Generating Attack Scenarios with Causal Relationship

Yu-Chin Cheng, Chien-Hung Chen, Chung-Chih Chiang, Jun-Wei Wang, Chi-Sung Laih

Department of Electrical Engineering
National Cheng Kung University,
Tainan, Taiwan, Republic of China
laihcs@eebox.ncku.edu.tw

ABSTRACT

With the incoming of information era, internet has been developed rapidly and offered more and more services. However, intrusions, viruses and worms follow with the grown of internet, spread widely all over the world within high speed network. Although many kinds of intrusion detection systems (IDSs) are developed, they have some disadvantages in that they focus on low-level attacks or anomalies, and raise alerts independently.

In this paper, we give a formal description about attack patterns, attack transition states and attack scenarios. We proposed the system architecture to generate an attack scenario database correctly and completely. We first classify and extract attack patterns, then, correlate attack patterns with pre/post conditions matching and. Moreover, the approach, *Attack Scenario Generation with Casual Relationship (ASGCR),* is proposed to build an attack scenario database Finally, we present the combination of our attack scenario database with security operation center (SOC) to implement the related components concerning alert integrations and correlations. It is shown that our method is better than CAML [4] since we can generate more attack scenarios effectively and correctly to help system managers to maintain network security.

Keywords

Attack scenario database, security operation center, alert correlation, attack pattern.

1. INTRODUCTION*

Network attacks are increased as the technologies of network communication grow tremendously. In order to defense these attacks, network security devices such as IDSs and IPSs were developed to detect and prevent our system. Other security tools such as firewalls and antivirus software are also becoming essential equipments. However, the alerts generated by these devices have high false positive rate and redundancies [4]. In general, most of the attacks are composed of multiple steps and it is a need to detect all of these steps together. Traditional IDSs face three major problems in dealing with these multi-step attacks:

1. IDSs focus on generating alerts independently and do not analyze or correlate them.
2. The system managers will not effectively analyze main attacks when great mounts of attacks or plan-set attacks occur.
3. IDS will generate alerts and report to manager after attacks.

According to these reasons, it is necessary to construct attack scenarios from alerts that are raised by detection sensors, e.g. IDS. In general, there are three strategies that are proposed so far to solve the above problems. The first strategy (e.g., Spice [8], the probabilistic alert correlation [10]) correlates alerts by using alerts' attributes generated by a number of detection sensors. Based on this strategy, the detail information of the alerts will be shown, most including timestamp, IP address, port number and attack type. The alerts can also be merged by the same attributes. Though they are effective for correlating some alerts, they could not fully discover the causal relationships between related alerts. The second strategy (e.g., LAMBDA [3], CAML [4] and the data mining research [5]) correlates alerts based on conditions. There are some pre-conditions that can make an attack to be successful. After an attack being made, it generates other results called post-conditions. These two conditions of produced alerts are related to each other. Although all attack scenarios are known to construct pre/post conditions, it further uses machine learning to collect unknown attack to be scenarios. The third strategy (e.g., JIGSAW [9]) is similar with the second strategy that is based on conditions and consequences. It correlates alerts if the pre-condition of some later alerts is satisfied by the consequences of some earlier alerts. So, conditions are the key points. In our approach, we concentrate on building an attack scenario database based on these conditions. In our experiment, we make a comparison with CAML and obtain a better result.

The rest of this paper is organized as follows. In Section 2 we introduce alert correlation based on pre/post conditions. Then, an approach to generate an attack scenarios is proposed in Section 3. In Section 4, we make

* The work is partially supported by National Science Council (NSC) of Taiwan under grants**NSC96-2628-E-006-021-MY3** and **NSC95-3114-P-001-002-Y02 (iCAST, I**nternational **C**ollaboration for **A**dvancing **S**ecurity **T**echnology)

0-7695-3032-X/07 $25.00 © 2007 IEEE
DOI 10.1109/GrC.2007.117

comparison with CAML and discuss the result. In Section 5, we apply our approach into SOC and characterize some conclusions in Section 6.

