Defence Science Journal, Vol. 66, No. 6, November 2016, pp. 559-567, DOI : 10.14429/dsj.66.10795

REVIEW PAPER

Attack Graph Generation and Analysis Techniques

Mridul Sankar Barik*, Anirban Sengupta, and Chandan Mazumdar

**Department of Computer Science and Engineering, Jadavpur University, Kolkata - 700 032, India*

**E-mail: mridulsankar@gmail.com*

ABSTRACT

As computer networks are emerging in everyday life, network security has become an important issue. Simultaneously, attacks are becoming more sophisticated, making the defense of computer networks increasingly difficult. Attack graph is a modelling tool used in the assessment of security of enterprise networks. Since its introduction a considerable amount of research effort has been spent in the development of theory and practices around the idea of attack graph. This paper presents a consolidated view of major attack graph generation and analysis techniques.

Keywords: Attack graph, alert correlation, network hardening, security metric

1. INTRODUCTION

Defending large scale enterprise networks from adversary attacks is an uphill task faced by present day network administrators. Defense approaches against such attacks traditionally have been mostly host centric, where attention is given to identifying vulnerabilities of the individual hosts and taking measures to mitigate them. Vulnerability scanning tools, such as Nessus, OpenVAS, Nexpose, etc. provide per host vulnerability information and help in achieving these objectives. However, one major problem with this approach is that it emphasises more on host specific local information and does not consider them in the light of global security context of the network. Theoretically, an exhaustive vulnerability searching and patching may lead to a secure system. However, this may not be possible in practice due to the costs involved and operational constraints. Moreover, in many cases, attackers combine elementary attacks to launch multistage attacks against critical assets. These elementary attacks exploit vulnerabilities of individual hosts and may be either remote or local. Intrusion Detection Systems, either network or host based, can detect those elementary attacks but cannot report whether they are part of a larger attack chain or not.

An attack graph is an important modelling tool used in the assessment of security of enterprise networks. Using attack graphs, network administrators can understand how an attacker can combine vulnerabilities in multiple hosts in a multi-stage attack to compromise critical resources in a network. Moreover the size of an attack graph has direct impact on the perceived risk. Intuitively, a larger attack graph can mean more number of vulnerabilities that can be exploited or more number of attack paths to a resource or more attack spread; all implying less security and hence more risk. An exhaustive attack graph of a network provides global view of its security posture, enabling quantitative assessment of the same. Such assessments, when performed periodically help a network system to evolve over time.

Since its introduction in 1998, attack graph has attracted lots of attention from researchers and a considerable amount of research effort has been spent in the development of theory and practices around the idea of attack graph. In its earlier days, dedicated security teams (called Red teams) used to determine overall security of networks by hand-drawing gigantic attack graphs and then analysing them. Obviously, this approach was tedious, error prone and did not scale up as the network size grew. This gave rise to the need for automated methods of attack graph generation. Automated techniques also guarantee that the generated attack graph is exhaustive and succinct. An exhaustive attack graph contains all possible attack paths and a succinct attack graph contains only those initial network states from where the attacker can reach the goal. Initial research proposed custom algorithms, model checking, logic based approaches as attack graph generation methods. However, the scalability issue in attack graph generation is still a challenging research problem. Other research efforts aimed at using attack graphs for analysing or quantifying security risks of enterprise networks.

Lippmann and Ingols[9] has published a survey on attack graph generation and analysis techniques, in 2005. This paper aims at providing a consolidated view of major attack graph generation and analysis techniques reported till now.

2. ATTACK GRAPH GENERATION

Graph is a natural choice as formalism while considering an automated technique for network security analysis. Dacier[3] introduced the concept of privilege graph for describing vulnerabilities of a network system. Each node in a privilege graph represents a set of privilege owned by a user or a set of users and each edge represents a vulnerability. In an attack

Received : 31 December 2015, Revised : 16 June 2016
Accepted : 18 July 2016, Online published : 31 October 2016

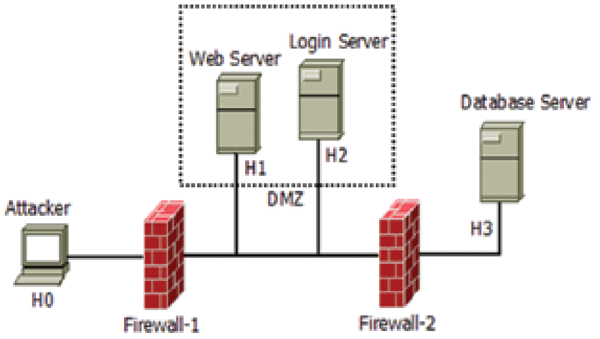

Figure 1. Simple network configuration.

tree[22] each path to a leaf node represents sequence of attacks by which an attacker can reach goal state from its initial state. An attack graph essentially is a consolidated representation of an attack tree where some or all common nodes across different attack paths are merged.

Figure 1 shows a simple network configuration that is used as a running example throughout this paper. Corresponding attack graph representations, namely the exploit dependency attack graph; logical attack graph and multiple prerequisite attack graph is described in relevant subsections.

In this network configuration Firewall-1 controls traffic between the external and internal network. Assumed location of that attacker is on host H0 in the external network. In the DMZ of internal network a web server runs on host H1 and a login server (via ssh) on host H2. The web service requires access to a back end database server which is running on host H3. Firewall-1 allows http and ssh traffic to the web server and login server respectively, and blocks all other traffic. Firewall-2 allows access to the database server coming only from the web server. Host H1 is running a vulnerable version of Apache web server, which has vulnerability (CVE-2006-3747) that allows a remote attacker to exploit and gain user privilege on the Web Server. The ssh service on H2 has a vulnerability (CVE-2002-0640), which allows remote attackers to gain user privilege. Database server H3 is a Linux box running MySQL database which has a remotely exploitable vulnerability (CVE-2009-2446), enabling attacker to gain user privilege. The Linux kernel in host H3 also has vulnerability (CVE-2004-0495) that allows local user to gain root privilege. Attacker's objective is to gain root privilege on the database server. For notational convenience short symbols are used for different vulnerabilities as shown in Table 1.

