. Schweitzer, Attribute decoration of attack-defense trees, Int. J. Secur. Softw. Eng. 3 (2012) 1–35. http://dx.doi.org/10.4018/jsse.2012040101.

[104] A. Roy, D. Kim, K. Trivedi, Attack countermeasure trees (ACT): Towards unifying the constructs of attack and defense trees, Security and Communication Networks 5 (2012) 929–943. http://dx.doi.org/10.1002/sec.299.

[105] N. Ghosh, S. Ghosh, An approach for security assessment of network configurations using attack graph, in: Proc. of the 1st International Conference on Networks and Communications, (NETCOM 2009), 2009, pp. 283–288. http://dx.doi.org/10.1109/NetCoM.2009.83.

[106] R. Lippmann, K. Ingols, C. Scott, K. Piwowarski, K. Kratkiewicz, M. Artz, R. Cunningham, Evaluating and strengthening enterprise network security using attack graphs, technical report, MIT Lincoln Lab, 2005.

[107] R. Lippmann, K. Ingols, C. Scott, K. Piwowarski, K. Kratkiewicz, M. Artz, R. Cunningham, Validating and restoring defense in depth using attack graphs, in: Proc. of IEEE Military Communications Conference (MILCOM 2006), 2006, pp. 1–10. http://dx.doi.org/10.1109/MILCOM.2006.302434.

[108] M. Dacier, Y. Deswarte, M. Kaaniche, in: Models and Tools for Quantitative Assessment of Operational Security, Technical Report LAAS Re- serach Report 96493, LAAS-CNRS, 1996. http://homepages.laas.fr/deswarte/Publications/IFIP-SEC-96.pdf.

[109] R. Ortalo, Y. Deswarte, M. Kaaniche, Experimenting with quantitative evaluation tools for monitoring operational security, IEEE Transactions on Software Engineering 25 (1999) 633–650. http://dx.doi.org/10.1109/32.815323.

[110] M. Albanese, S. Jajodia, S. Noel, Time-efficient and cost-effective network hardening using attack graphs, in: Proc. of the 42nd Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN 2012), 2012, pp. 1–12. http://dx.doi.org/10.1109/DSN.2012.6263942.

[111] S. Noel, S. Jajodia, L. Wang, A. Singhal, Measuring security risk of networks using attack graphs, International Journal of Next-Generation Computing 1 (2010) 135–147.

[112] X. Ou, A. Singhal, Security risk analysis of enterprise networks using attack graphs, in: Quantitative Security Risk Assessment of Enterprise Networks, SpringerBriefs in Computer Science, Springer New York, 2011, pp. 13–23. http://dx.doi.org/10.1007/978-1-4614-1860-3.4.

[113] O. Sheyner, J. Wing, Tools for generating and analyzing attack graphs, in: M.M. Bonsangue, S. Graf, W.-P.d. Roever (Eds.), Formal Methods for Components and Objects, Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2004, pp. 344–371.

[114] S. Jha, O. Sheyner, J. Wing, in: Minimization and Reliability Analyses of Attack Graphs,Technical Report CMU-CS-02-109, School of Computer Science, Carnegie Mellon University, 2002.

[115] P. Ammann, J. Pamula, R. Ritchey, J. Street, A host-based approach to network attack chaining analysis, in: Proc. of the 21st Annual Computer Security Applications Conference (ACSAC 2005), 2005, pp. 72–84. http://dx.doi.org/10.1109/CSAC.2005.6.

[116] P. Ning, Y. Cui, D. Reeves, Constructing attack scenarios through correlation of intrusion alerts, in: Proc. of the 9th ACM Conference on Computer and Communications Security (CCS 2002), in: CCS '02, ACM, New York, NY, USA, 2002, pp. 245–254. http://dx.doi.org/10.1145/586110.586144.

[117] P. Ning, X. Xu, Learning attack strategies from intrusion alerts, in: Proc. of the 10th ACM Conference on Computer and Communications Security (CCS 2003), CCS '03, ACM, New York, NY, USA, 2003, pp. 200–209. http://dx.doi.org/10.1145/948109.948137. http://doi.acm.org.ezproxy1.lib.asu.edu/10.1145/948109.948137.

[118] P. Bhattacharya, S. Ghosh, Analytical framework for measuring network security using exploit dependency graph, Information Security, IET 6 (2012) 264–270. http://dx.doi.org/10.1049/iet-ifs.2011.0103.

[119] G. Modelo-Howard, S. Bagchi, G. Lebanon, Determining placement of intrusion detectors for a distributed application through bayesian network modeling, in: R. Lippmann, E. Kirda, A. Trachtenberg (Eds.), Recent Advances in Intrusion Detection, Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2008, pp. 271–290.