2. RELATED WORK

2.1 Alert Correlation

Alert correlation is a process that can be used to analyze the alerts produced by one or more intrusion detection systems. To our best knowledge, there are several alert correlation frameworks proposed in recent years. For example, Cuppens and Miege proposed an alert correlation framework [2]. Valdes and Skinner [10] built correlation by using mathematics method to calculate probabilistic and evaluating the similarities of alert attributes. In addition, Qin and Lee [7] proposed alert correlation algorithm that also used mathematics method based on a statistical research. Ning et al. [6], Cuppens and Miege [2], and Cheung et al. [1] built alert correlation systems based on the pre-conditions and post-conditions of individual alerts. A scenario language named LAMBDA [3] was also proposed by using this method. Steven, Martin and Fong proposed a pre/post conditions algorithm called CAML [4] and they used a modular research, where a module represents an inference step and modules could be linked together to detect multi-step scenarios.

Since our approach improves LAMBDA and CAML methods, we describe LAMBDA and CAML briefly as follows.

2.2 LAMBDA: A Language to Model a Database for Detection of Attacks

LAMBDA is a language that focuses on the attack relationship. The point of view of LAMBDA is based on intrusion that comes from an attacker. The language allows us to define the relationship among attacks. It is constructed by three components: State Description, Transition Description and Combining Events, as shown in Figure 2.1. State Descriptions describe all pre/post conditions of an attack. Every event is an object which represents the core of Transition Description and those objects are the collection of attributes. Finally, the task of Combining Events is using calculus algebra to combine several events. There is no database constructed in this method. Nevertheless, LAMBDA identifies attack scenarios in real time.

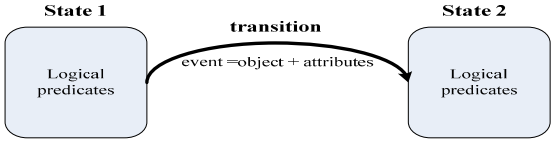

Figure 2.1. LAMBDA system module.

2.3 CAML: Correlated Attack Modeling Language

In 2003, Steven, Ulf and Martin proposed the Correlated Attack Modeling Language (CAML) which is focus on attack state and characterize the alert correlation based on pre/post conditions. It has a module which represents an inference step and correlates steps together according to its relationship to detect multi-step scenarios. The purpose of CAML is to predefine the condition of attack and could be used to construct a library of predicates which functions as a vocabulary to describe the properties of system states and events. Thus, CAML is used to describe the detailed, sensible condition and construct attack scenario. CAML constructs an attack scenario database in offline. Then it could easily recognize attacks in real time and produces reports that identify multi-step attack scenarios discovered in the alert stream.

In this paper, we present a formal description about attack patterns, pre/post conditions and attack scenarios. Then we proposed our method by using this description to generate attack scenarios with causal relationship.

3. GENERATING ATTACK SCENARIOS WITH CAUSAL RELATIONSHIP

We present our major techniques in this section. We start by introducing definitions such as attack patterns, attack transition states and attack scenarios in Section 3.1. Given a set of alerts or events information, we are interested in how our system can proceed to generate an attack scenario database more effectively. The technique is presented in Section 3.2. A two-phase method designed and defined to generate an attack scenario database is presented in Section 3.3 and 3.4, which focuses on discovering casual relationship of attack patterns.

3.1 Attack Patterns, Attack Transition States and Attack Scenarios

Various definitions of attack have been proposed over the course of decades of research. Attack pattern is an *action* of executing or utilizing tools by attackers to target a computer system. Stated another way, attack pattern involves the vulnerabilities used by attackers and the evidences to exist in targets. In this paper, we call them *Pre-condition* and *Post-condition*. Attack transition states, by definition, represent hacker's strategies of the attacking process. In general, there are several sequential states in the attack transition states. Among them, *Scan*, *Escalation*, *Remote* and *Local*, are the most discussed states and we only discuss them in this paper. Attack scenario is a scenario that describes the cascading patterns of attackers' action to perform the intrusion. A simple attack scenario can be formed by one or two attack patterns. A complex attack scenario is formed by many attack scenarios, that is, attackers execute many actions and tools in each of attack transition states.