Table 1. Short symbols used for different vulnerabilities

Software	Vulnerabiity (CVE-ID)	Short symbol
Apache Web Server v1.3,	CVE-2006-3747	V1
OpenSSH v2.3.1-v3.3	CVE-2002-0640	V2
MySQL v4.0, v5.0	CVE-2009-2446	V3
Linux Kernel v2.4, v2.6	CVE-2004-0495	V4

Major approaches to attack graph generation are as follows.

2.1 State Enumeration Based Approach

State enumeration based approaches were the initial attempts to automated attack graph generation. These approaches are based on either custom algorithms or based on model checking techniques.

The concept of attack graph was first introduced by Phillips and Swiler[19,27], in 1998. In their formalism of attack graph also known as the state enumeration graph; nodes represent possible system state during execution of an attack. A system state comprises of information on host(s), user access levels and effects of the attack so far. Edges represent a change of state, caused by a single action of the attacker and may be weighted based on the attacker's effort required or the time to succeed. Based on this formalism, the authors have presented a method[28] for automated generation of attack graph with three kinds of input information: attack templates, a configuration file and an attacker profile. Attack templates represent attacks (both known and hypothesised) in the form of a sub-graph describing conditions necessary for successful execution of the attack and also any new conditions that it may enable. A configuration file contains information about the network system under consideration. This information includes network topology, configuration of network elements such as hosts, routers, firewalls, etc. Attacker profile contains information about attacker's capabilities. The attack graph generation algorithm starts from the initial state. It matches attack templates to the configuration of the network system and attacker's profile in a forward exploration manner and generates the graph iteratively. Put in other ways an attack graph is an instantiation of the attack templates to the configuration information and attacker profile. The generation method addressed the issues of redundant path, node and directed cycle elimination. But the authors have not provided any complexity analysis; only empirical results on small examples are given. The obvious problem with this kind of formalism is the issue of state-space explosion.

Ritchey and Ammann[20] first proposed the use of model checking techniques for attack graph generation. Model checking in general is a technique which checks whether a formal model M of a system satisfies a given property p or not. If the property p is false in the model M, then the model checker outputs a counter example describing a sequence of state transitions which ends in violation of the property. Model checking techniques are popularly used for verifying finite-state reactive systems, such as sequential circuit designs and communication protocols. One benefit of using model checking techniques over custom algorithms for attack graph generation is that users need not worry about the problem of handling large state space, which is otherwise elegantly handled by standard model checking software. The first step in this approach is to build a model of the system under analysis. This model includes information on hosts along with their vulnerabilities, host connectivity, current view of the attacker and exploits that change the model's state. The kind of input information required to build the system model is quite similar to the approach of Phillips and Swiler[19,27] except that in this case, the host connectivity information represent connectivity between hosts that remain after considering all traffic-filtering devices such as routers, firewalls etc. The current view of the attacker is same as that of attacker profile and the exploits describe similar things as attack templates. Authors have used the SMV model checker tool. Desirable security properties of the system are specified in temporal logic formulae and the model checker finds out whether the model of the system satisfies the formulae. If not, it generates counter examples which show the sequence of states, from the initial state to the state where the property is violated. As each state-change corresponds to the event of execution of an exploit, a counter example produces an attack path which describes sequence of exploits that leads an attacker from its initial position to a compromised resource. Set of all such counter examples would produce a complete attack graph.

Sheyner[23], *et al.* proposed a method for automated generation of attack graph using NuSMV model checking tool. The description of the model of the network is fed to a compiler which translates it to the input language of the model checker NuSMV. This model is essentially a finite labelled transition system. Each state is labelled with a propositional formula. This network model includes

(i) host connectivity information expressed as a ternary relation R(h_1, h_2, p) meaning that host h_1 can reach host h_2 on port p

(ii) trust relationship Tr(h_1, h_2) meaning that a user of host h_2 can log into h_1 without any authentication i.e. host h_1 trusts host h_2

(iii) intruder privilege levels on hosts (none/user/root)

(iv) Intrusion detection systems (IDS) as a relationship ids(h_1, h_2, a)={s, d, b} meaning that an attack a from host h_1 to h_2 is stealthy/detectable/both by an IDS sensor.

Each atomic attack is specified as rules and has four components: intruder preconditions, network preconditions, intruder effects and network effects. Safety property of the system is specified in computation tree logic (CTL). If the safety property is not satisfied, the NuSMV model checker generates all possible counter examples whereas the SMV model checker used by Ritchey and Ammann generates only one counter example. They termed the resulting graph as a scenario graph. However, compared to state enumeration graphs which take more of an 'attack centric' view of the system, scenario graphs are more generic and can model both benign and malicious system events. Sheyner[25] augmented the idea of scenario graph to include liveness requirement in addition to safety requirements. For this, he used a system model which supports both finite and infinite execution.

Performance of attack graph generation techniques based on model checking[20,23,25] is dependent on the efficiency of the model checker tools themselves, which are specifically designed for handling large state spaces. These techniques encode network state information in number of state variables which increase considerably with increasing network size, number of atomic attacks and vulnerabilities. They also face significant exponential state-space problems even for moderate-sized networks. Performance results reported by Sheyner[23], *et al.* gives some idea about this problem. For 2 hosts with 4 atomic attacks, the model has 91 bits of state information, and 110 reachable states. For 4 hosts with 8 atomic attacks, the model has 229 bits of state information, and 6190 reachable states. The resulting attack graph has 5948 nodes and 68364 edges and needed 2 h to generate.

