

An Evolutionary Approach of Attack Graph to Attack Tree Conversion

Md. Shariful Haque

Department of Computer Science, University of Alabama, Tuscaloosa, AL 35478, USA
E-mail: mshaque@crimson.ua.edu

Travis Atkison

Department of Computer Science, University of Alabama, Tuscaloosa, AL 35478, USA
E-mail: atkison@cs.ua.edu

Received: 04 July 2017; Accepted: 07 September 2017; Published: 08 November 2017

Abstract—The advancement of modern day computing has led to an increase of threats and intrusions. As a result, advanced security measurements and threat analysis models are necessary to detect these threats and identify protective measures needed to secure a system. Attack graphs and attack trees are the most popular form of attack modeling today. While both of these approaches represent the possible attack steps followed by an attacker, attack trees are architecturally more rigorous than attack graphs and provide more insights regarding attack scenarios. The goal of this research is to identify the possible direction to construct attack trees from attack graphs analyzing a large volume of data, alerts or logs generated through different intrusion detection systems or network configurations. This literature summarizes the different approaches through an extensive survey of the relevant papers and identifies the current challenges, requirements and limitations of an efficient attack modeling approach with attack graphs and attack trees. A discussion of the current state of the art is presented in the later part of the paper, followed by the future direction of research.

Index Terms—Attack graph, Attack tree, Intrusion detection, Attack modeling, Survey.

I. INTRODUCTION

Computer technology has become ubiquitous with the increasing trend of computational capability into devices used in our everyday life. The dependency of these devices on the network services and applications marks network security as a demanding research domain. Software bugs, security policy errors, or an inefficient network configuration can cause security violations any time in a system. A person with a malicious intent can make attempts to gain unauthorized access using these vulnerabilities. These attempts are termed an “attack” in computer security, and are defined by IETF as “an intentional act by which an entity attempts to evade security services and violate the security policy of a system” [1]. To guard against attacks, a system must be

diagnosed and assessed for risks; then, possible countermeasures must be suggested. Attack representation models (ARMs) are the most effective means of analysis in these scenarios. Attack graphs and attack trees are the most popular forms of representation models.

A. Attack Representation Model

Intrusion detection systems, or other security components like firewalls, generate security alerts if any vulnerable activities are observed by these systems. Alerts generated by these sources are isolated. Efficient detection of any attack scenario requires correlating these isolated alerts. Attack representation modeling is the process of identifying the relations between system alerts and developing an attack scenario recognition system. The purpose of the attack representation model is to determine the path of an attack and generate reports accordingly. Cheung et al. identified the necessary steps to develop a model which can recognize cyber attack scenarios successfully [2]. In Figure 1, these steps are sequentially organized to describe the ARM.

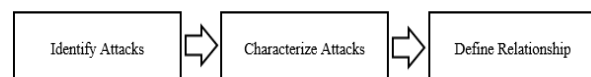


Fig.1. Attack Representation Model.

The initial step of representation modeling requires identifying the attacks and dividing them into attack subgoals until each of the logical attacks are identified by the detection system. In the next phase of the modeling, these attacks need to be attributed based on the observed events, system states and interfaces. In the final stage, the relationship among these attacks needs to be developed based on the temporal relationship (the sequence of identified attacks), attribute-value relationship (attacks might be generated from similar sources) and prerequisite relationship (one attack triggers another attack) [2].

B. Attack Graph

Attack graphs represent a detailed view of system

security by determining if an attacker can reach the final goal state(s) by penetrating the security holes of the system from an initial state. These graphs are composed of nodes and edges where the representation of these components changes with the definition of a particular attack graph. Typically, nodes in an attack graph represent states, and the edges refer to the transition of the different states defined through various post- and pre-conditions. Sheyner et al. proposed an attack graph using the similar notion [3].

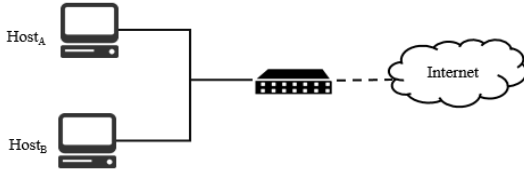


Fig.2. Example Network.

- 1) *Definition:* An attack graph is a tuple $G = \{S, \tau, S_0, S_S\}$ where $\tau \subseteq S \times S$ refers to the transition relation between states and $S, S_0 \subseteq S$ and $S_S \subseteq S$ refer to sets of states, initial states and success states, respectively. [3]
- 2) *Example Scenario:* For a better understanding of an attack graph model, let's consider the example network shown in Figure 2. *Host_A* and *Host_B* are two workstations connected to the Internet through a switch. *Host_B* is owned by a malicious user who wants to gain access to *Host_A*. Because *Host_B* knows the network address of *Host_A*, he can either use the remote login feature that is used by trusted users or deploy a *.rhost* file utilizing an FTP vulnerability. In both scenario, the attacker can create a trusted relationship with the target machine, and easily exploit the buffer overflow to gain root access. In this case, the intruder can deploy the required binary codes or create the file locally.

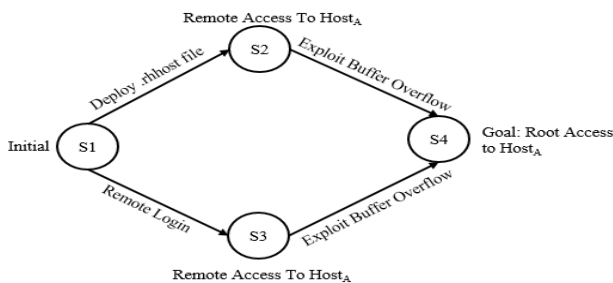


Fig.3. Attack Graph based on Example Network

Figure 3 shows a simple attack graph that is generated based on the scenario described above. Here S1 denotes the initial state and S4 denotes the goal of the attacker, i.e. gaining root access to *Host_A*. S2 and S3 are intermediary states where the attacker gets remote access to *Host_A* by deploying the *.rhost* file and performing the remote login operation, respectively. Edges in this figure describe the transition from one state to the next state through actions from the attacker.

C. Attack Tree

The concept of an attack tree (threat logic tree) was first introduced in the context of information systems by Weiss, but was not widely accepted as a part of security until it was popularized by Schneier at the end of the 1990s [4]. Attack trees are a powerful approach of modeling the security vulnerabilities of information systems. They analyze different security threats, identify different paths to achieve the goal, and build a structure that describes how a threat helps malicious users reach their goal. This structure is organized as a tree, where the elementary attacks are placed at the leaf level and the primary attack is placed at the root.

The internal (non-leaf) nodes in the tree represent a combined attack of the elementary nodes or non-leaf nodes located in the next higher level in the tree. In the top-down approach, the internal nodes are actually considered as a refinement of the higher-level nodes. This refinement can be either conjunctive (aggregation or "AND" node) or disjunctive (choice or "OR" node). In conjunctive refinement, all the immediate child nodes will need to be in action to achieve the goal. However, in disjunctive refinement, any attack will be sufficient to fulfill the goal [5].

- 1) *Definition:* An attack tree is a 3-tuple (N, n_0, \rightarrow) , where N refers to a set of nodes, $n_0 \in N$ refers to the root node, and \rightarrow is an acyclic relation of type $\rightarrow \subseteq N \times M^+(N)$ where $M^+(N)$ denotes a multi-set of set N , such that every node in N can be reachable from n_0 [5].
- 2) *Example Scenario:* Let's consider the similar scenario depicted in Figure 2 to describe the attack tree. In this case, since the goal of the malicious user is to gain root access to *Host_A*, this will be the root of the attack tree. In order to get root access to *Host_A*, the attacker first needs to get remote access to the system; therefore, this will be in the next level of the attack tree. The final level of the attack tree will consist of actions like deploying *.rhost* and the remote login operation. As either of these actions allows a user to get remote access, these two nodes form a disjunctive connection with their parent node. Figure 4 shows the attack tree created for this example.

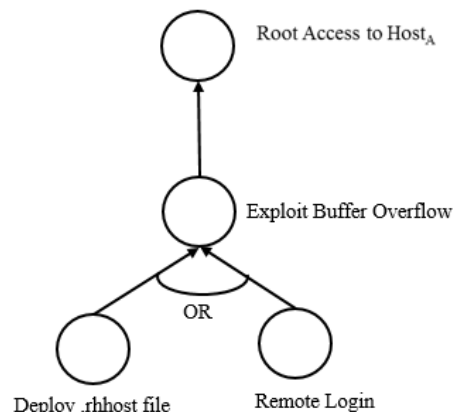


Fig.4. Attack Tree based on Example Network.

II. ANALYSIS OF CORE PAPERS

The primary goal of this research is to identify an efficient approach for attack tree construction. The process requires correlating the alerts generated by various Intrusion Detection Systems (IDS) and network configuration, then analyzing and transforming the data accordingly. This section contains an analysis of five papers related to alert correlation, attack graph generation and attack tree reduction.