For example, if an intruder wants to steal target information, he will scan the target first and try to find vulnerabilities. So, the attacking strategies might include network

surveillance steps. Besides, he can break into the target using a known vulnerability of system or service from the remote access. Then, the privilege escalation steps will be used to improve attackers' authority on the target. Finally, there are some purpose steps such as information theft or denial of some system service. In this example, we obtain four attack patterns.

3.2 System Architecture

In this subsection we describe the architecture of system, which will generate an attack scenarios database. As shown in Fig. 3.1, our system involves four main components. We collect many kinds of attacks found in networks and several operating systems. Because of different attack types, we cluster them into four types which are *scan*, *escalation, remote access* and *local access*. Then, we add *pre/post conditions* to each attack pattern according to its attacking type and details and correlate the attack patterns based on *pre/post conditions matching*. Finally, we have a generating scenario algorithm to correlate these patterns into attack scenarios and store them to a database called *attack scenario database*. The attack scenario database will be essential to understanding the security threats and taking appropriate actions.

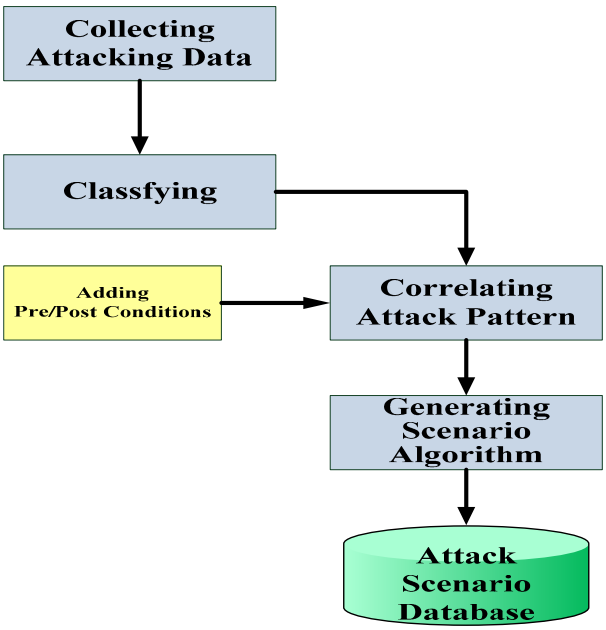

Figure 3.1. System Architecture

3.3 Correlating Attack Patterns with Pre/Post Conditions Matching

As mentioned above, how to correlate the attack patterns and how to improve the scenario generating is important after extracting and classifying the attack patterns. Our approach further determines the causal relationships between attack patterns .

Definition 1 Given an attack pattern *i* with the symbol P_i and let $\mathbf{P}$ be the set of all attack patterns.

An attack pattern ***i*** consists of *pre-conditions*, *attack pattern name*, *post-conditions*., shown in Figure 3.2

Figure 3.2. Attack Pattern *i* components

Definition 2 Given a pre-condition ***Rj***, defined as the condition that should be satisfied by computer system for an attack to be feasible. Let ***R*** be the set of all pre-conditions ***R= { r1, r2, r3, … , rm }***. Pre-conditions of the attack pattern ***i*** is defined as $R_i = \{r_{i1}, r_{i2}, ..., r_{ik}\}$, where $r_{ij} \in R$, $\forall 1 \le j \le k.$, Note that a pre-condition R_i of P_i is a subset of ***R*** since it may include some pre-conditions.