2.2 TVA Approach

Earlier approaches to attack graph generation suffered from scalability issues as the attack graph representation used in those approaches, i.e. the state enumeration graph or the scenario graph, assumed full exponential state-space. The monotonicity assumption on attacker's behaviour, first introduced by Ammann[1], *et al.* was a key enabler in handling this issue. This assumption says that preconditions of an attack are never invalidated by successful execution of another attack. Although this may not be true in all the cases (i.e. buffer overflow attack against a service causes it to terminate, preventing further use in other attacks), the monotonicity assumption helps in reducing the complexity of analysis from exponential to polynomial. The resulting graph which enumerates all such possible exploit sequences is known as the exploit dependency graph. In worst case, this representation has number of nodes as quadratic to the number of exploits. In exploit dependency attack graph, each exploit or dependency appears only once and there are no edges between independent exploits. Whereas, in a state enumeration attack graph, there may be edges between exploits even though there are no dependency relationships between them. Moreover, a single attack path may appear more than once in a state enumeration attack graph.

Topological vulnerability analysis (TVA)[6,7] adopts a topological approach to network vulnerability analysis. It considers a set of modelled attacker exploit on a network and then finds out different sequences of exploits or attack paths starting from attacker's initial state leading to compromise of critical network assets. For this, TVA requires an extensive knowledgebase of known vulnerabilities and attack techniques. TVA attack graph generation engine uses the algorithm proposed by Ammann[1], *et al.* for attack graph generation. It has O(n^6) computational complexity, which was later improved to O(n^3) Ammann[2], *et al.*; n being the number of hosts in the network.

The attack graph model used in TVA uses two types of nodes, exploit nodes and security condition nodes. This model of attack graph is based on exploit dependency graphs. Exploit nodes represent attacks (exploitation of certain vulnerabilities) and security condition nodes represent either the attack post-condition or pre-condition. An exploit is defined by pre and post conditions. Directed edges from security condition nodes to attack nodes represent pre-conditions of an attack, of which all must be met for an attack to be successful. A directed edge from an attack node to a security condition node represents post-condition of an attack. An advantage of exploit dependency graphs is that instead of modelling hosts, exploits on hosts are modeled, thus reducing the computational complexity. On the other hand, this model requires information on low-level attack details. Vulnerability information are based on pre- and post-conditions. Figure 2 shows the exploit dependency attack graph for the example network of Fig. 1.

In exploit dependency attack graph, ovals represent exploits and are labelled with corresponding vulnerabilities. Other nodes represent either some network condition or

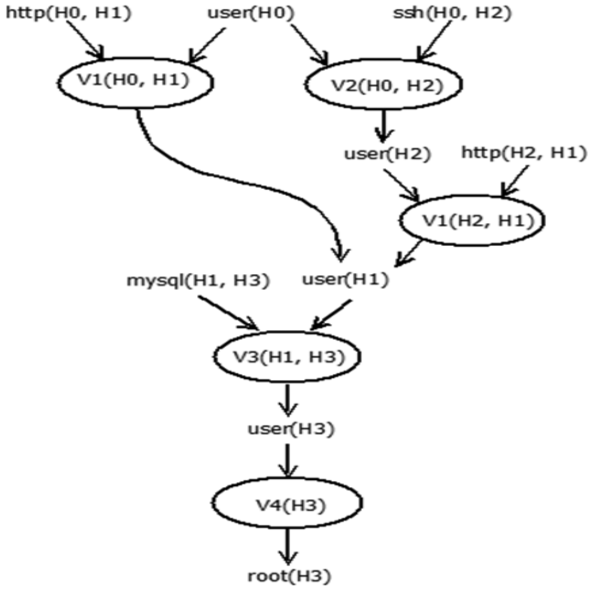

Figure 2. Exploit dependency attack graph.

attacker's capability. For example, network condition http (H0, H1) means accessibility of web service on host H1 from host H0. Attacker capability user(H0) means, attacker has user privilege on host H0. Directed edges in and out of exploit nodes identify pre and post-conditions respectively of an attack. For example, exploitation of MySQL vulnerability CVE-2009-2446 on host H3 from host H1 i.e. V3(H1, H3) requires pre-conditions user(H1) and mysql(H1, H3) and generates a post-condition user(H3). Exploit dependency attack graph elegantly enumerates different attack paths leading to a critical resource. In this example, there are two attack paths leading to attacker gaining root privilege on H3. They are V1(H0, H1) → V3(H1, H3) → V4(H3) and V2(H0, H2) → V1(H2, H1) → V3(H1, H3) → V4(H3) respectively.

2.3 Logic Programming Based Approach

Multi host, multistage vulnerability analysis (MulVAL)[17] is a logic-programming based approach to network security analysis. It uses a representation of attack graph known as logical attack graph which shows logical dependencies among attack goals and configuration information. A node in the logical attack graph is a logical statement which encodes only some part of network state. Unlike the state enumeration graph or scenario graph, it does not represent or encode the entire state of the network. Edges represent the causality relationships between various network configurations. Size of a logical attack graph is polynomial to the network being analysed. However, one requirement of logical attack graph is that the cause of an attacker's potential privilege should be expressible as a propositional formula in terms of network configuration information. A logical attack graph is a directed graph. It can also be represented as a tree.