A. Statistical Causality Analysis of INFOSEC Alert Data [6]

- 1) *Key Motivation:* As the number of intrusion detection systems increases, the volume of generated alerts becomes too numerous for a system administrator to analyze the alerts and respond to any true attack in time. Therefore, it is necessary to devise an efficient alert correlation technique that will reduce the number of alerts by identifying similar alert characteristics coming from different sources. The correlation system should also be able to define the relationship between those alerts and detect a new form of attack. Previous alert correlation systems depended on prior knowledge and consequences of alerts [2, 7, 8], and lacked the ability to detect a new attack. In [6], Qin et al. used a clustering technique to generate a high-level aggregated alert, and causal analysis to discover new relationships from the attacks. This approach does not depend on the previous knowledge for pattern matching and is therefore capable of discovering new attacks.

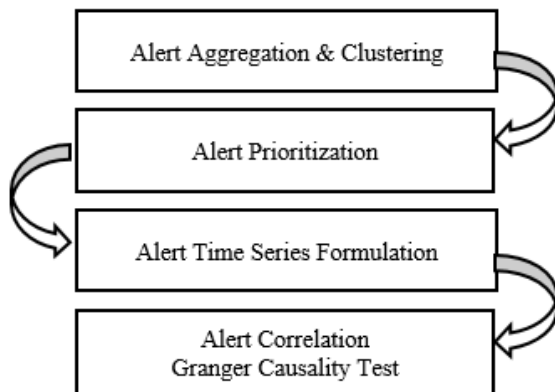


Fig.5. Attack Scenario Construction Process.

- 2) *Core Contribution:* As a part of this research, Qin et al. used time series and statistical analysis to combine the low-level alerts using their attribute information, thus ensuring a reduction of the high volume of alerts. Clustering was used to change the low-level alerts into aggregated alerts. Then, the alerts were prioritized depending on the relationship to the networks, hosts and goals of the attacks. Finally, attack scenarios were constructed based on the correlated alerts generated through causality analysis. Figure 5 illustrates the attack correlation process developed by Qin.

a) *Alert Aggregation and Clustering:* The four-step attack scenario generation process starts with alert fusion and reduction of the large volume of alerts. Based on the Intrusion Detection Message Exchange Format (IDMEF) standard, each alert is attributed with a timestamp, user name, process name, attack class and sensor ID, source and destination IP, and port [9]. In the alert aggregation step, alerts with overlapping attribute values are first combined considering a negligible difference in the timestamp field using the multivariate matching algorithm. Later, alerts are aggregated based on the attribute data and the source of the alerts [7].

Aggregated alerts are further grouped using different clustering algorithms, such as conceptual clustering, based on similar characteristics of the alerts [10]. In this case, alerts with the same attributes fall into the same cluster. This step reduces the number of redundant alerts. Next, hyper-alerts are generated which contain the same attributes, but different timestamps.

b) *Alert Prioritization:* Alert prioritization is a step of assigning each hyper-alert a rank for more analysis and correlation. Two attributes are considered in computing the alert priority - how pertinent is the alert to the configuration of a secure network and host, and the rank of severity assessed by the analysts. This approach was used by Porras et al. in their mission-impact-based model using a correlation engine called M-Correlator [8].

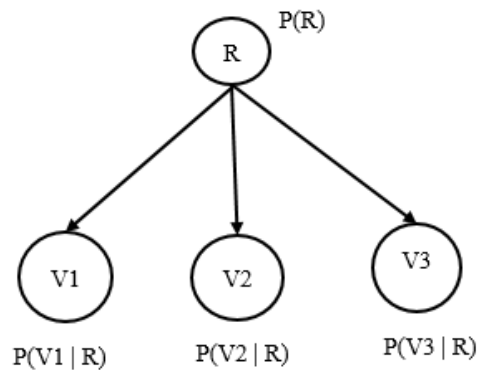


Fig.6. Sample Bayesian Network.

Priority ranking is assigned to the alerts with a priority computation model based on a Bayesian Network. A Bayesian network is a directed acyclic graph where nodes represent a variable with multiple states, and edges represent the dependency between the variables, as shown in Figure 6. Each child node has an associated Conditional Probability Table (CPT) where entries have the following format, shown in Equation 1, illustrated based on the child node *V1* given in Figure 6.

$$CPT_{ij} = P(V1 = j | R = i) \quad (1)$$

In this priority computation model, Bayesian inference is used to calculate a belief over a particular state of a variable. This computation produces results in the range [0,1]. The priority of a hyper-alert is calculated comparing the dependencies associated with an attack with respect to the configuration of the network. A pre-

built knowledge base with the entry of hyper-alerts associated with host configurations is used for this comparison. The comparison process performs another set of filtering on the alerts.

c) *Alert Time Series Formulation*: In this step, a series of equally distant time slots is designed to accommodate the already processed hyper-alerts and formulate a hyper-alert time series variable. If R is considered as a time range with equal time intervals T , then the number of total time slots is, $S = R/T$. For a cluster A with the following formation:

$$A = \{A_1, A_2, A_3, \dots, A_N\}t(m) \quad (2)$$

$m = \{0, 1, 2, \dots, k\}$ is the set of alerts that falls in the m^{th} time slot.

d) *Alert Correlation*: In this phase of the alert scenario generation step, the Granger Causality Test (GCT) algorithm is used for pairwise correlation between alerts [11]. If there are two time series variable x and y , then GCT can be applied to these two variables to see if x contains necessary statistical information about y . Variable x is said to be Granger-causes y , if a relationship is found between the two. The GCT algorithm is based on the following fact: if B occurs as an effect of A , then event A should always happen before event B . Therefore, A is considered a cause-event and B an effect-event.

In GCT, the variable b against the effect-event B is modeled against the Autoregressive Model (AR model) and the Autoregressive Moving Average Model (ARMA model). AR model generates the value of variable b based on its k previous values; however, ARMA model predicts the current value of b based on the past k values of both b and a , variables associated with cause-event A . Finally, Granger Causality Index (GCI), g , is calculated based on the residuals of these two models.

With GCT, a relationship between a set of hyper-alerts is basically determined based on a particular hyper-alert A . The GCI value of each pair is then stored and compared to generate a list of increasingly ordered values with the GCI value. Then, a list of candidate alerts is marked as causally related to alert A .

The correlation step also includes identification and elimination of background alerts. The Ljung-Box test is used in this method to identify background alerts [12]. Background alerts usually contain random attributes that help to identify the particular type of alert [13].

3) *Limitation*: Qin's statistical approach of identifying correlated alerts can be a vital part of an attack graph generation process. It can identify a new form of attack without any prior knowledge about the nature of the attack. However, this approach lacks the ability to generate scenarios without the intervention of a system analyst. CPTs in the attack prioritization phase are neither adaptive nor updated according to the mission goals. Instead, they are developed based on prior experience and domain knowledge. Qin's approach also suffers from false causality alert scenarios when the volume of

background alerts is large. Also, the Ljung-Box test cannot completely filter the background alerts, and advanced knowledge is required to inspect the alert candidates produced from GCT.

B. A Scalable Approach to Attack Graph Generation [14]

1) *Key Motivations*: Though research on attack graphs has been running for decades, most of the works suffer from proper scalability and lack of logical formulation. Philips et al. developed a state-based attack graph representation model in 1998 which experienced an exponential problem in analyzing attack scenarios. They applied partial-order reduction to remove duplicate attack paths, but did not indicate any clear performance efficiency [15]. The model-checking based approach for attack graphs by Sheyner et al. also suffers from this same issue [14]. Therefore, Ou et al. proposed a logical attack graph. It illustrated the way to produce a derivation trace and generate an attack graph using the trace in quadratic time.

The customization of input information and the resulting graph data structures is another problem with most of attack graph tools. These tools require additional input in a specialized data format and often produce complex, unclear attack graphs. Ou's proposed logical attack graph clearly specifies the configuration information of the system and potential privileges of the attacker. Finally, Ou's approach also answers the reasoning of attack scenarios rather than simply showing the attack steps in an attack graph.

2) *Core Contribution*: Although it encounters the state explosion problem, the logic-based approach proposed by Sheyner et al. is superior to the ad-hoc attack graphs from two points of view. It is more efficient than customized algorithms for a complex scenario, and it also enables further analysis of the graph data structure. Sheyner's approach built on a reasoning system called MulVAL [16]. The proposed approach of Ou et al. is designed based on the same system [14].

a) *MulVAL (Multihost, multistage Vulnerability Analysis)*: MulVal is a framework for designing the interactive relationship between different software vulnerabilities with the configuration of a system and network. It basically determines the impact of various software bugs on a network. This sort of system usually demands two features: automatic integration of vulnerability specifications generated by the bug-reporting community, and scalable analysis irrespective of the size of the network. MulVAL was designed to fulfill both of these features. It consists of a scanner that is run asynchronously in each host, and an analyzer which runs whenever the scanner captures new information. The MulVAL reasoning engine uses XSB to analyze interaction rules based on input data. XSB is

basically a model that provides proper semantics for logic programs [17].

```
execCode(Attcker, Host, User) :-
    networkService(Host, Program,
        Protocol, Port, User)
    vulExists(Host, VulID, Program,
        remoteExploit, privEscalation)
    netAccess(Attacker, Host,
        Protocol, Port)
```

Fig.7. Datalog Interaction Rule.

b) *Interaction Rule*: MulVAL uses Datalog facts to represent the configuration information. Inputs for MulVAL analysis include advisories, system and network configuration, information related to user and policy, and interaction records. The following is an example scenario represented in Datalog.