[120] N. Poolsappasit, R. Dewri, I. Ray, Dynamic security risk management using bayesian attack graphs, IEEE Transactions on Dependable and Secure Computing 9 (2012) 61–74. http://dx.doi.org/10.1109/TDSC.2011.34.

[121] P. Xie, J. Li, X. Ou, P. Liu, R. Levy, Using Bayesian networks for cyber security analysis, in: Proc. of IEEE/IFIP International Conference on Dependable Systems and Networks (DSN 2010), 2010, pp. 211–220. http://dx.doi.org/10.1109/DSN.2010.5544924.

[122] L. Wang, T. Islam, T. Long, A. Singhal, S. Jajodia, An attack graph-based probabilistic security metric, in: V. Atluri (Ed.), Data and Applications Security XXII, in: Lecture Notes in Computer Science, Vol. 5094, Springer Berlin / Heidelberg, 2008, pp. 283–296.

[123] L. Wang, S. Noel, S. Jajodia, Minimum-cost network hardening using attack graphs, Computer Communications 29 (18) (2006) 3812–3824. http://dx.doi.org/10.1016/j.comcom.2006.06.018.

[124] L. Wang, A. Singhal, S. Jajodia, Measuring the overall security of network configurations using attack graphs, in: S. Barker, G.-J. Ahn (Eds.), Data and Applications Security XXI, Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2007, pp. 98–112.

[125] L. Wang, A. Singhal, S. Jajodia, Toward measuring network security using attack graphs, in: Proc. of the 2007 ACM Workshop on Quality of Protection (QoP 2007), QoP '07, ACM, New York, NY, USA, 2007, pp. 49–54. http://dx.doi.org/10.1145/1314257.1314273.

[126] L. Wang, S. Jajodia, An approach to preventing, correlating, and predic, in: Intrusion Detection Systems, in: Advances in Information Security, Springer US, 2008, pp. 93–128.

[127] D. Patsos, S. Mitropoulos, C. Douligeris, Expanding topological vulnerability analysis to intrusion detection through the incident response intelligence system, Information Management & Computer Security 18 (2010) 291–309. http://dx.doi.org/10.1108/09685221011079207.

[128] Z. Zhang, S. Wang, Boosting logical attack graph for efficient security control, in: Proc. of the 7th International Conference on Availability, Reliability and Security (ARES 2012), 2012, pp. 218–223. http://dx.doi.org/10.1109/ARES.2012.72.

[129] H. Almohri, in: Security Risk Prioritization for Logical Attack Graphs, Ph.D. thesis, Kansas State University, 2008.

[130] B. Chen, Z. Kalbarczyk, D. Nicol, W. Sanders, R. Tan, W. Temple, N. Tippenhauer, A. Vu, D. Yau, Go with the flow: Toward workflow-oriented security assessment, in: Proc. of the 2013 Workshop on New Security Paradigms Workshop (NSPW 2013), NSPW '13, ACM, New York, NY, USA, 2013, pp. 65–76. http://dx.doi.org/10.1145/2535813.2535821.

[131] SINTEF, SeaMonster security modelling software. URL: https://sourceforge.net/p/seamonster/wiki/Home/. (last accessed: 11 June 2017).

[132] Isograph Ltd, AttackTree+. URL: http://www.isograph-software.com/2011/. (last accessed: 11 June 2017).

[133] Amenaza Technologies Ltd., SecurITree. URL: http://www.amenaza.com/. (last accessed: 11 June 2017).

[134] A. T. Ltd., RedSeal Networks. URL: http://www.redseal.net. (last accessed: 11 June 2017).

[135] S. S. Inc, Skybox. URL: http://www.skyboxsecurity.com. (last accessed: 11 June 2017).

[136] I. CyVision Technologies, Cauldron. URL: https://www.cyvisiontechnologies.com/. (last accessed: 11 June 2017).

[137] F. Jia, J. Hong, D. Kim, Towards Automated Generation and Visualization of Hierarchical Attack Representation Models, in: Proc. of the IEEE International Conference on Dependable, Autonomic and Secure Computing (DASC 2015), 2015, pp. 1689–1696. http://dx.doi.org/10.1109/CIT/IUCC/DASC/PICOM.2015.255.

[138] L. Firemon, Firemon. URL: https://www.firemon.com/. (last accessed: 11 June 2017).