Figure 3 shows the logical attack graph, corresponding to the simple network configuration of Fig. 1. It contains two types of nodes, i.e. derivation nodes and fact nodes. Rectangles and circles represent derivation nodes and fact nodes respectively. Derivation nodes are labeled with interaction rules. Fact nodes are labeled with logical statements in the form of a predicate applied to its arguments. Shaded circles are primitive fact nodes i.e. facts that hold true in the initial state. Un-shaded circles represent derived fact nodes, i.e. new facts that are generated as a result of application of interaction rules over existing facts. Edges in a logical attack graph represent a 'depends on' relationship.

To describe system properties MulVAL uses Datalog, which is a syntactic subset of the Prolog programming language. Required input data such as software vulnerability advisories, configuration and network topology information are encoded as Datalog tuples, whereas attack techniques are specified using Datalog rules. These rules are hand-coded and specify exploits such as code execution, file access, and privilege escalation. MulVAL uses Prolog logic engine XSB[35] as reasoning engine for evaluating rules on input facts. XSB computes all possible paths to satisfy a defined goal.

Following is an example of interaction rules in Datalog.

Rule 1: Remote exploit of a privilege-escalation vulnerability in a service program

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

This is a generic rule which specifies the pre and post-condition for this attack:

```
if
            (Program is running as User on Host as a service listening
on Protocol and Port)  AND
            (it contains a remotely exploitable vulnerability whose
impact is privilege escalation) AND
            (the attacker can access the service through the network)
then
            (the attacker can execute arbitrary code on the machine as
User)
```

Similarly Rule 2, 3 and 4 are other interaction rules that have been used for generating the example attack graph.

Rule 2: Local exploit of a privilege-escalation vulnerability in a service program

```
execCode(Host, root) :-
            vulExists(Host, VulID, Program, localExploit,
privEscalation),
            execCode(Host, User).
```

Rule 3: Direct network access

```
netAccess(Source, Target, Protocol, Port):-
            hacl(Source, Target, Protocol, Port)
            located(attacker, Source)
```

Rule 4: Multi-hop access

```
netAccess(Source, Target, Protocol, Port):-
            execCode(Source, User)
            hacl(Source, Target, dbProtocol, dbPort)
```

Network (router and firewalls) configurations are modeled as abstract host access-control lists (HACL) in form of logical statements using the predicate hacl. Predicate vulExists encodes output of vulnerability scanner tool.

Following is a complete list of labels of nodes in the logical attack graph of Fig. 3.

1. hacl(H0, H1, httpProtocol, httpPort)
2. located(Attacker, H0)
3. direct network access
4. netAccess(H0, H1, httpProtocol, httpPort)
5. networkService(H1, httpd, httpProtocol, httpPort, Apache)
6. vulExists(H1, ’CVE-2006-3747’, httpd, remoteExploit, privEscalation)
7. remote exploit of a server program
8. execCode(H1, Apache)
9. hacl(H1, H3, dbProtocol, dbPort)
10. multi-hop access
11. netAccess(H1, H3, dbProtocol, dbPort)
12. networkService(H3, mysqld, dbProtocol, dbPort, mysql)
13. vulExists(H3, ’CVE-2009-2446’, mysqld, remoteExploit, privEscalation)
14. remote exploit of a server program
15. execCode(H3, Apache)
16. vulExists(H3, ’CVE-2004-0495’, linux-kernel, localExploit, privEscalation)
17. local exploit of OS kernel
18. execCode(H3, root)
19. hacl(H0, H2, sshProtocol, sshPort)
20. direct network access
21. netAccess(H0, H2, sshProtocol, sshPort)
22. networkService(H2, sshd, sshProtocol, sshPort, SSH)
23. vulExists(H2, ’CVE-2002-0640’, sshd, remoteExploit, privEscalation)
24. remote exploit of a server program
25. execCode(H2, SSH)
26. hacl(H1, H2, httpProtocol, httpPort)
27. multi-hop access
28. netAccess(H2, H1, httpProtocol, httpPort)
29. remote exploit of a server program

The MulVAL logic programming based approach[17] has $O(n^2)$ complexity under the assumption of constant table look-up time. Empirical results show that, worst case running time varies between $O(n^2)$ to $O(n^3)$ for number of hosts upto 1000 with up to 100 vulnerabilities.

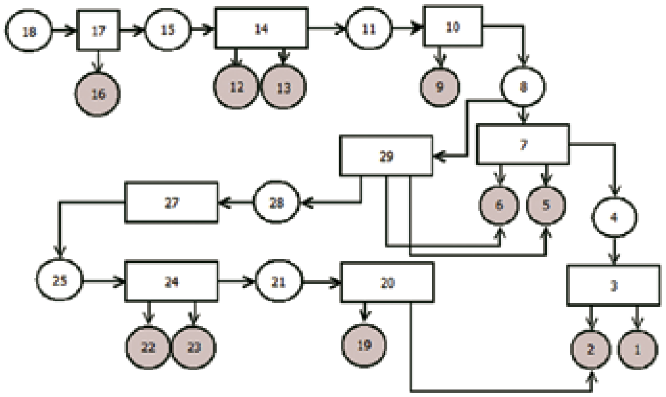

Figure 3. Logical attack graph.

2.4 NetSPA Approach

The network security planning architecture (Net SPA)[5] attack graph generation system is based on a new representation of attack graph, i.e. the multiple prerequisites graph which scales linearly to the size of the network. This tool uses readily available source of data to automatically compute network reachability, classify vulnerabilities, build the graph and recommend actions to improve network security. Multiple prerequisite graphs are much faster to generate and have greater expressive power than the author’s previous works on predictive graphs[10,11].