The code snippet in Figure 7 represents a rule similar to a vulnerable scenario where the attacker wants to achieve a higher privilege by remotely exploiting a vulnerability of a host. A *Program* running in the *Host* contains the vulnerability with the privilege *User* and scanning the *Port* and *Protocol*. To exploit the service for upgrading privilege, the attacker first accesses the network and executes a random code as *User*. The first line of the Datalog rule contains the conclusion while the rest of the lines are the conditions needed to meet the conclusion. Capitalized words represent Datalog variables. Predicates are divided into multiple categories: *primitive* predicates refer to configuration information recorded by the host and network scanners, and *derived* predicates are generated by successively applying interaction rules on the primitive predicates.

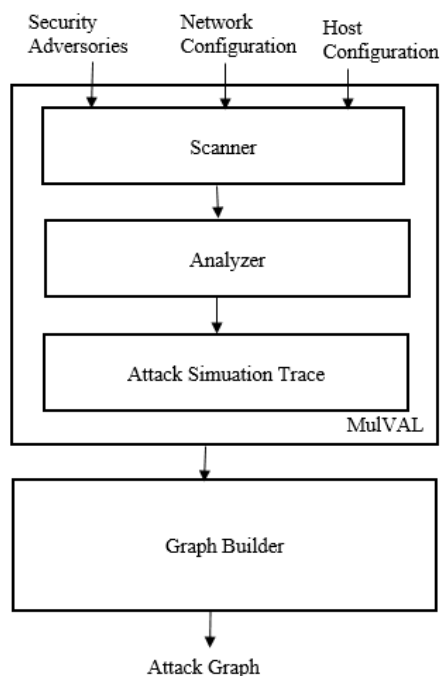


Fig.8. Logical Attack Graph Generator.

c) *Logical Attack Graph*: Ou et al. modified the MulVAL reasoning engine to include an additional functionality - record the trace of the evaluation performed by XSB. These traces, also known as *attack simulation traces*, are passed to the *graph builder*. Figure 8 illustrates the steps of a logical attack graph generator.

The graph builder generates a directed graph with a tree-like structure. Two different types of nodes are used in the graph to represent the *derivation node* and *fact node*. *Fact* nodes also have two different forms: primitive and derived fact nodes. Every fact node has a predicate applied to its argument, and a derivation node has an interaction rule. Edges in the graph represent a dependency relationship between nodes. An edge from the fact node to a derivation node determines how a fact depends on an interaction rule, and an edge from a derivation node to the fact node means that the fact has satisfied the precondition rule of the derivation node. In this graph, the derivation nodes usually form a conjunction node that requires multiple facts to satisfy it; on the other hand, fact nodes form a disjunction node because there might be multiple ways to generate a fact.

```
execCode(Attcker, Host, User) :-
    networkService(Host, Program,
        Protocol, Port, User)
    vulExists(Host, VulID, Program,
        remoteExploit, privEscalation)
    netAccess(Attacker, Host,
        Protocol, Port)
    assert_trace(because(
        'remote exploit of a server program',
        execCode(Attcker, Host, User)
        [networkService(Host, Program,
            Protocol, Port, User)
        vulExists(Host, VulID, Program,
            remoteExploit, privEscalation)
        netAccess(Attacker, Host,
            Protocol, Port)]))
```

Fig.9. Attack Simulation Trace.

d) *Interaction Rule to Graph Transformation*: The graph builder builds a logical attack graph based on the *attack simulation trace*. The MulVAL reasoning engine is modified to record this trace by adding an additional subgoal as an output of the engine. In this scenario, the rule shown in Figure 7 is transformed into the form illustrated in Figure 9. The attack graph is computed from all the simulation traces that are produced by traversing all the possible derivation paths. Simulation traces are recorded in the form (*TraceStep*, *Fact*, *Conjunct*). In the attack graph, *TraceStep* forms the derivation node, *Fact* is designated as the parent, and *Conjunct* becomes the child node. The sample attack graph shown in Figure 10 is derived from Figure 7.

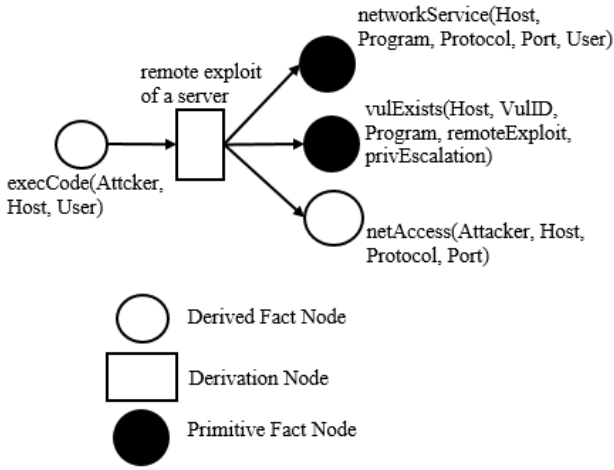


Fig. 10. Attack Graph.

3) *Limitation:* Ou’s proposed model shows a definite superiority over the other models in terms of efficiency. It can generate an attack graph polynomial to the network size. However, to create a complete attack graph, attack conditions should be expressed in propositional formulas - otherwise, this method overlooks that particular attack condition. Also, an attack graph generated in this way contains loops which restrict it from converting a graph to a tree. Ou discussed a possible solution to this problem, but how this solution affects the proposed algorithm is not properly explained.

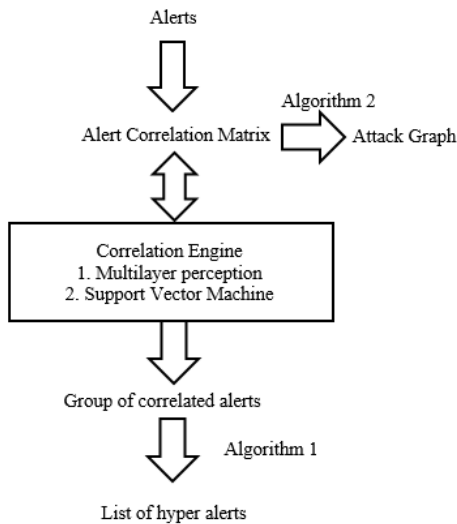


Fig. 11. Proposed Attack Correlation and Graph Generation Approach.

relationships. All these approaches lack features to identify relationships between alerts: some cannot identify causal relationships, some are applicable only in familiar situations, and some require predefined rules and expected consequences [19, 20, 21, 22, 23, 24]. Zhu et al. developed a correlation method that enables automatic extraction of an attack strategy from intrusion alerts without prior knowledge of the alerts. They used an Alert Correlation Matrix (ACM) to store the strength of any pair of alerts. ACM is initialized in the training phase, and then used for retrieving strategies.

2) *Core Contribution:* Intrusion or anomaly detection depends on detailed information of a system, and proper understanding about the anomalous behavior of the system that occurs due to different attacks. Building an exact profile of the general behavior of the system meets the first requirement in this case, while discovering attack strategy from IDS generated alerts fulfill the second criteria. The approach Zhu’s proposed method is developed based on the concept of a neural network. A knowledge base is first built through supervised learning and stores required relationship attributes between alerts like correlation strength and average time difference between two alerts. The correlation engine uses Multilayer Perception (MLP) and Support Vector Machine (SVM) to assign correlation probability to an alert. This correlation engine is used to generate hyper-alert graphs and attack graphs that represent a real attack scenario. Figure 11 depicts the general model of this proposed method.

	a1	a2	a3
a1	$c(a1,a1)$	$c(a1,a2)$	$c(a1,a3)$
a2	$c(a2,a1)$	$c(a2,a2)$	$c(a2,a3)$
a3	$c(a3,a1)$	$c(a3,a2)$	$c(a3,a3)$

Fig. 12. Attack correlation matrix.

C. Alert Correlation for Extracting Attack Strategies [18]

1) *Key Motivation:* Alert correlation is an integral part of designing an efficient intrusion detection and response system. Identifying the attack strategy and analyzing a large volume of alerts generated by the Intrusion Detection System (IDS) is the principal goal of alert correlation. There are different approaches of alert correlation based on feature similarity, known scenario, or cause and effect

a) *Alert Correlation Matrix:* The Alert Correlation Matrix (ACM) is an $n \times n$ matrix where each cell contains a correlation weight between two alerts from a set of n alerts $\{a_1, a_2, \dots, a_n\}$. The correlation weight also refers to the causal relationship between the alerts. An ACM with 3 alerts $\{a_1, a_2, a_3\}$ looks like the one shown in Figure 12. Each correlation is defined by $c(a_i, a_j)$ which refers to a relationship between two alerts a_i and a_j where a_i arrives before a_j and holds a weight of $W_{c(a_i, a_j)}$ calculated from N number of occurrences of these two

alerts with probability p . This structure and correlation weight of ACM also provide some knowledge such as correlation strength (Π), temporal and causal relationship. The correlation strength (Π) helps determine the correlation weight by calculating *Backward Correlation Strength* (Π^b) and predicting intrusion. It also helps recognize an attack by measuring *Forward Correlation Strength* (Π^f). The temporal relationship refers to the sequence of the occurrence of the alert in a cell, while the causal relationship reveals the actual relation. ACM gets updated through the correlation engine over the time when any alert is generated.

b) *Feature Selection*: Zhu followed an approach proposed by Valdes et al. while selecting features and computing probabilities based on these to find the actual correlation weight between alerts [7]. The 6 features included in the methods are: the similarity between the source IP address, target IP address, the similarity in port numbers, the similarity between source and target IP address of two successive alerts, backward correlation strength of two alerts, and frequency of the correlation between the alerts.

c) *Alert Correlation*: The alert correlation technique in Zhu's method uses both Multilayer Perception (MLP) and Support Vector Machine (SVM) to determine the relationship between any two alerts and the probability of correlation. MLP uses an error-correction rule, and SVM is developed based on the concept of risk minimization principle. MLP usually suffers from the over-fitting problem while SVM might not produce accurate output all the time. Therefore, this approach follows a combination of both these methods to determine the relationship.