Definition 3 Given a post-condition **O*j***, defined as the effect of successful execution of an attack. Let ***POST*** be the set of all post-conditions ***O= {o1, o2, o3,…,on }***. Post-conditions of the attack pattern ***i*** is defined as $O_i = \{o_{i1}, o_{i2}, ..., o_{il}\}$, where $o_{ij} \in O$, $\forall 1 \le j \le l.$

Example 1 Here we give an example to explain our approach, related to correlating attack patterns with pre/post conditions matching. The target suffers from the RPC service attack and is compromised. Three phenomena can be observed. The first is that the target will open port 111 and port 2049. Secondly, RPC service will be started and the NFS will be mounted. Finally, attackers can access the target files through the RPC-NFS channel. Therefore, *open_port 111 and open_port 2049* are pre-conditions of the attack pattern P_i, named *Enumerate RPC Service-NFS.* The *RPC_service-NFS & mounted* are the post-condition of the *Enumerate RPC Service NFS* attack. Another attack patter P_j, called *Mount Public Directory,* has pre-condition *RPC_service-NFS & mounted.* The post-condition is to *access target files.* The two attack patterns can correlate using pre/post conditions matching, that is, finding the correlation among attack patterns with casual relationship. Figure 3.3 illustrates the correlation between attack patterns *Enumerate RPC Service-NFS* and *Mount Public Directory* with pre/post conditions.

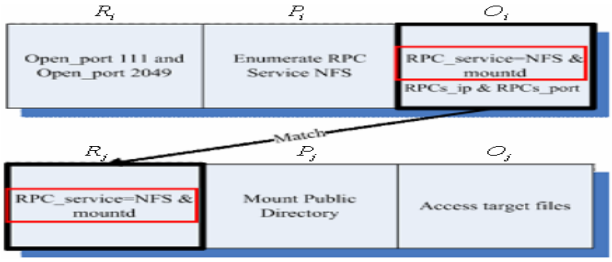

Figure 3.3. Pre/post conditions matching

These pre/post conditions are described by logical condition. Thus, these logical conditions handle the computer systems states that resemble potential targets of the attacks. We give **Definition 4** to describe how to correlate two attack patterns as following.

Definition 4 Two attack patterns P_i and P_j are *match* if $R_j \subseteq O_i$, and *unmatch* if $R_j \not\subseteq O_i$.

We use symbol φ and $\overline{\varphi}$ to represent match and unmatch. For example, $P_i \varphi P_j$ means P_i match P_j , and $P_i \overline{\varphi} P_j$ means P_i unmatch P_j .We call P_i is the ancestor of P_j if $P_i \varphi P_j$. On the contrary, P_j is the successor of P_i . As described in Example 1, how patterns P_i and P_j are matched with each other. P_i has post-condition that include NFS and mountd RPC_service. P_j also has the same RPC_service in pre-condition. According to our definition, P_i and P_j are match since $R_j \subseteq O_i$.

Definition 5 We define S_k as the symbol of attack scenario k, that is, $S_i = \{P_{i1}, P_{i2}, ..., P_{ik}\}$ if and only if $P_{i1} \varphi P_{i2} \varphi ... P_{ik}$ while $P_i \in \mathbf{P}$, $\forall P_{ij} \in \mathbf{P}$, $1 \le j \le k$.

An attack scenario may be composed of many attack patterns at each attack transition states. A simple attack scenario can be formed by one or two attack patterns at one state. A complex attack scenario is formed by many attack patterns at different transition states. Given two methods of *generating attack scenarios* M_1 and M_2. Method M_1 is called a *better method* if M_1 can generate more attack scenarios correctly and probably than M_2 by given a fixed attack patterns with corresponding pre/post conditions. Therefore, we can consider many attack patterns to be involved when an attack scenario is occurred.

3.4 Attack Scenario Generation with casual relationship

As mentioned above, our system can extract and correlate real attack patterns from various data. If we are able to capture these casual relationships between four phases, it may help us build step wise attack scenarios correctly. Attack Scenario Generation with Casual Relationship (ASGCR) is our algorithm to generate an attack scenario database using pre/post conditions matching. Our approach improves LAMBDA and CAML methods. We divide our algorithm into two methods:

1. *Ancestor-free method*, originally proposed by CAML.
2. *t-ancestor method*, we proposed this method by merging post-conditions of each attack patterns while LAMBDA is processing pre-conditions.