A multiple-prerequisite attack graph consists of three types of nodes, i.e. state nodes, prerequisite nodes, and vulnerability instance nodes. State nodes represent attacker’s level of access on a given host. Prerequisite nodes represent either a reachability group or preconditions of one or several attacks. Vulnerability instance nodes represent particular vulnerabilities. Directed edges from state nodes to prerequisite nodes represent the capabilities those states enable for the attacker. Prerequisite nodes point to vulnerability instance nodes that represent the set of attacks that the prerequisite node enables. Directed edge from vulnerability instance nodes to a single state node represent the state the attacker can reach by successfully exploiting the vulnerability. Put in other way, a state provides prerequisites, which allow exploitation of vulnerability instances, which in turn provide more states to the attacker. The concept of prerequisite nodes helps in reducing the number of edges compared to having state nodes pointing directly to vulnerability instance nodes, since many state nodes can imply the same set of attacks.

Figure 4 shows the multiple prerequisite attack graph corresponding to the network configuration of Fig. 1. Circles represent state nodes, rectangles represent prerequisite nodes, and triangles represent vulnerability instance nodes. State nodes A, B, C, D, E represent attacker’s level of access on different hosts i.e. user(H0), user(H1), user(H2), user(H3), and root(H3) respectively. Directed edge from state node A i.e. user(H0) to prerequisite node ‘Can Reach H1, H2’ represent the capability of attacker that is enabled by this state. Directed edge from prerequisite node ‘Can Reach H1, H2’ to vulnerability instance nodes V1 and V2 represent attacks enabled by this prerequisite. Only when the attacker is in state B, it enables reachability to host H3. The attacker can now gain state D via exploitation of vulnerability V3.

NetSPA generation method scales roughly as O(n log n). Experimental results show that it can scale up to 50000 hosts for synthetic networks.

3. ATTACK GRAPH BASED SECURITY ANALYSIS

The sole objective of generating an attack graph is to enable assessment of security. There are many ways in which information encoded in an attack graph can be used to gain vital insight into the global security posture of a network. This helps security administrators to make correct decisions about mitigation strategies.

3.1 IDS Alert Correlation and Sensor Placement

Multi-step network intrusions comprise of multiple attack

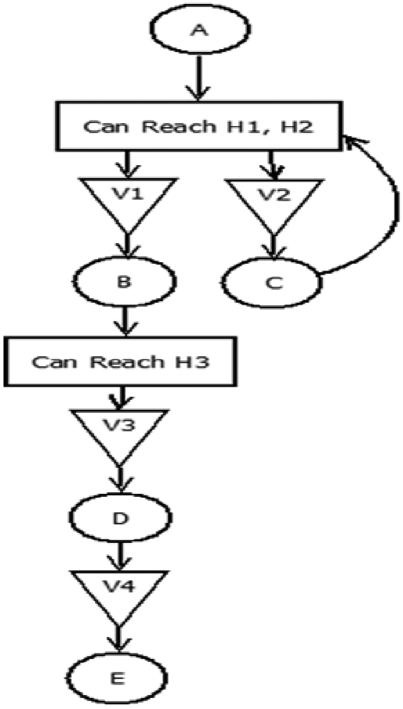

Figure 4. Multiple prerequisite attack graph.

steps with one preparing for the next. Intrusion detection system alert correlation techniques help in deciding whether an isolated alert is part of an ongoing multi-step network intrusion. It also helps in attack scenario reconstruction. Popular alert correlation techniques use prior knowledge about attack strategies or alert dependencies. Other techniques aggregate alerts with similar attributes (such as alerts with same destination addresses) or statistical patterns.

Almost all these techniques use nested loop approach where each new alert is compared with previously received alerts to mark those which prepare for the new one. This approach fits well in off-line applications such as intrusion forensics, by storing in memory index of received alerts. However, for applications like defense against multi-step intrusions which require near real time alert correlation, this approach fails miserably due to the growing size of the alert index. One solution to this problem is to maintain a sliding window of previously received alerts close enough to the new alert. But an attacker can defeat this method either by passively delaying the second step or actively introducing new alerts between two steps.

Noel[14], *et al.* first reported use of attack graphs in minimising the effect of false alarms by correlating isolated intrusion alerts as part of multistep attack paths. Their alert correlation method is based on the shortest distance between exploits in the attack graph. Also, any IDS alert which does not feature in the possible future activities of the attacker (as can be observed in the attack graph) can readily be classified as false. However this solution is based on the assumption that the attack graph is updated in a timely fashion corresponding to changes in network topology and configuration. Also, the entire attack graph should be resident in memory for this analysis, which is a severe restriction for large enterprise networks.

Wang[30], *et al.* proposed a queue graph based approach for removing the limitations of the nested loop approach. The queue graph data structure only keeps in memory the latest alert matching each of the known exploits.

TVA attack graph has been used for planning optimal placement of IDS sensors[15] against all possible attacks. In this technique, isolated intrusion alerts are mapped to known exploits (represented as nodes) in an attack graph. It enables correlation of alerts corresponding to a multi-step attack scenario, and also prioritisation of alerts based on distance from critical network assets. Further, using the knowledge of network vulnerability paths encoded in an attack graph, network administrators can formulate best possible options for responding to attacks.