When the relationship between a set of alerts is defined, hyper-alert graphs are generated from the output of the correlation. Hyper-alert graphs generated this way help in identifying multiple goals of an attacker. *Correlation threshold* and *correlation sensitivity* are used to construct the graph. The first one determines the probability of the relationship between alerts, and the second one determines to what extent a new alert can join a hyper-alert from a set of hyper-alerts. In this correlation method, the ACM is continuously updated so that the latest correlation strength can be applied to the future correlation process.

d) *Attack Graph*: Based on the training data and output of the correlation algorithm, attack graphs are generated. The attack graph eventually helps network administrators predict the probable attack while observing the pattern of the received alerts. Attack graph generation starts with an alert representing a particular attack. This process iteratively scans the ACM horizontally to identify the alerts likely to appear next considering one alert as a reference point and run until no other alert is found in the matrix.

There are considerable differences between the hyper-alert graph and the attack graph in Zhu's proposed method. Hyper-alerts can be considered as an instance of an attack graph. Usually, hyper-alert graphs do not contain any cycle, while attack graphs may have a cycle

if the same hyper-alert appears multiple times.

3) *Limitations*: Zhu's method proposed a different technique for alert correlation as it introduced Support-Vector Machine (SVM) and used a combination of both Multilayer Perception (MLP) and SVM to determine the relationship between alerts. However, like other proposed methods, it has also some limitations. As this method follows a supervised learning approach, both MLP and SVM need manually generated and labeled training. This methodology requires additional effort and might produce an erroneous result. Also, the attack graph contains loops, and no approaches to eliminate the loops are discussed.

D. Scalable Attack Representation Model using Logic Reduction Techniques [25]

1) *Key Motivations*: Attack graphs and Attack trees are the most popular attack modeling techniques. Since the inception of the concept of presenting attacks through graph structure, various models were proposed in the form of attack graphs or attack trees. These models include automatic construction of attack scenarios in both forms, but none of these approaches are proved to be efficient [3, 4, 26]. Analyzing an attack graph for a seemingly larger network suffers from the state-explosion problem while an attack tree lacks the feature for covering all the attack scenarios [14, 25]. Efforts to generate attack graphs and then transform them into a tree structure also fails due to lack of scalability as the methods generate nodes at an exponential rate. Therefore, Hong et al. introduce two logic reduction techniques to enable an automatic transformation of an attack tree as well as ensure reduction of the size of the tree.

2) *Core Contributions*: The attack representation model goes through four different phases to get a complete shape. Its life cycle start with generation of the model. It, then, gets a visualized form through textual or graphical representation. Next, this representation model is analyzed against the particular network in which it is designed. Finally, the modification phase reconstructs the model based on the changes that occurred in the network system. The logic reduction techniques Hong proposed are devised considering the reconstruction phase of the attack tree. The automatic construction approaches of the attack trees maintain the integrity of the original tree while reducing the size. In the Full Path Calculation (FPC) approach, similar nodes are configured as a single group. On the other hand, the Incremental Path Calculation (IPC) approach minimizes repetition of the node through recursive expansion of the attack path. The approaches are described based on an example scenario shown in Figure 2 and used throughout this section. In this network, the nodes are labeled alphabetically with

the attacker host as A and the target host (remote access to host) as E. The intermediary attack steps, such as deploy .rhhost file, remote login and exploit buffer overflow, are vulnerabilities in different hosts labeled as B, C and D. This example attack scenario can be expressed as an attack tree with conjunctive and disjunctive nodes as illustrated in Figure 13.

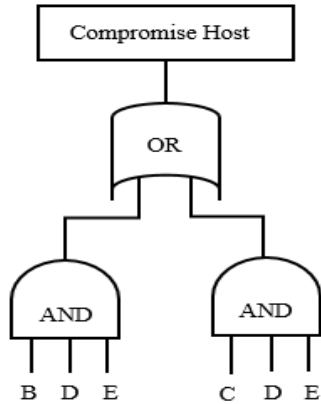


Fig.13. Full Attack Tree Generated From Example Network.

a) *Full Path Calculation (FPC)*: Logic reduction using full attack path initially requires the full attack tree to be represented in a logical expression. The reduction process eliminates the attack sequence and groups similar nodes together. In this case, an element from the attack tree is selected first, then the selected node is factorized from the complete logical expression. This process iteratively runs by selecting a common element from the rest of the nodes of the graph. Paths with similar nodes are evaluated as a container of similar information in this approach. The logical expression of the full attack path is shown in Equation 3 and in reduced form using the FPC as shown in Equation 4.

$$BDE + CDE \tag{3}$$

$$BDE + CDE = DE(B+C) = E(D(B+C)) \tag{4}$$

The simplified attack tree considers AND and OR nodes with two inputs as illustrated in Figure 14.

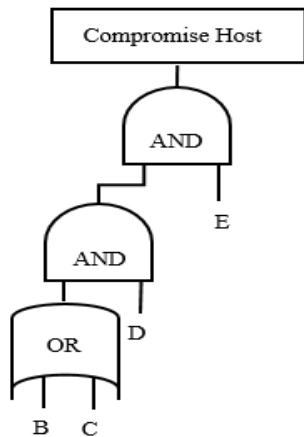


Fig.14. FPC and IPC Generated Attack Tree.

b) *Incremental Path Calculation (IPC)*: Full Path computation of the complete attack tree suffers from the lack of efficiency when all the paths are included in the computation. IPC considers reachability information from every node, and overcomes the problem of FPC. Reachability information against each node is separately maintained while calculating the attack path. The IPC approach follows this information while evaluating a particular node and includes the next attack path. Eventually, the process stops when all the possible attack paths are included.

Table 1. Reachability of Nodes

Nodes	A	B	C	D	E
Neighbors	B+C	D	D	E	-

Now, let's consider the given scenario based on the reachability shown in Table 1. The algorithm starts from the attacker host A with the target shown in Equation 5 referenced from the reachability table.

$$B + C \tag{5}$$

In the next step, reachability is checked for each of the components found in the first iteration and recursively moves forward until the target is reached. Steps after the initialization are shown in Equations 6 and 7.

$$B(D) + C(D) \tag{6}$$

$$B(D)(E) + C(D)(E) \tag{7}$$

Factorizing the final step ultimately generates the similar expression shown in Equation 2 which results in the similar tree depicted in Figure 14. In the IPC approach, repeated nodes are eliminated to avoid cycles. If any node has any ancestor in its reachability list, that ancestor is not considered again in a logical expression.

3) *Limitations*: The evaluation and complexity analysis of the logical reduction approaches show the efficient utilization of the attack tree after transformation, but these approaches have some limitations. These proposed algorithms perform well in small attack trees; however, with a larger attack tree, FPC is affected by the exponential number of nodes that it requires to process and IPC suffers from inefficient memory allocation.

E. *Efficient Attack Graph Analysis through Approximate Inference [27]*

1) *Key Motivations*: Protecting networks through effective vulnerability identification and prevention is often affected by the lack of expertise, and requires interruption to the system. Therefore, optimized resources for protection need to be determined by risk-driven security measures - which demands network risks assessment, critical

vulnerability prioritization and finally calculating the risk based on the severity and probability of the threat. Dependencies between the attacks are mostly ignored in these scenarios which eventually affect these analyses as well. Munoz-Gonzalez et al. provide an elaborate discussion on the Bayesian Attack Graph (BAG) which maintains a rigid relationship among random variables and enables modeling uncertainty about an intruder's intention and capabilities [28]. Variable elimination (VE) and Junction Tree (JT) also allow exact inference in BAG by computing unconditional probabilities for each node; however, exact inference can only be applied to a smaller graph and computation is marked as an NP-Hard problem. Approximate inference through Loopy Belief Propagation (LBP) is introduced by Munoz-Gonzalez et al. to overcome the issue with network size.

