An Atomic-Domains-Based Approach for Attack Graph Generation

Fangfang Chen, Chunlu Wang, Zhihong Tian, Shuyuan Jin, Tianle Zhang

Abstract—Attack graph is an integral part of modeling the overview of network security. System administrators use attack graphs to determine how vulnerable their systems are and to determine what security measures to deploy to defend their systems. Previous methods on AGG (attack graphs generation) are aiming at the whole network, which makes the process of AGG complex and non-scalable. In this paper, we propose a new approach which is simple and scalable to AGG by decomposing the whole network into atomic domains. Each atomic domain represents a host with a specific privilege. Then the process for AGG is achieved by communications among all the atomic domains. Our approach simplifies the process of design for the whole network, and can give the attack graphs including each attack path for each host, and when the network changes we just carry on the operations of corresponding atomic domains which makes the process of AGG scalable.

Keywords—atomic domain, vulnerability, attack graphs, generation, computer security

I. INTRODUCTION

As networks of hosts and security incidents continue to grow, it becomes increasingly more important to automate the process of evaluating network security. Attack graphs are composed by all the attack paths which lead to intruders intentions. The attack path is formed by a chain of exploits, where each exploit is realized by taking advantage of known vulnerabilities in various of services and systems. The term vulnerabilities refers to exploitable errors in Configurations and server software implemented to provide network services. Essentially, attack graphs represent the security state of network, and can serve as an useful tool in several areas of network security, including intrusion detection, defense, and forensic analysis[8]. On the one hand, system administrators can use attack graphs to collect information about their system's security state[9][18][23]. On the other hand, minimal critical sets of vulnerabilities and key hosts or vulnerabilities can be computed by attack graphs[7]. Then measures can be made to strengthen the network security. This active defense can achieve better effect than the passive defense which is achieved by collecting succeed attacks.

Supported by the National High-Tech Research and Development Plan of China under Grant Nos. 2009AA01Z438, 2009AA01Z431; the National Natural Science Foundation of China under Grant No.60703021
Fangfang Chen is with the Department of Computer Science and Technology, Beijing University of Posts and Telecommunications, Beijing, 100876, China e-mail: lulynn_2008@163.com.
Chunlu Wang,1969, Beijing University of Posts and Telecommunications. Associate Professor, Computer Architecture and Computer Security, wangcl@bupt.edu.cn
Zhihong Tian, Ph.D. Computer Network and Computer Security, tianzhihong@software.ict.ac.cn
Shuyuan Jin, Ph.D. Computer security, jinshuyuan@software.ict.ac.cn
Tianle Zhang, Ph.D. Computer Network, ftzhang@bupt.edu.cn

Most of methods for AGG are based on privilege promotion [1][6][7][8] and information used to describe pre- and post-conditions for exploitor attack components must be entered by hand. This is labor intensive and difficult [20]. Now, the Common Vulnerability Scoring System (CVSS) provides an open framework for communicating the characteristics and impacts of IT vulnerabilities[17]. There are many descriptions for vulnerabilities pre- and post-conditions. But for generating attack graph, there is an essential characteristic called relationship between pre- and post-conditions, then a vulnerability post-conditions can be part of another vulnerabilities pre-conditions. Although some other vulnerability database also provide information relation to pre- and post-conditions, they do not contain the machine-readable details required to accurately produce many of the attack graphs shown in past papers [20]. The lack of study on vulnerability relatedness has been the bottleneck in attack graphs generation, analysis and usability.