3.2 Minimum Cost Network Hardening

An attack graph reveals the different ways network resources can be compromised, but it does not provide any direct solution to harden the network. One of the network hardening measure is to remove or patch vulnerabilities. A good network hardening approach should remove specific vulnerabilities so that none of the attack paths leading to given critical resources can be realised, and also the cost involved in removing those vulnerabilities is minimum[13]. Jha[8] , *et al.* have presented a technique which allows analysts to determine minimal set of security measures (i.e. removal of attacks) that would guarantee safety of the system. They have provided a formal characterisation of the problem and have proved that it is polynomially equivalent to the minimum hitting set problem. However these solutions suffer from implementation issues as some of the vulnerabilities are consequences of exploiting other vulnerabilities and the consequences cannot be removed without first removing the causes. So, for removing a single vulnerability there may be multiple choices with different costs, considering the different sets of vulnerabilities it may implicitly depend upon. A different approach of minimum cost of network hardening was presented by Noel[13], *et al.* and improved by Wang[29], *et al.* The authors considered each vulnerability as a Boolean variable and derived a logical statement for the negation of given critical statement, in terms of initial conditions. If this logical statement is represented in disjunctive normal form (DNF), then each of the disjunctions in it provides a different hardening option. The option with minimum cost is chosen as the network hardening solution.

3.3 Network Forensics

Forensic analysis is typically performed after an incidence of break-in occurs. Its objective is to find attacker's probable actions, to assess damage and to collect digital evidence in case legal action is required. Clever attackers use anti-forensics techniques and tools to prevent proper forensic investigation. These techniques aim at reducing quality and quantity of digital evidences or traceable information captured by different tools.

Attack graph based forensic analysis enables administrators to prove that a series of IDS alerts are not isolated; rather they correspond to a sequence of attacks in a coherent attack plan. Liu[12], *et al.* proposed a solution where they have augmented attack graphs with anti-forensic activity nodes that help in identifying missing evidences.

3.4 Attack Graph Based Security Metrics

To improve the security of a network, administrators must be able to measure the same. A security metric measures or assesses the extent to which a system meets its security objectives. Using suitable security metrics one can measure how secure a network currently is, and how secure it would be after introducing new security mechanisms or configuration changes. This is necessary if a network has to evolve through network hardening.

An attack graph shows all possible attack paths that an attacker may follow to achieve her goal. However, in general, it does not provide any solution to harden the network. Traditionally, network hardening involves removal or patching of vulnerabilities, thereby preventing attackers from exploiting them to compromise critical hosts. However, it is not always possible to remove all vulnerabilities in a network setting. This may be due to high cost involved or owing to operational constraints. So, there is a need to have good security metrics which enable comparison of relative security of different network configuration options. A number of security metrics have been proposed in literature based on attack graphs.

Network compromise percentage (NCP) security metric indicates the percentage of network hosts where the attacker has obtained user or superuser privilege. Asset values can be associated with individual hosts before computing the NCP metric. This metric was proposed by Lippmann[11], *et al.* Computation of this metric may require traversal of the entire attack graph.

Weakest adversary security metric[18] measures the security strength of a network in terms of the strength of the weakest adversary who can successfully penetrate the network. It computes the minimal set of initial conditions the weakest adversary should satisfy in order to compromise the network.

Idika and Bhargava[4] have presented a solution which overcomes the shortcomings of existing attack graph based security metrics, i.e. Shortest path metric, number of paths metric, and mean of path lengths metric. Their approach combines these existing metrics with other metrics, to overcome their shortcomings.

Wang [31], *et al.* proposed attack resistance security metric to measure the attack resistance of a network configuration in terms of measures of individual exploits. The authors introduced the notion of exploit dependency graph which shows different possible sequences of exploits that an attacker can execute to compromise critical hosts in a network setting. The dependency relation between the exploits may be either conjunctive or disjunctive. They also introduced two types of composition operators corresponding to disjunctive and conjunctive dependency relationships between individual exploits to compute the overall security measure, i.e. the attack resistance of the network configuration. Authors used a function to capture the information about how execution of one exploit may affect the resistance value of another exploit. This is helpful in cases where two or more exploits involve the same vulnerability. Such exploits are related by this function to indicate the fact that successful exploitation of one instance of the vulnerability should reduce the resistance of the others due to the attacker's accumulated experiences and tools.

Zero day vulnerabilities are those for which there is no prior knowledge or experience. The *k*-zero day safety[34] security metric is based on the number of unknown zero day vulnerabilities. A network is said to be *k*-zero day safe if at least *k* unknown vulnerabilities are required for compromising a network asset, regardless of types of those vulnerabilities. A higher value of *k* indicates a relatively secure network. The authors have introduced the idea of zero day attack graph for computing zero day safety of a network.

Noel and Jajodia[16] proposed a suite of security metrics based on attack graph of a network, for measuring overall security risk. The metrics are grouped into families which are then combined into a single score. Single risk score is often beneficial for network administrators in situations where they have to interpret multiple scores. The different families of security metrics are

(i) Victimisation: scores network services and their vulnerabilities,
(ii) Size: measures risk in terms of the attack graph size,
(iii) Containment: measures security risk in terms of the degree with which the attack graph contains attacks across different network protection domains such as different subnets, and
(iv) Topology: based on graph theoretic properties of the attack graph such as the weakly connected components, strongly connected components, length of maximum shortest path etc.

The construction of an attack graph is based on the assumption that, a vulnerability can always be exploited. But, in reality, there is a wide range of probabilities associated with exploitability of vulnerabilities. This probability is dependent on the skill of the attacker and the difficulty of the exploit. Attack graphs show only what is possible without any indication of what is likely.

Attack graph based probabilistic security metric[32,26] approach uses common vulnerability scoring system (CVSS) values for individual exploits and computes a cumulative score considering the causal relationship among exploits and security conditions.

Wang[33], *et al.* presented an approach which integrates attack graph and Hidden Markov model (HMM) together, for exploring the probabilistic relation between system observations and states. They have used a modified version of dependency attack graph to represent network assets and vulnerabilities. This is then fed to HMM for estimating attack states, whereas their transitions are driven by a set of predefined cost factors associated with potential attacks and countermeasures. A heuristic searching algorithm is employed to automatically infer the optimal security hardening through cost-benefit analysis.