- 2) *Core Contributions:* Attack graphs are a well-known approach for analyzing and understanding attacks and identifying efficient preventive measures by the system administrators [3, 29, 30]. Static and dynamic approaches are followed to analyze the attack graphs. In the static analysis of an attack graph, the possible paths are devised based on previous knowledge about network vulnerabilities. This knowledge is eventually used in selecting appropriate countermeasures. On the other hand, dynamic analysis of attack graphs enables capturing an attack scenario at any moment of a particular attack and determines the possible target or upcoming threats. Due to nature of these analyses, these are also referred as proactive and reactive approaches [27]. Munoz-Gonzalez et al. introduced approximate inference through LBP for both static and dynamic analysis of BAG. They also proposed sequential and parallel implementations of LBP and draw a comparison between these two implementations. The efficiency of LBP in terms of allowing a system administrator to plan protective strategies in advance by monitoring estimated probability in every iteration was shown.

a) *Bayesian Attack Graph:* A Bayesian attack graph is a model built upon the concept of a Bayesian network. The network is basically a directed acyclic graph (DAG) that includes nodes that are random variables and edges that are dependencies between variables. Though attack graphs often include cycles, Munoz-Gonzalez et al. utilize the solution provided by Wang et al. to manage cycles in an attack graph to generate a DAG [31].

The nodes in a BAG represent security states an intruder can achieve. Each node of an attack graph is attributed with conditional probability $\Pr(n_i|p_i)$ which refers to the probability of the node n_i based on the probability of its parent node p_i . This computation also considers the vulnerability v_i used to compromise the node and this value is estimated based on the Common Vulnerability Scoring System (SSCVSS) [32]. To build the conditional probability table, two scenarios are

considered based on the given estimation of vulnerability. In the first scenario, all the conditions need to be satisfied to reach a particular node and build an *AND* conditional table for that particular node. On the other hand, any of the conditions will suffice to execute the attack on a node and this builds an *OR* conditional table.

BAG also allows eliminating cycles in the graph through breaking the complete graph into smaller parts instantiated with the node that initiates an attack. This approach eventually helps in converging the value of the nodes and increases the efficiency of LBP estimation.

b) *Risk Assessment:* Munoz-Gonzalez's approach contains a combination of static and dynamic risk analysis based on the BAG. In both models, the BAG is first built based on either the network topology or the analysis on alerts. Then, the static analysis approach computes the conditional probability tables of the nodes based on the CVSS score. LBP is used as an approximate inference technique to identify the vulnerable points of the network. In the dynamic assessment, the conditional probabilities are recomputed based on the detected attack in a network at any point of time. In this case, the state of the variable of the compromised node is set to 1. This updated state of one node eventually affects the posterior probabilities of the rest of the nodes.

c) *Inference of Bayesian Attack Graph:* Belief Propagation (BP) and LBP are approaches to derive inference in BAG. BP is a well-suited approach for exact inference in tree or poly-tree graphs, while LBP can compute approximate inference in graph structures containing cycles [33]. Bayesian Network facilitates computing the joint probability of a set of random variables in the form of product operation of factors of the variables or their subset. This decomposition of factors introduces factor nodes, and eventually a factor graph along with the nodes representing the random variables [34]. Unconditional probability of all the nodes in a graph is calculated by BP through computing the messages passing between factor nodes and variable nodes. Dynamic analysis is different in the sense that it requires re-computation of the factor of the nodes whose value depend on the updated variables. LBP is an extension of Belief Propagation. Munoz-Gonzalez describes two different types of LBP: Serial Loopy Belief Propagation (S-LBP) and Parallel Loopy Belief Propagation (P-LBP). In the case of S-LBP, messages are computed iteratively for each node until it reaches a maximum range of iterations or the value converges. A scheduling technique is applied for favorable convergence and better efficiency of this approach. P-LBP allows simultaneous updating of messages for all variables and factor nodes based on the value achieved in the earlier iterations.

- 3) *Limitations:* Munoz-Gonzalez shows an experimental result of the efficiency of the proposed S-LBP and P-LBP methods in terms of accuracy, convergence and execution time. But, LBP cannot always guarantee convergence [35]. Also, the convergence of LBP does not always ensure

correctness of the probability estimations. The alternative approaches to ensure convergence are found to be inefficient in most of the scenarios [27].

III. SYNTHESIS OF CORE PAPERS

The approach for building an attack graph to attack tree conversion technique first requires identifying paths to construct the attack graph, then eventually converting the attack graph into an attack tree. In a general attack tree modeling approach, attack goals are decomposed into sub-goals until the sub-goals become atomic attacks [36]. Attack trees can also be constructed manually or using already developed attack tree designing tools [37, 38, 39, 40]. Section II highlighted the various models of developing an attack graph based on the information acquired from different alerts and network configuration [6, 14, 18], the process of analyzing the attack path [27] and scalable representation of an attack tree [25]. Part of the research presented in this paper is to bridge between these concepts to develop an automatic attack tree generation approach. Therefore, in this section, the contribution of these concepts in our research will be discussed.

A. Attack Graph Modeling

Three of the five selected papers discuss different approaches to attack scenario modeling and generating attack graphs. These approaches are considered to be the initial step to formulate attack trees from general alerts and system configuration. In attack graph generation, not only does system configuration plays an important role in determining the model of an attack, identifying attack relevant alerts and correlating those alerts based on their attributes is also vital. The proposed methods of Qin et al. and Zhu et al. concentrate on determining the relationship between alerts through an alert correlation approach. However, a model-checking approach is used by Ou et al. to develop an attack graph in a scalable manner.

Both Qin et al. and Zhu et al. developed their model based on the dataset of the Grand Challenge Problem (GCP) from DARPA. From this large volume dataset, they extracted some important attributes which mostly described an alert and tried to find the relationship between these alerts based on those attribute values. Qin et al. defined the relationship between alerts based on the similarity of the attribute values, while Zhu et al. selected six primary features that, in some cases, considered similar values in different attributes (e.g., the similarity of the source IP of one alert with the destination IP of another alert). Qin et al. thus reduced the number of alerts generated by different IDS or firewalls and eventually grouped the alerts. This is an essential approach in determining the particular set of information that must be analyzed from a large volume of data. Qin et al. then allowed system administrator's input to determine the priority of hyper-alerts generated through fusion of alerts. The system administrator uses configuration information of the host and network to determine the rank of the alerts.

Prioritized alerts were further arranged based on the

time sequence and provided as input for the Granger Casualty Test (GCT) to determine the ultimate relationship [6]. Zhu et al. considered the time series based relationship as a temporary measure, and determined the causal relation based on forward and backward correlation strength. The temporal and causal relationship eventually built the cell weights of the Alert Correlation Matrix (ACM). Zhu et al. also used Multilayer Perception (MLP) and Support Vector Machine (SVM) to build the relation between new alerts and hyper-alerts and also update the ACM with new relationships and correlation weights [18]. Zhu et al. used a stronger measure to correlate different alerts and generated new attack scenarios by determining the relationship with new alerts and existing hyper-alerts from the ACM. The alert aggregation approach used by Qin et al. can complement Zhu's method by reducing the number of alerts and eventually increasing efficiency. Both of these approaches require human intervention either for prioritizing alerts or for supervised learning.

The logic-based approach proposed by Ou et al. is mostly dependent on the system configuration, and defines the attack path based on the rule determined from the configuration [14]. The rule also includes the possible vulnerabilities associated with a host and network. These features were already provided by MulVAL, but Ou et al. added the attack simulation trace to generate the final attack graph [14, 16]. Although this approach determined the attack path by iteratively looking into a large set of rules and associated attack traces, it lacked the feature of dynamically analyzing the data generated by various IDS, and highly depended on the rules and vulnerabilities set by the system administrator.

B. Attack Graph Analysis

Attack graph analysis is another important phase of constructing the proper model of attacks. It facilitates the process of assessing the risk and identifying the actual vulnerable point on the attack path both statically or dynamically. Researchers have proposed different matrices for risk assessment over the years: rate at which the asset can be acquired, measure of risk based on the weakest path, measurement based on the number of attacks, length of the shortest path and standard deviation, normalized mean, median and mode of the length of the paths [15, 41, 42, 43, 44]. Munoz-Gonzalez et al. developed their approach for analyzing an attack tree based on Bayesian Attack Graph (BAG) and used the Common Vulnerability Scoring System (CVSS) values as a standard of measurement [45]. Their approach identified an attack path based on the pre-recorded information through static analysis and detected possible threats through dynamic analysis. The inclusion of Loopy Belief Propagation (LBP) to measure approximate inference enabled the algorithm to be applicable in a larger network. An attack graph analysis designed with this approach basically helps in devising a concrete attack path by removing the less vulnerable nodes from the attack graph, and can help in reducing the false positive alerts using both static and dynamic analysis.

C. Logic Reduction

Nodes in attack trees are built with various forms of boolean algebra where the status of a particular network or host plays the role of a boolean variable. These forms of boolean logic can often be further reduced to a simplified structure for better understanding and faster analysis. Hong et al. proposed two logic reduction algorithms to simplify the structure of attack trees [25]. These approaches can be applied partially or fully in an already constructed attack tree and can improve the performance of attack tree evaluation. In their first algorithm, all vulnerable nodes in an attack path are included with groups containing the same nodes to define the simpler attack path. In their second approach, vulnerable nodes are iteratively included in a path based on the reachability of the initial attack node. Although both of these approaches suffer from memory and performance efficiency while deriving the simplified version of attack tree, they showed their superiority by constructing less-complex attack trees.