The generation of attack graphs has experienced two stages including manual generation and automatic generation[7][8]. In automatic generation, model checking owns the most widespread application and approval[1][6][22]. (Such as the usage of model checking tool called NuSMV[7][8][9][22]) Besides, logic programming is also demonstrated to be an effective method owing to attack graphs automatic generation[4]. But for practical application, there are several shortcomings as follows in methods above:

1. Non-automated: For model checking, there is no transformation tool for the networks automatic modeling[7][8][9][22]. For modern complex and large-scale network, this manual network modeling is labor intensive, difficult, and error-prone.
2. Incomplete: In model checking, the attack graphs are aiming at some host[7][8]. In fact, the administrator is not sure about which host is needed to be protected, so it is essential to generate the attack graphs for the whole network.
3. Complex: All the present methods for attack graphs generation including model checking and logic programming are aiming at the whole network during operation. It is quite complex to deal with the information of the whole network including all the hosts, network topology information and vulnerabilities.
4. Non-scalable: Using the present methods, the scalable generation is not achieved. When information of network are changed (such as some hosts are added or removed) then processes for generation are again.
5. Impractical: Almost all the network topology discovery tools can just achieve the getting of topology information concerning some specific host[19][20]. This makes the process of the whole network information first generation second
unrealistic. The method of this paper provides away by which the attack graph can be generated without the whole network topology information. We just need the realization of getting topology information concerning some host.

This paper describes a new approach to attack graphs generation which is concern about the information design of each host, but not the whole network. This method is essentially a re-distribution of the information of network. Our method has the advantages as follows:

1. Simplify the process of attack graphs generation: Comparison with the original methods which is dealing with the whole information of network, we are aiming at information of each host. Besides, the design for each host is in the same template.

2. Gain the attack graphs for all the hosts of the target network: When analyzing the security of an enterprise network, it is important to consider multi-host attacks [4]. At present, there are hand generated [15], programming with some programming languages [12][13][14] and so on. But, the methods above have poor scalability and are limited in practical applications [20]. Our approach has the ability to multiple goals attack graph generation and has good scaling.

3. Achieve the generation of attack graphs directly without concerning the attack path: Previous work on attack graphs generation all followed the order: attack paths first, then attack graphs [9][7][20]. Our method can achieve the goal without the information of attack paths. And if necessary, attack paths can be obtained by analyzing attackgraphs.

II. RELATED WORK

Philips and Swiler [17] propose the concept of attack graph and describe a tool [21] for attack graphs generation. After that, a variety of approaches to generating various forms of attack graphs appear, including custom-design graph-based [10][11], logical deduction [4], and model checking [1][6][7]. In model checking method, the attack paths are given as counter-examples which are related with the security policy given by network administrators. NuSMV is the improvement of SMV and all the counter-examples that violate the safety policy can be achieved. In the prior network models, the counter-examples are attack paths corresponding in the safety policy that the intruder can not earn some privilege on some host [1][8]. By integrating all the attack paths, the attack graph is generated [8][16][22]. Different from priorwork, our approach aims at generating the attack graph for the whole network. In other words, we integrate all the attack paths for all the hosts in the target network. Past attack graphs are subsets of our attack graph and attack graphs for each host can be earned by reverse depth-first search from target host. There are various of attack graphs with different kinds of nodes or edges. In paper [8][21], the nodes in their attack graphs represent the state of the network, and the edges represent an attacker actions that change the state. While in [10], their nodes of attack graphs represent an host and the edges represent the vulnerability used to attack next host. Taking into account the complexity of attack graphs, the former attack graphs have too many nodes, while the latter one has too many edges on each nodes. In paper [1], the authors give one design which is the compromising position of the two illustrations above. In the initial stage, there is no limiting factors and the attack graph is complete that means all the attack paths are included. Then the significant exponential explosion problem happens. Ammann, et al. pointed out that for most computer attacks, one can assume the monotonicity property, where an attacker does not decrease his ability by launching attacks, and hence does need to relinquish privileges he already gained. Under this assumption, an attackers privileges always increase during the analysis. Since there are only a polynomial number of privileges an attacker can gain, the analysis algorithm will terminate in polynomial time [11]. Our approach achieve this by designing of atomic domains. In paper [22], the authors use an efficient semantic evaluation program in the MuVAL reasoning engine to generate the logical attack paths. They got better running time than Ammanns. After that, Rattikorn Hewett and Phongphun Kijsanayothin got even better running time than paper [22] by host-centric model checking [1]. This paper achieve even better improvement than [1] by eliminating unnecessary internal attacks in some host.