[139] S. Noel, E. Harley, K. Tam, M. Limiero, M. Share, CyGraph: Graph-based Analytics and Visualization for Cybersecurity, Handbook of Statistics 35 (2016) 117–167. http://dx.doi.org/10.1016/bs.host.2016.07.001. Cognitive Computing: Theory and Applications. http://www.sciencedirect.com/science/article/pii/S0169716116300426.

[140] O. Gadyatskaya, R. Jhawar, P. Kordy, K. Lounis, S. Mauw, R. Trujillo-Rasua, Attack trees for Practical Security Assessment: Ranking of Attack Scenarios with ADTool 2.0, in: Proc. of the 13th International Conference on Quantitative Evaluation of Systems (QEST 2016), Springer International Publishing, Cham, 2016, pp. 159–162. http://dx.doi.org/10.1007/978-3-319-43425-4.10.

[141] C. Probst, J. Willemson, W. Pieters, in: The Attack Navigator, Springer International Publishing, Cham, 2016, pp. 1–17. http://dx.doi.org/10.1007/978-3-319-29968-6.1.

[142] B. Foo, Y.-S. Wu, Y.-C. Mao, S. Bagchi, E. Spafford, ADEPTS: Adaptive Intrusion Response using Attack Graphs in an E-commerce Environment, in: Proc. of the IEEE/IFIP International Conference on Dependable Systems and Networks (DSN 2005), 2005, pp. 508–517. http://dx.doi.org/10.1109/DSN.2005.17.

[143] Y. Wu, B. Foo, Y. Mao, S. Bagchi, E. Spafford, Automated adaptive intrusion containment in systems of interacting services, Computer Networks 51 (2007) 1334–1360. http://dx.doi.org/10.1016/j.comnet.2006.09.006.

[144] S. Ardi, D. Byers, N. Shahmehri, Towards a Structured Unified Process for Software Security, in: Proc. of International Workshop on Software Engineering for Secure Systems (SESS 2006), ACM, New York, NY, USA, 2006, pp. 3–10. http://dx.doi.org/10.1145/1137627.1137630. http://doi.acm.org.ezproxy1.lib.asu.edu/10.1145/1137627.1137630.

[145] D. Byers, S. Ardi, N. Shahmehri, C. Duma, Modeling Software VulnerabilitiesWith Vulnerability Cause Graphs, in: Proc. of the 22nd IEEE International Conference on Software Maintenance (ICSM 2006), 2006, pp. 411–422. http://dx.doi.org/10.1109/ICSM.2006.40.

[146] C. Ten, C. Liu, M. Govindarasu, Vulnerability Assessment of Cybersecurity for SCADA Systems Using attack trees, in: Proc. of IEEE Power Engineering Society General Meeting, 2007, pp. 1–8. http://dx.doi.org/10.1109/PES.2007.385876.

[147] L. Meshkat, L. Xing, S. Donohue, Y. Ou, An overview of the phase- modular fault tree approach to phased mission system analysis, (2003).

[148] T. Schaberreiter, C. Wieser, I. Sanchez, J. Riekki, J. Roning, An Enumeration of RFID Related Threats, in: Proc. of the 2nd International Conference on Mobile Ubiquitous Computing, Systems, Services and Technologies (UBICOMM 2008), 2008, pp. 381–389. http://dx.doi.org/10.1109/UBICOMM.2008.32.

[149] G. Louthan, P. Hardwicke, P. Hawrylak, J. Hale, Toward Hybrid Attack Dependency Graphs, in: Proc. of the 7th Annual Workshop on Cyber Security and Information Intelligence Research (CSIIRW 2011), CSIIRW '11, ACM, New York, NY, USA, 2011, pp. 1–4. http://dx.doi.org/10.1145/2179298.2179368. http://doi.acm.org.ezproxy1.lib.asu.edu/10.1145/2179298.2179368.

[150] G. Louthan, Hybrid Attack Graphs for Modeling Cyber-physical Systems, University of Tulsa, 2011. https://bitbucket.org/duplico/tuthesis/src/98897526892a4f219efc2cc8f4c512282b801696/pres/louthan_ms_defense.pdf.

[151] X. Dong, S. Jauhar, W. Temple, B. Chen, Z. Kalbarczyk, W. Sanders, N. Tippenhauer, D. Nicol, in: The Right Tool for the Job: A Case for Common Input Scenarios for Security Assessment, Springer International Publishing, Cham, 2016, pp. 39–61. http://dx.doi.org/10.1007/978-3-319-46263-9.3.

[152] J. Beale, R. Deraison, H. Meer, R. Temmingh, C. Walt, The NESSUS project, Syngress Publishing (2002). http://www.nessus.org.