4. CONCLUSION

Attack graph is a useful abstraction of security state of a network, enabling automated solutions for reasoning about the same. This paper has given a short review of attack graph generation and analysis techniques reported so far in literature. Some of the research challenges in attack graph based network security analysis are.

- Attack graph generation solutions require input information which is captured using different software tools; but, this is not a completely automated process. Information about different vulnerabilities, i.e. their pre- and post-conditions are still manually encoded by domain experts. This is due to the fact that, public information about vulnerabilities are mostly available in the form of unstructured or semi-structured texts, rendering their automated extraction difficult. Natural language processing (NLP) techniques can be used for this purpose.
- Scalability issue in attack graph generation for moderate and large enterprise networks is still a challenging task. Big data framework for large graph processing is a promising solution towards achieving this goal.
- As far as the different attack graph based analysis techniques are concerned, use of attack graph in network forensics analysis is a relatively unexplored area.
- For forecasting possible future attacks, existing attack graph based techniques correlates intrusion alerts. Possible future attack paths from that point onwards forms a prediction. But in practice, many such attack paths are never tried by attackers. This lack of precision can be overcome by considering more context information such as intrusion response actions, exploit probabilities etc.
- Also, many of the analysis techniques are tied with a particular representation of attack graph. There is a need of uniform semantics of attacks, so that analysis techniques can be applied irrespective of the underlying representation and generation method.

REFERENCES

1. Ammann, P.; Wijesekera, D. & Kaushik, S. Scalable, graph-based network vulnerability analysis. *In* Proceedings of the 9th ACM Conference on Computer and Communications Security, 2002. pp. 217-224.
2. Ammann, P.; Pamula, J.; Ritchey, R. & Street, J. A host-based approach to network attack chaining analysis. *In* 21st Annual Computer Security Applications Conference, Tucson, AZ, 2005. doi: 10.1109/CSAC.2005.6
3. Dacier, M. & Deswarte, Y. Privilege graph: An extension to the typed access matrix model. *In* Proceedings of the 3rd European Symposium on Research in Computer Security (ESORICS '94), D. Gollman, ed., Brighton, U.K., Lecture Notes in Computer Science 875, Springer-Verlag, 1994. pp. 317-334.
4. Idika, N. & Bhargava, B. Extending attack graph-based security metrics and aggregating their application. *IEEE Trans. Dependable Secure Comput.*, 2012, **9**(1), 75-85. doi: 10.1109/TDSC.2010.61
5. Ingols, K.; Lippmann, R. & Piwowarski, K. Practical attack graph generation for network defense. *In* ACSAC '06: Proceedings of the 22nd Annual Computer Security Applications Conference. Washington, DC, USA: IEEE Computer Society, 2006. pp. 121–130.
6. Jajodia, S.; Noel, S. & O'Berry, B. Topological analysis of network attack vulnerability. *In* Managing cyber threats: Issues, approaches and challenges. *Edited by* V. Kumar, J. Srivastava, A. Lazarevic. Kluwer Academic Publisher, 2003.
7. Jajodia, S. & Noel, S. Topological vulnerability analysis: A powerful new approach for network attack prevention, detection, and response. *In* ASIACCS '07: Proceedings of the 2nd ACM symposium on Information, computer and communications security, 2007.
8. Jha, S.; Sheyner, O. & Wing, J. Two formal analyses of attack graphs. *In* Proceedings of the 15th IEEE workshop on Computer Security Foundations (CSFW '02). IEEE Computer Society, Washington, DC, USA, 49-63. doi: 10.1109/CSFW.2002.1021806
9. Lippmann, R. & Ingols, K. An annotated review of past papers on attack graphs. MIT Lincoln laboratory Project Report, 31 March 2005. Project Report ECS-TR-2005-054.
10. Lippmann, R. & Ingols, K. Evaluating and strengthening enterprise network security using attach graphs. MIT Lincoln laboratory, August 2005. Project Report Number,
11. Lippmann, R.; Ingols, K.; Scott, C.; Piwowarski, K.; Kratkiewicz, K.; Artz, M. & Cunningham, R. Validating and restoring defense in depth using attack graphs. *In* IEEE Military Communications Conference, 23-25 Oct. 2006. MILCOM 2006. pp.1-10. doi: 10.1109/MILCOM.2006.302434.
12. Liu, C.; Singhal, A. & Wijesekera, D. Using attack graphs in forensic examinations. *In* 7th International Conference on Availability, Reliability and Security (ARES), 20-24 Aug. 2012, pp. 596-603. doi: 10.1109/ARES.2012.58
13. Noel, S.; Jajodia, S.; O'Berry, B. & Jacobs, M. Efficient minimum-cost network hardening via exploit dependency graphs. *In* Proceedings of 19th Annual Computer Security Applications Conference, 8-12 Dec. 2003. pp. 86-95. doi: 10.1109/CSAC.2003.1254313
14. Noel, S.; Robertson, E. & Jajodia, S. Correlating intrusion events and building attack scenarios through attack graph distances. *In* 20th Annual Computer Security Applications Conference, 2004, pp.350-359, 6-10 Dec. 2004. doi: 10.1109/CSAC.2004.11
15. Noel, S. & Jajodia. S. Optimal IDS sensor placement and alert prioritization using attack graphs. *J. Network Syst. Manag.*, 2008, **16**(3), 259-275. doi: 10.1007/s10922-008-9109-x
16. Noel, S. & Jajodia, S. Metrics suite for network attack graph analytics. *In* Proceedings of the 9th Annual Cyber and Information Security Research Conference (CISR 2014), Robert K. Abercrombie and J. Todd McDonald (Eds.). ACM, New York, NY, USA, 5-8. doi:10.1145/2602087.2602117
17. Ou, X.; Boyer, W.F. & McQueen, M.A. A scalable approach to attack graph generation. *In* Proceedings of the 13th ACM conference on Computer and communications security. New York, NY, USA: ACM, 2006. pp. 336-345.
18. Pamula, J.; Jajodia, S.; Ammann, P. & Swarup, V. A weakest-adversary security metric for network configuration security analysis. *In* Proceedings of the 2nd ACM workshop on Quality of protection (QoP 2006). ACM, New York, NY, USA, 31-38.