III. ATTACK MODEL

A. Definitions

A network attack model is an attack model where all informations concerning network including network topology, hosts, vulnerabilities and so on are organized to be a finite state machine whose state transitions are based on intruders attack actions [11][17][22]. Former approaches for attack graphs generation are aimed at the whole network [1][3][6][7][8][9]. In this paper, our approach divided the network modeling into design of each host. The generation of attack graphs can be achieved by communications between atomic domains. Here are some definitions for our system:

Definition 1. $Vul = \langle ID, pre, post, h, s \rangle$ is a five-tupled definition for a vulnerability. Each vulnerability has five properties as follows:
1. $Vul.ID$, the CVE standard vulnerability name [32]
2. $Vul.pre$, intruder pre-condition
3. $Vul.post$, intruder post-condition
4. $Vul.h$, the host owning the current vulnerability
5. $Vul.s$, a boolean and when valued 1(true), it means the intruder has been succeeded in capitalizing on the current vulnerability.

Definition 2. An $AD_h^n = \langle h, n, VN, VA, t \rangle$ is an atomic domain for host $h$ with privilege $n$. The following is a list of the components for each $AD_h^n$:
1. $h$, a host with the name $h$
2. $n$, the privilege for $h$
3. $VN$, $VA$, $t$ are three values for $n \in (0, 1, 2)$ representing ( none, user, root ).
3. \( \text{VN (Vulnerabilities Needed)} \), a set of vulnerabilities owning to host \( h \) or hosts which are connected to host \( h \) directly and making it possible to achieve privilege \( n \) on host \( h \) in an exploit.

4. \( \text{VA (Vulnerabilities Available)} \), a set of vulnerabilities owning to host \( h \) or hosts which are connected to host \( h \) directly and being available to use when intruder achieves privilege \( n \) on host \( h \).

5. \( t \), a boolean variable. When valued \( 1 \) (true), it means the intruder has achieved the privilege \( n \) on host \( h \). Then attacks concerning the set of vulnerabilities \( VA \) can be started.

**Definition 3.** An attack is achieved by communication between atomic domains and the communication is achieved by two actionss as follows:

1. \( AD^h_n, s(vul) \): sending action performed by \( AD^h_n \) whose vulnerabilities set \( VA \) owns \( vul \).
2. \( AD^h_n, a(vul) \): receiving action performed by \( AD^h_n \) whose vulnerabilities set \( VN \) owns \( vul \).

The two above are booleans and mean the success of exploit concerning \( vul \) when both valued \( 1 \) (true).

**Definition 4.** Success label: we associate a boolean function for each vulnerability, abstractly representing whether or not the current vulnerability is used successfully. For the vulnerability \( Vul \), we define the function \( a^2(Vul) \).

**Definition 5.** In each attack path such as \((s_1, s_2, \ldots, s_n)\) where \( s_i \) indicates an \( AD^h_n \) intruded during attacking, \((s_i, s_{i+1})(1 \leq i \leq n)\) is called \( AA \) (atomic attack). \( AA \) has two elements as follows:

1. \( ss \) (single source), the attack sponsor in an \( AA \).
   
   If \( (AA = (s_1, s_{i+1})) \)
   
   Then \( AA.s.s = s_i \).
2. \( st \) (single target), the attack goal in an \( AA \).
   
   If \( (AA = (s_1, s_{i+1})) \)
   
   Then \( AA.s.t = s_{i+1} \).

**Definition 6.** \( NAD \) (Neighbor Atomic Domains) of host \( h \) is a set of \( AD^h_n \), in which the \( AD^h_n \) owning to the host which is reachable to the current host \( h \) directly.