D. Summary

Papers for this literature study have been chosen based on their relevance in different steps of developing an attack tree from an attack graph. The structure of the attack tree basically depends on either the configuration of the host/network system, or the information and alerts produced by the IDS. The initial set of selected papers concentrates on the construction of attack graphs based on the host or network configuration or generated alert data. One paper discussed vulnerability analysis which eventually can determine the efficiency of the already generated attack graph developed by the first set of papers. The last paper in the selected set can be used to generate a simpler attack graph from a full attack tree through two proposed algorithms. The approaches discussed from these papers manage aspects of developing an attack tree; yet, these approaches either suffer from the lack of efficiency or cannot completely satisfies all the required properties and attributes for efficient evolution from the raw information or rule to build an attack tree. These limitations left opportunities for further research to find effective techniques for attack tree construction that can reduce false positive alerts, manage a large volume of data and larger networks, and be applied in real-time intrusion analysis.

IV. THE CURRENT STATE OF THE ART

Attack graphs and attack trees are used for attack path analysis. Over the years, several attack models have been developed based on these two basic ARMs, and different matrices have been included for computing the possible attack paths from attackers' and system administrators' perspectives. These models are also used in the cloud environment for intrusion detection and attack prediction. In this section, recent researches on these areas will be highlighted from the context of a variety of applications in the various environment.

A. General Attack Analysis Approaches

Attack trees and graphs are generally used to create a model to analyze attacks that occur in computer networks. Attack and Protection trees, Defense Trees and Attack-Defense trees were a few of the models built upon the regular attack tree and used to analyze attacks from both an attacker's and a defender's point of view [46, 47, 48]. Edge et al. built a Protection tree to identify the possible protection areas by analyzing the attack tree and calculating the impact, probability and cost of the attacker's goal. In this scenario, the Protection tree is separate from the Attack tree, but built upon calculated data found in the attack tree [46]. Bistarelli et al. introduced a Defense tree as an extension of an attack tree. It can be used to measure the return of the actions for both an attacker and defender [48]. Two indexes, Return on Investment (ROI) and Return on Attack (ROA), define the success of the defender in terms of applying security measures against an attack and success of an attacker in terms of successful exploitation of a vulnerability, respectively. Kordy et al. have devised their model to overcome some limitations of attack trees. First, attack trees are not capable of identifying the interaction between attacks in a system and defenses implanted against the particular attack. Second, attack tree cannot properly visualize the evolution of security for a system from the action of an attacker and defender. Unlike Protection tree, both Defense tree and Attack-Defense tree accommodate two types of nodes to refer actions regarding attack and defense. These models are also capable of analyzing the economic effectiveness of actions taken either for an attack or defense.

Recent research reveals the application of attack trees through game-theoretic analysis of attack scenarios. In a game-theoretic approach, an attacker's best option is deduced based on multi-object optimization and an administrator or defender improves his information by keeping track of every action of the attacker [49]. Game theory is considered to be an adaptive and faster algorithm to learn to determine an attacker's future action and behavior. Kordy et al. showed the equivalence of both the Attack-Defense tree and binary Zero-Sum extensive game in terms of maintaining outcomes and structures while converting between these two forms [50]. The game theoretic analysis approach involves actions of an attacker followed by the actions of a defender in the form of an attack graph. It eventually helps in determining the preventive measure for each attack.

Attack tree computation with multiple parameters illustrates the complete analysis associated with an attacker's action. Initially attack trees were analyzed only with cost. Buldas et al. included success probability and penalty for both failure and success in their analysis [51]. Later, Jurgenson et al. introduced the concept of *Gains* and *Expanses* calculated from attack cost and achievement, and probability and penalties in their analysis which successfully helped in analyzing the possible attack path with a more realistic approach [52].

Though attack graphs are an established tool for risk analysis and attack prediction – it does not consider the

uncertainty of probability while measuring the probability of vulnerability exploitation. GhasemiGol et al. has recently developed an attack forecasting model using intrusion alerts, active responses, and a dependency graph considering uncertainty of attack probabilities [53]. Their approach includes an uncertainty-aware attack graph as the primary component. Uncertainty of the probability of attacks defines the probability of vulnerability exploitation of this graph - active attacks captured by the IDS increases the probability of other attacks in the future while responses from the intrusion response components against an attack lower the probabilities of the corresponding attack. Dependency graphs play a vital role in this approach by defining the relationship between the services and the processes of the network.

B. Attack Analysis in Cloud Environment

ARMs are currently used successfully in constructing attack scenarios and determining system configurations in the cloud environment as well [54, 55, 56, 57, 58]. Nagaraju et al. have used attack graphs in the cloud environment to ensure clarity in configuration and validation of cloud components, especially during multi-factor authentication [54]. They have assessed the efficiency of multi-factor authentication by creating configuration graphs with access points dedicated for authentication which later constructs an attack graph associated with other components of the system. These graphs help system administrators identify the vulnerable points in a cloud architecture. Network Intrusion detection and Countermeasures Election (NICE) is an intrusion detection system developed based on the concept of an attack graph [56]. In a cloud environment, each server is monitored through NICE-A, an intrusion detection component of NICE, and can be moved to the inspection stage based on the vulnerability points assessed by NICE.

Ficco et al. introduced Attack Evaluation Tree (AET) as an event correlation process to detect intrusions in the cloud environment. Cloud monitoring systems, such as mOSAIC monitoring cloudlets, cloud agency, and IDS systems in other layers, collect information from different layers and pass the information to a Security Engine (SE). This SE adopts a Bayesian network based correlation technique to identify a causal relationship between the generated events. A predefined set of rules are defined by the system administrator which are carried out by the SE to detect anomalous activities of the cloud environment [55].

Wang et al. proposed a modified Attack-Defense Tree (ADT) to solve the threat risk analysis (TRA) problem in cloud environment [59]. Existing approaches of threat risk analysis with attack trees take reflective thinking of the attacker and defender into account, which is not always guaranteed. The proposed approach of Wang et al. is built upon the interactive-oriented TRA scheme and ADT which ensure proper response to the perceived and actual intrusions in cloud services considering the thinking of the attacker and defender.

Alhebaishi et al. present another threat modeling

technique using attack graph and attack trees [60]. They constructed two cloud infrastructures using the concept and services provided by Cisco, VMWare and OpenStack, and built the threat models to understand, evaluate and improve security attributes in a cloud environment. Their proposed approach shows that attack trees can be used at a higher abstraction level to design the attack path while an attack graph can accommodate specific vulnerabilities. This approach also facilitates the quantification process using security matrices based on these attack representation models.

V. FUTURE DIRECTION

The primary goal of our research is to develop an advanced technique to generate attack trees. Application of previous attack representation models demonstrates that an advanced and automatic generation approach of these models will make the attack analysis more dynamic in a different environment. A strong background of mathematical analysis and application of various analytical approaches to attack trees make it a strong candidate to continue further research. In this section, we will explore the opportunities of research related to these attack representation models as a part of our future direction.

A. Attack Graph to Attack Tree Conversion

An attack tree is effective for both qualitative and quantitative analysis as a detailed analysis can be possible through feature extraction of attacks, and mathematical analysis is convenient by assigning a cost and probability for each action of an attack node. Attack trees, thus, increase the possibility of identifying certain attacks or facilitating computation of the cheapest preventative cost of a possible attack [61]. However, the success of an attack tree mostly depends on its completeness and its formation. Completeness refers to the fact that an attack tree contains all the possible atomic attacks at the node level and maintains the proper sequence and path. Proper formation ensures correct placement of an attack in the attack tree with detailed information. However, generating an attack tree that maintains both features is a challenge.

Research around the attack tree generation process has been running for years, but they are suffering from lack of scalability, lack of completeness and generated models are mostly limited to a specific attack scenario [62, 63, 64]. A solution of an attack graph to attack tree generation is proposed through min-cut computation, which is an NP-Complete problem [65, 66]. Attack graph generation techniques discussed in this literature either take IDS generated alerts or system configurations for analysis and generate the graph. Some of these approaches are not capable of correlating new alerts with older alerts, while others cannot identify new attacks as they do not perform dynamic alert analysis or lack completeness.

Considering all these facts and findings, we have planned to find a unified solution to map all sorts of

attacks in a general attack graph and transform graphs into trees ensuring scalability and performance in the transformation approach. We will incorporate existing approaches to attack graph construction from alerts and system configurations. We will also analyze the areas required to fill up the gap to unify the graphs generated through two different approaches: the alert correlation based approach and the model checking based approach. None of the papers discussed in this literature proposed any efficient technique to convert attack graphs to attack trees. BAG based analysis, described in Section II, contains a discussion on the elimination of cycles from a graph which can help in finding a probable tree structure; however, this will need further research to build a computationally sound attack tree model.