doi: 10.1145/1179494.1179502

19. Phillips, C. & Swiler, L.P. A graph-based system for network-vulnerability analysis. *In* Proceedings of the 1998 workshop on New Security Paradigms (NSPW '98). ACM, New York, NY, USA, 71-79. doi: 10.1145/310889.310919
20. Ritchey, R.W. & Ammann, P. Using model checking to analyze network vulnerabilities. *In* Proceedings of the IEEE Symposium on Security and Privacy. Washington, DC, USA: IEEE Computer Society, 2000, pp. 156-165. doi: 10.1109/SECPRI.2000.848453
21. Sawilla, R.E. & Ou, X. Identifying critical attack assets in dependency attack graphs. *In* Proceedings of the 13th European Symposium on Research in Computer Security: Computer Security (ESORICS '08), Sushil Jajodia and Javier Lopez (Eds.). Springer-Verlag, Berlin, Heidelberg, 18-34. doi: 10.1007/978-3-540-88313-5_2
22. Schneier, B. & Secrets, Lies: Digital Security in a Networked World. Chapter 21. John Wiley & Sons, 2000.
23. Sheyner, O.; Haines, J.; Jha, S.; Lippmann, R. & Wing, J., M. Automated generation and analysis of attack graphs. *In* Proceedings of the 2002 IEEE Symposium on Security and Privacy. Washington, DC, USA: IEEE Computer Society, 2002. pp. 273-284. doi: 10.1109/SECPRI.2002.1004377
24. Sheyner, O. & Wing, J. Tools for generating and analyzing attack graphs. *In* Proceedings of Formal Methods for Components and Objects. Springer Verlag, 2004. pp. 344-371.
25. Sheyner, O.M. Scenario graphs and attack graphs. Carnegie Mellon Univ., Pittsburgh, PA, USA. 2004. (Ph.D. Thesis)
26. Singhal, A. & Ou, X. Security risk analysis of enterprise networks using probabilistic attack graphs. NIST Inter Agency Report 7788, 2011.
27. Swiler, L.P.; Phillips, C. & Gaylor, T. A graph-based network-vulnerability analysis system. *In* Sandia National Laboratories, Albuquerque, New. ACM Press, 1998, pp. 97-110. doi: 10.2172/573291
28. Swiler, L.P.; Phillips, C.; Ellis, D. & Chakerian, S. Computer attack graph generation tool. *In* Proceedings of the DARPA Information Survivability Conference and Exposition II, June 2001. doi: 10.1109/DISCEX.2001.932182
29. Wang, L.; Noel, S. & Jajodia, S. Minimum-cost network hardening using attack graphs. *J. Comp. Comm.*, 2006, **29**(18), 3812-3824. doi: 10.1016/j.comcom.2006.06.018
30. Wang, L.; Liu, A. & Jajodia. S. Using attack graphs for correlating, hypothesizing, and predicting intrusion alerts. *J. Comp. Comm.*, 2006, **29**(15), 2917-2933. doi: 10.1016/j.comcom.2006.04.001
31. Wang, L; Singhal, A. & Jajodia, S. Measuring the overall security of network configurations using attack graphs. *In* Proceedings of the 21st annual IFIP WG 11.3 working conference on Data and applications security. *Edited by* Steve Barker and Gail-Joon Ahn. Springer-Verlag, Berlin, Heidelberg, 98-112. 2007.
32. Wang, L; Islam, T.; Long, T.; Singhal, A. & Jajodia, S. An attack graph-based probabilistic security metric. *In* Proceeedings of the 22nd annual IFIP WG 11.3 working conference on Data and Applications Security, *Edited by* Vijay Atluri. Springer-Verlag, Berlin, Heidelberg, 283-296. 2008. doi: 10.1007/978-3-540-70567-3_22
33. Wang, S.; Zhang, Z. & Kadobayashi, Y. Exploring attack graph for cost-benefit security hardening: A probabilistic approach. *Computer Security*, 2013, **32**, 158-169. doi: 10.1016/j.cose.2012.09.013
34. Wang, L.; Jajodia, S.; Singhal, A.; Cheng, P. & Noel, S. k-Zero day safety: A network security metric for measuring the risk of unknown vulnerabilities. *IEEE Trans. Dependable Secure Comput.*, 2014, **11**(1), 30-44. doi: 10.1109/TDSC.2013.24
35. XSB, http://xsb.sourceforge.net/ [Accessed on 18th July 2016].

CONTRIBUTORS

Mr Mridul Sankar Barik is currently serving as Assistant Professor in the Department of Computer Science and Engineering, Jadavpur University. His research interests include distributed computing, network security, digital forensics, IoT etc.

Mr Anirban Sengupta is presently working as Principal Research Engineer in the Centre for Distributed Computing, Jadavpur University, Kolkata. His research interests include enterprise information security modelling, risk management and compliance.

Mr Chandan Mazumdar is currently serving as Professor in the Department of Computer Science & Engineering, Jadavpur University. His research interests include distributed computing, information and systems security etc.