**Definition 7.** An attack model is a finite automaton \( M = (AD^h_n, a^2s) \), where \( AD^h_n \) is a set of all the \( AD^h_n \) owning to the network and \( a^2s \) is a set of success labels for all vulnerabilities owning to the network. The value changes of the set \( a^2s \) mean the state transformationsof the finite automaton.

**B. Modeling**

The process for network modeling is as follows:

1. To store the hosts newly discovered each time, we create a queue called \( Q \).
2. Using network topology discovery tool, we get the set \( C \) of hosts which are reachable directly to the attacker and store the hosts of \( C \) into the queue \( Q \). Then the vulnerabilities information of hosts in \( C \) can be got by the vulnerability scanning tool Nessus. Finally, the initialization of \( AD^h_n (\text{atomic domain}) \) concerning attacker can be achieved.

3. Take a host from \( Q \) and go with the following steps:
   1) Using the network topology discovery tool, we get the set \( C \) of hosts which are reachable directly to the current host.
   2) Queue the hosts newly discovered in (1) into \( Q \).
   3) Get the vulnerabilities information of each host from \( C \).
   4) Initialize the \( AD^h_n (\text{atomic domain}) \) concerning the current host.

4. Check whether the queue \( Q \) is empty. If \( Q \) is empty, the modeling for network is achieved. Otherwise, go back to 3.

1) Parameter Settings: \( n \) In accordance with the definition in 2.1, for the target network with \( n \) hosts, there should be \( 3(n+1) AD^h_n \) (including three \( AD^h_n \) concerning the intruder). But for most computer attacks, an attacker follows the monotonicity property, where he dose not decrease his ability by launching attacks and will focus on privilege promotion [4]. This property can be valuable to the complexity reduction of attack graphs. During setting of \( AD^h_n \), we achieve monotonicity property by rules as following:

1. In the initial state of the network, as the intruder owns the privilege of \( h \) administer on its own, it is not needed to notice the other two lower privileges. Then only one atomic domain is necessary for the host owned by the attacker: \( AD^2_h \).
2. In the initial state of the network, as the intruder owns at lest the privilege of none on each host of target network, there is no necessary to care about the privileges of none for each host. The intruder can achieve the privilege of none on each host without attacks. Then only the \( AD^h_n \) concerning the privileges of user and administer on each host are necessary. Above all, \( (2n+1) \) but not \( 3(n+1) AD^h_n \) are necessary in our design. We name the hosts of the target network as \((1, 2, \ldots, n)\). Then the set \( AD^h_n \) owning the attack model \( M \) represents a set of \((2n+1)\) agents \( AD^h_n = (AD^h_{n,1}, AD^h_{n,2}, AD^h_{n,3}, AD^h_{n,4}, \ldots, AD^h_{n,2n+1}). \)
   In which \( (AD^h_{n,1}, AD^h_{n,2})(1 \leq i \leq n) \) are the \( AD^h_n \) concerning the host \( i \).

2) Initialization of \( AD^h_n \): \( n \) For atomic domain \( AD^h_n \), the initialization of the properties \((AD^h_{n,VN}, AD^h_{n,VA}, AD^h_{n,t})\) concerning \( AD^h_n \) is as follows:

1. \( AD^h_{n,VN} \): The attack towards host \( h \) can be achieved just by exploiting concerning its own vulnerabilities. Then elements of \( AD^h_{n,VN} \) are from host \( h \) and another condition should be: \( Vul.post = n \).
2. \( AD^h_{n,VA} \): The vulnerabilities available for \( AD^h_{n} \) can either come from host \( h \) or hosts which are reachable to \( h \) directly. Then the setting of \( AD^h_{n,VA} \) is as follows:
   1. \( AD^h_{n,VA} \): The vulnerabilities owning host \( h \), the one which meets the rule: \( Vul.pre = n \) and is not included in \( AD^h_{n,VN} \) can be included in \( AD^h_{n,VA} \). In our design, we prevent the redundant attacks by making there is no common elements in \( AD^h_{n,VA} \) and \( AD^h_{n,VA} \).
   2. For the vulnerabilities owning hosts connected physically with host \( h \), the one which meets the rule: \( Vul.pre \leq n \) can be included in \( AD^h_{n,VA} \). When gets higher privilege on a...
host, the intruder can achieve all the exploits concerning the vulnerabilities with lower pre-conditions.