The first method, *ancestor-free*, is a method to correlate two or more attack patterns that is directly match at each other without considering the relationship of their ancestors. After finishing our works with this method, we continue to the next method. In general, there is a situation that an attack pattern P_k cannot be matched with $P_i \varphi P_j$ because $R_k \not\subseteq O_j$. However, P_k actually has relationship with the union of P_i and P_j . Thus we should consider P_i accomplishing a complete attack scenario. In order to obtain a complete attack scenario, we proposed a method called *t-ancestor generating attack scenarios with causal relationship*. We define t-ancestor match when $P_{t+1} \overline{\varphi} P_t$ if $P_1 \varphi P_2 ... \varphi P_t$ and $R_{t+1} \subseteq O_1 \cup O_2 ... \cup O_t$, where operator $\cup$ means *Union.*

4. COMPARISON AND DISCUSSION

To evaluate the effectiveness of our techniques, we performed experiments in Testbed@TWISC [11], which is the secure testbed for real security testing. We collect 75 attack patterns and classified them into *scan, escalation, remote access* and *local access* as shown in Table 4.1.

Table 4.1. Number of attack patterns at four stats.

Transitions state	Scan	Escalation	Remote	Local
Number	2	23	40	10

We use 75 attack patterns to generate various scenarios according to our algorithm and to compare with CAML.

Ancestor-free Method We correlate *scan* & *escalation* patterns into scenarios and get the same results of 71 attack scenarios both with our approach and CAML. In this phase, scan and escalation are directly matched because the pre-conditions of escalation state are included in scan state.

t-Ancestor Method After finishing the ancestor-free phase, we generate different attack scenarios according to different level ancestors.

1-Ancestor We correlate the attack patterns of *Scan & Escalation & Remote*. We obtain 450 attack scenarios by using ASGCR which perform better than CAML. We accumulate post conditions to correlate more attack patterns, and obtain more attack scenarios.

2-Ancestor All attack patterns at *Scan & Escalation & Remote & Local* are used to generate attack

scenarios The numbers of attack scenarios produced by CAML and ASGCR are 192 against 3952. It is apparent that our *2-ancestor method* is more effective in generating attack scenarios than CAML.

As shown in Table 4.2, our method can generate more attack scenarios than that in CAML more completely. Thus we can consider many attack steps to be involved when an attack event is occurred.

Table 4.2. Number of attack scenario.

Method	Scan & Escalation	Scan & Escalation & Remote	Scan & Escalation & Remote & Local
CAML	71	162	192
ASGCR	71	450	3952

5. APPLICATION

After building an attack scenario database, we apply it to security operation center (SOC). An SOC is the core of the whole security infrastructure. As a technical system, it provides assistance in automation to security policy management, security organizational management and security operation management at the upper level. Meanwhile, as the most important, SOC operates to the whole technical layer. It collects all information from both security and non-security products and carries out the unified automatic risk evaluation to tell if they are complying with the policy and baseline of security management. It will report to the designated decision maker and give any necessary respond. By linking the security management and security technology, SOC secures the correct deployment of those security products according to the requirements of security management.

While establishing the SOC, it is necessary to notice that it is not a simplex technology system but an operating method. It combines the security policy management, security organizational management, security operation management and security technology framework. SOC has changed the abstract security risk management to be evaluative and controllable in our daily life.

We can describe SOC as a center to manage alert from various attacks. By applying our attack scenario database into SOC, the system managers are able to understand and analyse the steps of various attacks, also the ability to predict the next step attackers may execute. Thus, an early warning can be achieved before intrusion damages our system.

The system architecture of SOC applying our attack scenario database is shown as below.