B. Attack Analysis in Cloud Environment

Cloud computing is a convenient model for on-demand access to shared computational architecture which requires minimal effort for resource management and can be provisioned rapidly. Cloud computing has been developing for a decade with numerous opportunities and challenges. Trustworthiness is one of the principal concerns associated with the cloud environment. To subside the vulnerabilities associated with a cloud environment, it is necessary to employ better anomaly and misuse detection systems. The effect of flooding attacks, insider attacks, and attacks on virtual machines can be eliminated through faster analysis alerts from Intrusion Detection Systems (IDS) [67]. IDS in cloud can be categorized into four primary categories: Distributed IDS, Network-based IDS, Host-based IDS and Hypervisor-based IDS. There are two problems associated with the large volume of data or alerts generated by these IDS - 1. Correlating the large volume of alerts generated by different IDS and generated from different layers of cloud architecture, and 2. Handling large volumes of false positive or false negative alert data [68]. Recent research shows that ARMs can be applied in a cloud environment [55, 56]. As a part of our future research, we also plan to extend the attack tree generation approach developed for the smaller network to larger cloud architecture. We will explore how data generated from different layers and IDS can be grouped and correlated to generate attack graphs, and whether the general approach for attack graph to attack tree conversion can be applied in a cloud scenario as well.

C. Real-time Attack Analysis

Real-time attack or misuse analysis has been a part of active research for a long time [69, 70, 71]. The purpose of this research is to identify efficient techniques to detect anomalies in a system through active monitoring of the traffic and alerts generated by IDS and assessing and predicting the possible risk of the system nearly real-time. These sorts of prediction or detection systems allow system administrators to take a preventive measure from inside or outside attacks. Though different models have been discussed to demonstrate their success in real-time intrusion detection, ARMs have never been applied in

this scenario. Quantitative and qualitative analysis of attack trees have the potential to develop a model based on real-time anomaly analysis. We are including this option as a part of our future research goal to explore how attack tree modeling brings advancement in this area.

VI. CONCLUSION

Throughout this literature study, different attack representation models have been discussed along with their opportunities and techniques to develop them from raw alerts and network or host configurations. Alert analysis methodologies and attack tree transformation techniques as steps of an attack graph to attack tree conversion process have also been discussed. Recent research conducted around these models have been explored, and based on all the theories and facts, a list of new prospective research dimensions has been introduced. From our analysis, we have found that attack trees are more a computationally efficient model for attack analysis; therefore, in our future exploration, we will prioritize the use of attack trees from a different direction. We strongly believe that an advanced attack tree construction approach makes attack analysis more convenient in various environments and eventually makes real-time intrusion detection more efficient.

REFERENCES

- [1] R. W. Shirey, "Internet security glossary, version 2," 2007.
- [2] S. Cheung, U. Lindqvist, and M. W. Fong, "Modeling multistep cyber attacks for scenario recognition," in *DARPA information survivability conference and exposition, 2003. Proceedings*, vol. 1. IEEE, 2003, pp. 284–292.
- [3] O. Sheyner, J. Haines, S. Jha, R. Lippmann, and J. M. Wing, "Automated generation and analysis of attack graphs," in *Security and privacy, 2002. Proceedings. 2002 IEEE Symposium on*. IEEE, 2002, pp. 273–284.
- [4] B. Schneier, "Attack trees," *Dr. Dobbs journal*, vol. 24, no. 12, pp. 21–29, 1999.
- [5] S. Mauw and M. Oostdijk, "Foundations of attack trees," in *International Conference on Information Security and Cryptology*. Springer, 2005, pp. 186–198.
- [6] X. Qin and W. Lee, "Statistical causality analysis of infosec alert data," in *International Workshop on Recent Advances in Intrusion Detection*. Springer, 2003, pp. 73–93.
- [7] A. Valdes and K. Skinner, "Probabilistic alert correlation," in *International Workshop on Recent Advances in Intrusion Detection*. Springer, 2001, pp. 54–68.
- [8] P. A. Porras, M. W. Fong, and A. Valdes, "A mission-impact-based approach to infosec alarm correlation," in *International Workshop on Recent Advances in Intrusion Detection*. Springer, 2002, pp. 95–114.
- [9] I. D. W. Group *et al.*, "Intrusion detection message exchange format data model and extensible markup language (xml) document type definition," *Internet-Draft*, pp. 21–26, 2003.
- [10] K. Julisch and M. Dacier, "Mining intrusion detection alarms for actionable knowledge," in *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2002, pp. 366–375.

- [11] C. Granger, "Investigating causal relations by econometric," *Rational Expectations and Econometric Practice*, vol. 1, p. 371, 1981.
- [12] G. M. Ljung and G. E. Box, "On a measure of lack of fit in time series models," *Biometrika*, vol. 65, no. 2, pp. 297–303, 1978.
- [13] A. J. Hayer, *Probability and Statistics for Engineers and Scientists*. Duxbury Press, 2002.
- [14] X. Ou, W. F. Boyer, and M. A. McQueen, "A scalable approach to attack graph generation," in *Proceedings of the 13th ACM conference on Computer and communications security*. ACM, 2006, pp. 336–345.
- [15] C. Phillips and L. P. Swiler, "A graph-based system for network vulnerability analysis," in *Proceedings of the 1998 workshop on New security paradigms*. ACM, 1998, pp. 71–79.
- [16] X. Ou, S. Govindavajhala, and A. W. Appel, "Mulval: A logic-based network security analyzer." in *USENIX security*, 2005.
- [17] P. Rao, K. Sagonas, T. Swift, D. S. Warren, and J. Freire, "Xsb: A system for efficiently computing well-founded semantics," in *International Conference on Logic Programming and Non-monotonic Reasoning*. Springer, 1997, pp. 430–440.
- [18] B. Zhu and A. A. Ghorbani, "Alert correlation for extracting attack strategies," *IJ Network Security*, vol. 3, no. 3, pp. 244–258, 2006.
- [19] F. Cuppens and A. Mieke, "Alert correlation in a cooperative intrusion detection framework," in *Security and privacy, 2002. proceedings. 2002 IEEE symposium on*. IEEE, 2002, pp. 202–215.
- [20] F. Cuppens and R. Ortalo, "Lambda: A language to model a database for detection of attacks," in *International Workshop on Recent Advances in Intrusion Detection*. Springer, 2000, pp. 197–216.
- [21] O. Dain and R. K. Cunningham, "Fusing a heterogeneous alert stream into scenarios," in *Proceedings of the 2001 ACM workshop on Data Mining for Security Applications*, vol. 13. Citeseer, 2001.
- [22] S. T. Eckmann, G. Vigna, and R. A. Kemmerer, "Statl: An attack language for state-based intrusion detection," *Journal of computer security*, vol. 10, no. 1, 2, pp. 71–103, 2002.
- [23] P. Ning, Y. Cui, and D. S. Reeves, "Constructing attack scenarios through correlation of intrusion alerts," in *Proceedings of the 9th ACM conference on Computer and communications security*. ACM, 2002, pp. 245–254.
- [24] S. J. Templeton and K. Levitt, "A requires/provides model for computer attacks," in *Proceedings of the 2000 workshop on New security paradigms*. ACM, 2001, pp. 31–38.
- [25] J. B. Hong, D. S. Kim, and T. Takaoka, "Scalable attack representation model using logic reduction techniques," in *2013 12th IEEE International Conference on Trust, Security and Privacy in Computing and Communications*. IEEE, 2013, pp. 404–411.
- [26] B. Schneier, *Secrets and lies: digital security in a networked world*. John Wiley & Sons, 2011.
- [27] L. Munoz-Gonzalez, D. Sgandurra, M. Barrere, and E. C. Lupu, "Exact inference techniques for the analysis of bayesian attack graphs," *arXiv preprint arXiv: 1510.02427*, 2015.
- [28] N. Poolsappasit, R. Dewri, and I. Ray, "Dynamic security risk management using bayesian attack graphs," *IEEE Transactions on Dependable and Secure Computing*, vol. 9, no. 1, pp. 61–74, 2012.
- [29] M. Albanese, S. Jajodia, and S. Noel, "Time-efficient and cost-effective network hardening using attack graphs," in *IEEE/IFIP International Conference on Dependable Systems and Networks (DSN 2012)*. IEEE, 2012, pp. 1–12.
- [30] K. Ingols, M. Chu, R. Lippmann, S. Webster, and S. Boyer, "Modeling modern network attacks and countermeasures using attack graphs," in *Computer Security Applications Conference, 2009. ACSAC'09. Annual*. IEEE, 2009, pp. 117–126.
- [31] L. Wang, T. Islam, T. Long, A. Singhal, and S. Jajodia, "An attack graph-based probabilistic security metric," in *IFIP Annual Conference on Data and Applications Security and Privacy*. Springer, 2008, pp. 283–296.
- [32] Forum of Incident Response and Security Teams, "Common vulnerability scoring system, v3 development update," <https://www.first.org/cvss>, [Accessed: 12-21-2016].
- [33] J. Pearl, "Reverend bayes on inference engines: A distributed hierarchical approach," in *AAAI*, 1982, pp. 133–136.
- [34] C. M. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2006.
- [35] Y. Weiss, "Correctness of local probability propagation in graphical models with loops," *Neural computation*, vol. 12, no. 1, pp. 1–41, 2000.
- [36] V. Saini, Q. Duan, and V. Paruchuri, "Threat modeling using attack trees," *Journal of Computing Sciences in Colleges*, vol. 23, no. 4, pp. 124–131, 2008.
- [37] K. S. Edge, "A framework for analyzing and mitigating the vulnerabilities of complex systems via attack and protection trees," DTIC Document, Tech. Rep., 2007.
- [38] K. Edge, R. Raines, M. Grimaila, R. Baldwin, R. Bennington, and C. Reuter, "The use of attack and protection trees to analyze security for an online banking system," in *System Sciences, 2007. HICSS 2007. 40th Annual Hawaii International Conference on*. IEEE, 2007, p. 144b.
- [39] I. Inc, "Attacktree+," <https://www.isograph.com/software/attacktree/>, [Accessed: 12-31-2016].
- [40] A. T. Ltd., "Securitree," <http://www.amenaza.com/>, [Accessed: 12-31-2016].
- [41] R. Lippmann, K. Ingols, C. Scott, K. Piwowarski, K. Kratkiewicz, M. Artz, and R. Cunningham, "Validating and restoring defense in depth using attack graphs," in *MILCOM 2006-2006 IEEE Military Communications conference*. IEEE, 2006, pp. 1–10.
- [42] J. Pamula, S. Jajodia, P. Ammann, and V. Swarup, "A weakest-adversary security metric for network configuration security analysis," in *Proceedings of the 2nd ACM workshop on Quality of protection*. ACM, 2006, pp. 31–38.
- [43] N. Idika and B. Bhargava, "Extending attack graph-based security metrics and aggregating their application," *IEEE Trans- actions on dependable and secure computing*, vol. 9, no. 1, pp. 75–85, 2012.
- [44] R. Ortalo, Y. Deswarte, and M. Kaaniche, "Experimenting with quantitative evaluation tools for monitoring operational security," *IEEE Transactions on Software Engineering*, vol. 25, no. 5, pp. 633–650, 1999.
- [45] L. Munoz-Gonzalez, D. Sgandurra, A. Paudice, and E. C. Lupu, "Efficient attack graph analysis through approximate inference," *arXiv preprint arXiv: 1606.07025*, 2016.
- [46] K. S. Edge, G. C. Dalton, R. A. Raines, and R. F. Mills, "Using attack and protection trees to analyze threats and defenses to homeland security," in *MILCOM 2006-2006*