3. **AD**<sub>n</sub>:t When existing one vulnerability *Vul* in the set **AD**<sub>n</sub>·*VN* which meets the rule: *Vul*·s = 1, then **AD**<sub>n</sub>:t can be valued 1.

Among all the pre-conditions needed for attack graph generation, network topology is an important part, while in prior work this part is handled as a single and complete part [1][6][7][8][9][10][16][22]. Then before generation attack graphs, the whole information about network topology is needed. As it is hard for the current network topology discovery tools to get the network topology, current methods for attack graphs generation are impractical for unknown network. In our approach, only the topology information concerning one host is needed and it is easy for topology discovery tools to achieve this. In our approach, we include the network topology information into the setting of **AD**<sub>n</sub>·*VN* and **AD**<sub>n</sub>·*VA*.

### C. Algorithm

After setting and initialization of **AD**<sub>n</sub>·*s* (atomic domains), the preparations for attack graphs generation have been achieved, then the process of generation can be achieved by communications between **AD**<sub>n</sub>·*s*.

Actually, our method gives the process of breadth-first penetration attacks to network. In our design, we achieve the lamination attack graph generation by the data structure queue. To the attack model *M* = (**AD**<sub>n</sub>·*s*, *a*<sup>2</sup>*s*), in which the set **AD**<sub>n</sub>·*s* represents a set of *2n* + 1 agents **AD**<sub>n</sub>·*s* = (**AD**<sub>1</sub>·*s*, **AD**<sub>2</sub>·*s*, **AD**<sub>3</sub>·*s*, **AD**<sub>4</sub>·*s*, ..., **AD**<sub>*n*</sub>·*s*, **AD**<sub>*n*</sub>·*s*), we can achieve the generation of attack graphs as the following:

1. To achieve the breadth-first traversal of the elements in the set **AD**<sub>n</sub>·*s*, we create a queue called *Q* to store the **AD**<sub>n</sub>·*s* need to be activated.
2. Started from **AD**<sub>2</sub>·*s*, the attacks concerning the vulnerabilities owning to **AD**<sub>2</sub>·*VA* can be started. Then the **AD**<sub>n</sub>·*s* being attacked are queued into *Q*.
3. Take one atomic domain from *Q* (named **AD**<sub>n</sub>), go with the following steps:
   1. Active **AD**<sub>n</sub>.
   2. Start the attacks concerning vulnerabilities owning to **AD**<sub>n</sub>·*VA*.
   3. Check the target **AD**<sub>n</sub>·*s* in (2), and queue the **AD**<sub>n</sub>·*s* which have not been activated into *Q*.
   4. Check whether the queue *Q* is empty. If *Q* is empty, the generation of attack graph is achieved. Otherwise, go back to 3.

### D. Consideration of Implementation

**SPIN** is a generic verification system that supports the design and verification of asynchronous process system. Spin verification are focused on proving the correctness of process interactions, and they can simulate the communications between processes. In our design, each atomic domain is corresponded to a process. Then the communications between ADs can be simulated and achieved. The modeling language for SPIN is Promela (Protocol/Meta Language). Fig. 1 shows the model for Promela.

In our design, we correspond the information of attack model to elements of Promela as shown in Table 1.

Table 2 shows the information for communication stored in channels: (**sAD, tAD, Vul**).

Once there exists one vulnerability in **AD**<sub>n</sub>·*VN* which is exploited by intruder, **AD**<sub>n</sub> is activated by valuing the property *t*. Then the attacks concerning vulnerabilities in the set **AD**<sub>n</sub>·*VA* can be started by **AD**<sub>n</sub>.