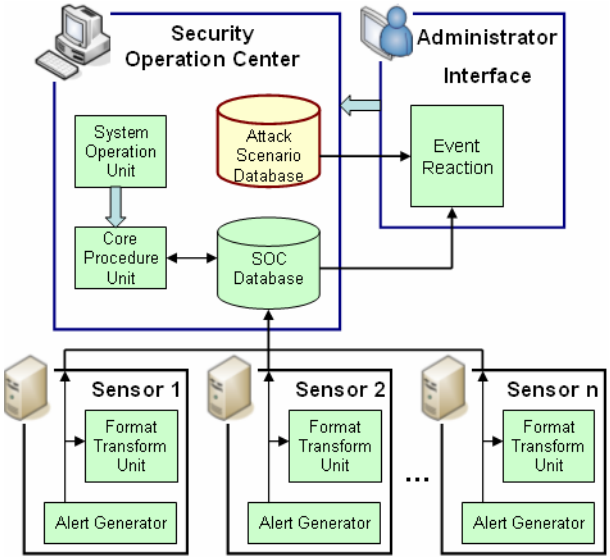

Figure 5.1. SOC with attack scenario database.

6. CONCLUSIONS

In this paper we proposed the system architecture to generate an attack scenario database correctly and completely. Firstly, we can classify and extract attack patterns from four transition states. Then, we correlate attack patterns with pre/post conditions matching, that is, to find all correlation of attack patterns in each transition state. Finally, we use an approach called *Attack Scenario Generation with Casual Relationship (ASGCR)* to build an attack scenario database One key concept in our approach is to generate attack scenarios in two phases, *Ancestor-free Phase* and *t-Ancestor Phase*. This approach can consider all possible attack steps to be involved into attack scenarios, and improve some disadvantages of LAMBDA [3] and CAML [4]. In our application, we apply this attack scenario database to SOC system and develop relational modules. Moreover, the next goal is to generate attack status graph automatically which can help system managers to observe the alerts and manage their relationships visually. Finally, this graph can be used to predict the next step of attacks.

7. REFERENCES

[1] S. Cheung, U. Lindqvist and M. W. Fong, Modeling Multistep Cyber Attacks for Scenario Recognition. *In Proceedings of the Third DARPA Information Survivability Conference and Exposition (DISCEX III)*, Washington, D.C, April 2003.

[2] F. Cuppens and A. Miege, Alert Correlation in a Cooperative Intrusion Detection Framework., *In Proceedings of the 2002 IEEE Symposium on Security and Privacy*, Oakland, CA, May 2002.

[3] F. Cuppens and R. Ortalo, LAMBDA: A language to model a database for detection of attacks, *In Proceed-*

ings of Recent Advances in Intrusion Detection (RAID 2000), pp:197-216, September 2000.

[4] S. Cheung, M. Fong and U. Lindqvist, Modeling Multistep Cyber Attacks for Scenario Recognition., *In Proceedings of the Third DARPA Information Survivability Conference and Exposition (DISCEX III)*, Washington, D.C.,April 22–24, 2003.

[5] O. Dain and R. Cunningham, Fusing a heterogeneous alert stream into scenarios., *In Proceedings of the 2001 ACM Workshop on Data Mining for Security Applications*, pp: 1-13, November 2001.

[6] P. Ning, Y. Cui, and D. S. Reeves, Constructing Attack Scenarios through Correlation of Intrusion Alerts., *In 9th ACM Conference on Computer and Communications Security*, November. 2002.

[7] X. Qin, and W. Lee, Statistical Causality Analysis of INFOSEC Alert Data., *In Proceedings of the 6th International Symposium on Recent Advances in Intrusion Detection (RAID 2003)*, Pittsburgh, PA, September 2003.

[8] S. Staniford, J. Hoagland and J. McAlerney, Practical automated detection of stealthy portscans., *Journal of Computer Security*, Vol.10, No.1-2, pp: 105-136, 2002.

[9] S. Templeton and K. Levit, A requires/provides model for computer attacks., *In Proceedings of New Security Paradigms Workshop,* pp:31-38. September 2000.

[10] A.. Valdes and K. Skinner, Probabilistic alert correlation., *In Proceedings of the 4th International Symposium on Recent Advances in Intrusion Detection (RAID)*, October. 2001.

[11] "Testbed@TWISC", http://testbed.ncku.edu.tw/.