- IEEE Military Communications conference*. IEEE, 2006, pp. 1–7.
- [47] B. Kordy, S. Mauw, S. Radomirović, and P. Schweitzer, “Foundations of attack–defense trees,” in *International Workshop on Formal Aspects in Security and Trust*. Springer, 2010, pp. 80–95.
- [48] S. Bistarelli, F. Fioravanti, and P. Peretti, “Defense trees for economic evaluation of security investments,” in *First International Conference on Availability, Reliability and Security (ARES’06)*. IEEE, 2006, pp. 8–pp.
- [49] Y. Luo, F. Szidarovszky, Y. Al-Nashif, and S. Hariri, “Game theory based network security,” *Journal of Information Security*, vol. 1, no. 1, p. 41, 2010.
- [50] B. Kordy, S. Mauw, M. Melissen, and P. Schweitzer, “Attack–defense trees and two-player binary zero-sum extensive form games are equivalent,” in *International Conference on Decision and Game Theory for Security*. Springer, 2010, pp. 245–256.
- [51] A. Buldas, P. Laud, J. Priisalu, M. Saarepera, and J. Willemson, “Rational choice of security measures via multi-parameter attack trees,” in *International Workshop on Critical Information Infrastructures Security*. Springer, 2006, pp. 235–248.
- [52] A. Jurgenson and J. Willemson, “Computing exact outcomes of multi-parameter attack trees,” in *OTM Confederated International Conferences “On the Move to Meaningful Internet Systems”*. Springer, 2008, pp. 1036–1051.
- [53] M. GhasemiGol, A. Ghaemi-Bafghi, and H. Takabi, “A comprehensive approach for network attack forecasting,” *Computers and Security*, vol. 58, pp. 83–105, 2016.
- [54] S. Nagaraju and L. Parthiban, “Analyzing configurations of authentication access points in cloud using attack graph,” in *2015 IEEE International Conference on Computer Graphics, Vision and Information Security (CGVIS)*. IEEE, 2015, pp. 72–76.
- [55] M. Ficco, L. Tasquier, and R. Aversa, “Intrusion detection in cloud computing,” in *P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC), 2013 Eighth International Conference on*. IEEE, 2013, pp. 276–283.
- [56] C. J. Chung, P. Khatkar, T. Xing, J. Lee, and D. Huang, “Nice: Network intrusion detection and countermeasure selection in virtual network systems,” *IEEE transactions on dependable and secure computing*, vol. 10, no. 4, pp. 198–211, 2013.
- [57] Z. Hu, S. Gnatyuk, O. Koval, V. Gnatyuk, and S. Bondarovets, “Anomaly Detection System in Secure Cloud Computing Environment,” *International Journal of Computer Network and Information Security*, vol. 9, no. 4, pp. 10–21, 2017.
- [58] S. Hashemi and P. Hesarlo, “Security, privacy and trust challenges in cloud computing and solutions,” *International Journal of Computer Network and Information Security*, vol. 6, no. 8, pp. 34–40, 2014.
- [59] P. Wang, W. Lin, P. Kuo, H. Lin, and T. Wang, “Threat risk analysis for cloud security based on Attack-Defense Trees,” in *2012 8th International Conference on Computing Technology and Information Management (ICCM)*, IEEE, 2012, pp. 106–111.
- [60] N. Alhebaishi, L. Wang, S. Jajodia, and A. Singhal, “Threat Modeling for Cloud Data Center Infrastructures,” in *International Symposium on Foundations and Practice of Security*, Springer, 2016, pp. 302–319.
- [61] B. Kordy, S. Mauw, S. Radomirovic, and P. Schweitzer, “Attack–defense trees,” *Journal of Logic and Computation*, p. 29, 2012.
- [62] R. Vigo, F. Nielson, and H. R. Nielson, “Automated generation of attack trees,” in *2014 IEEE 27th Computer Security Foundations Symposium*. IEEE, 2014, pp. 337–350.
- [63] M. G. Ivanova, C. W. Probst, R. R. Hansen, and F. Kammuller, “Attack tree generation by policy invalidation,” in *IFIP International Conference on Information Security Theory and Practice*. Springer, 2015, pp. 249–259.
- [64] S. Pinchinat, M. Acher, and D. Vojtisek, “Towards synthesis of attack trees for supporting computer-aided risk analysis,” in *International Conference on Software Engineering and Formal Methods*. Springer, 2014, pp. 363–375.
- [65] J. Dawkins and J. Hale, “A systematic approach to multi-stage network attack analysis,” in *Information Assurance Workshop, 2004. Proceedings. Second IEEE International*. IEEE, 2004, pp. 48–56.
- [66] M. Bruglieri, F. Maffioli, and M. Ehrgott, “Cardinality constrained minimum cut problems: complexity and algorithms,” *Discrete Applied Mathematics*, vol. 137, no. 3, pp. 311–341, 2004.
- [67] S. Roschke, F. Cheng, and C. Meinel, “Intrusion detection in the cloud,” in *Dependable, Autonomic and Secure Computing, 2009. DASC’09. Eighth IEEE International Conference on*. IEEE, 2009, pp. 729–734.
- [68] D. Yu and D. Frincke, “A novel framework for alert correlation and understanding,” in *International Conference on Applied Cryptography and Network Security*. Springer, 2004, pp. 452–466.
- [69] T. F. Lunt *et al.*, “Real-time intrusion detection,” *COMPCOM Spring*, vol. 89, pp. 348–353, 1989.
- [70] K. Haslum, M. E. Moe, and S. J. Knapskog, “Real-time intrusion prevention and security analysis of networks using hmms,” in *2008 33rd IEEE Conference on Local Computer Networks (LCN)*. IEEE, 2008, pp. 927–934.
- [71] A. K. Ghosh, C. Michael, and M. Schatz, “A real-time intrusion detection system based on learning program behavior,” in *International Workshop on Recent Advances in Intrusion Detection*. Springer, 2000, pp. 93–109.

Authors’ Profiles



Md Shariful Haque received his B.Sc. in Computer Science and Engineering Department in 2008 from University of Dhaka, Bangladesh. He is now a Ph.D. student in the department of Computer Science in the University of Alabama and working as a part of Digital Forensic and Control System Security Lab. His research interests lie in the areas of cloud security and privacy.



Dr. Travis Atkison received a B.S. in Electrical Engineering and a B.S. in Computer Science in 1995 from the University of Alabama, a M.S. degree in Computer Science in 1997 from University of Alabama, and a Ph.D. in Computer Science in 2009 from Mississippi State University. He, currently, is an Assistant Professor of Computer Science and the Director of the Digital Forensics and Control System Security Lab (DCSL) at the University of Alabama. His major research avenues include

transportation infrastructure security, control systems security, malicious application detection, and digital forensics.

How to cite this paper: Md. Shariful Haque, Travis Atkison, "An Evolutionary Approach of Attack Graph to Attack Tree Conversion", *International Journal of Computer Network and Information Security(IJCNIS)*, Vol.9, No.11, pp.1-16, 2017.DOI: 10.5815/ijcnis.2017.11.01