### IV. EXPERIMENT

Fig. 2 shows an example network used in paper[1]. The network contains two hosts: web server W and database server D. An attackers host locates in the external network. The packet forwarding or transmission is controlled by two firewalls: EF (external firewall) and IF (internal firewall). EF allows...
![Example Network](image)

### TABLE III

**SUMMARY OF EXPLOITS**

<table>
<thead>
<tr>
<th>Vulnerability/Trust</th>
<th>Victim host</th>
<th>Pre-con. On Attack host</th>
<th>Post-con. On Victim host</th>
<th>Exploit Mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>ap</td>
<td>W</td>
<td>Access ≥ 1</td>
<td>Access = 2</td>
<td>Remote</td>
</tr>
<tr>
<td>tns</td>
<td>D</td>
<td>Access ≥ 1</td>
<td>Access = 2</td>
<td>Remote</td>
</tr>
<tr>
<td>t1</td>
<td>W</td>
<td>Access ≥ 1</td>
<td>Access = 1</td>
<td>Remote</td>
</tr>
<tr>
<td>t2</td>
<td>D</td>
<td>Access = 1</td>
<td></td>
<td>Remote</td>
</tr>
</tbody>
</table>

### TABLE IV

**ELEMENTS OF EACH ATOMIC DOMAIN**

<table>
<thead>
<tr>
<th>ADhn</th>
<th>VN</th>
<th>VA</th>
<th>T</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADh2</td>
<td>ap</td>
<td>t1</td>
<td>1</td>
</tr>
<tr>
<td>ADh1</td>
<td>ap</td>
<td>tns, t2</td>
<td>0</td>
</tr>
<tr>
<td>ADw1</td>
<td>t2</td>
<td>ap, tns, t1</td>
<td>0</td>
</tr>
<tr>
<td>ADw2</td>
<td>tns</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

any packet from outside the network to be transmitted to W but IF allows only transmission from W to D but not vice versa as shown in Fig. 2. There are three hosts (Attackers host, Web Server and Databaseserver) and four Vulnerabilities (shown in Table 3) in the network.

### A. Modeling for Example Network

According to the design, we can achieve the modeling for network in Fig. 2 as follows:

1. Setting of $AD^n_h$s owning to attack model $M$: $AD^n_h = (AD^n_{h2}, AD^n_{w1}, AD^n_{w2}, AD^n_{d1}, AD^n_{d2})$; 
2. initialization of each atomic domain :Table IV
3. Encoding : this part is achieved by the language Promela.

### B. Attack Graph Generation and Analysis

By SPIN, we can get the attack graph as shown in Fig. 3, in which the processes (A2, W1, W2, D1, D2) are corresponded to $AD^n_h$s = $(AD^n_{h2}, AD^n_{w1}, AD^n_{w2}, AD^n_{d1}, AD^n_{d2})$. Different from prior study, our method achieved the generation of attack graph without knowing each attack path. We achieve the generation of attack graph first, then attack paths can be get by depth-first search to attack graph. As the goal of the study is the attack graph [1][6][7][8][16][22], it is not necessary to get attack paths.

Our method generate the complete attack graph which includes all the attack path concerning each host. This is also different from previous works which are aimed at a certain known host [1][6][7][8][16][22].

### V. COMPARISON

#### A. Applications

Consider the network attack graph shown in Fig. 4, which is called host-centric attack graph generated in paper [1]. Host-centric model checking achieved lower complexity than Network-centric and access graph [1]. But in Fig. 4, there appears exploits of privilege reduction in one hosts internal attacks (such as attack from (W, 2) to (W, 1)), which does not meet the rule of monotonicity property. In our methods, we avoid this kind of situation by definition and design of atomic domains.

Fig. 5 gives a new network which is achieved by adding two hosts on the network in Fig.2. The vulnerabilities for the...
new network are shown in Table 5.

According to the rules of ADs setting, the design for the atomic domains which are concerning about the two added hosts is shown in Table 6. And the attack graph for the new network is shown in Fig. 6.
REFERENCES


