JOURNAL OF INFORMATION SCIENCE AND ENGINEERING 17, 899-919 (2001)

An Intrusion Detection Model Based Upon Intrusion Detection Markup Language (IDML)*

YAO-TSUNG LIN, SHIAN-SHYONG TSENG AND SHUN-CHIEH LIN
Department of Computer and Information Science
National Chiao Tung University
Hsinchu, 300 Taiwan
E-mail: sstseng@cis.nctu.edu.tw

Due to the rapid growth of networked computer resources and the increasing importance of related applications, intrusions which threaten the infrastructure of these applications have are critical problems. In recent years, several intrusion detection systems designed to identify and detect possible intrusion behaviors. In this work, an intrusion detection model is proposed to for building an intrusion detection system which can solve problems involved in building an intrusion detection systems, including pattern representation, computability, performance, extendibility and maintenance problems. In this model, IDML is first designed to express intrusion patterns, and these patterns are transformed into intrusion pattern state machines. Once the intrusion pattern state machines are obtained, the corresponding intrusion detection mechanism that can use these state machines to detect intrusions is designed. To evaluate the performance of our model, an IDML-based intrusion detection experimental system based upon this architecture has been implemented.

Keywords: intrusion detection, intrusion pattern, IDML, XML, finite state machine

1. INTRODUCTION

In recent years, due to dramatic growth in networked computer resources, a variety of network-based applications have been developed to provide services in many different areas, e.g., e-commence services, public web services, and military services. Since many of these applications store and process confidential or important information, they are attacked by local or remote users, the infrastructures of these organizations or companies are threatened or damaged. Therefore, we are concerned with possible intrusion behaviors and securing the system infrastructure.

In this work, an intrusion detection model is proposed for building an intrusion detection system that can solve problems involved in building intrusion detection systems, including pattern representation, computability, performance, extendibility and maintenance problems. The representation of all intrusion patterns in this model determines what kinds of intrusions can be detected and influences the performance of intrusion detection. In our model, the Intrusion Detection Markup Language (IDML), including

Received January 5, 2001; accepted August 18, 2001.
Communicated by Chi Sung Laih.
* This work was partially supported by Ministry of Education and National Science Council of the Republic of China under Grand No. 89-E-FA04-1-4, High Confidence Information Systems.

Patterns, States, Comparators, Events and Properties, based on XML [1], is defined to as being able to describe an intrusion patterns.

General speaking, almost all intrusion patterns can be transformed into sequences of intrusion actions, which will lead the intruding process from the current state to the next state, where the State is used to keep track of the current status of the intruding process. In IDML intrusion pattern expressions, a Comparator which connects one State to another State is defined as the corresponding action of an intrusion that triggers a state transition according to some network information, system log information or some specific or user defined information. The information usually contains a set of properties, which will be structured as Event information in IDML, where each property consists of an attribute and a value. For example, a network packet event includes a set of properties extracted from fields in the packet.

With XML parsers for IDML, the corresponding intrusion pattern state machines can then be constructed for further intrusion detecting. Once the intrusion pattern state machines are obtained, the corresponding intrusion detection mechanism that will use these state machines to detect intrusions can be designed. The detecting algorithm is designed to provide efficient performance in the detecting process, based on the obtained intrusion pattern state machines. Thus, the system architecture of an IDML based intrusion detection model will be proposed, which includes a general model for the intrusion detection system developed based on IDML.

To evaluate the performance of our model, an IDML-based intrusion detection experimental system based upon this architecture has been implemented using existing tools. The IDML parser was implemented using the IBM XML4C [2] parser, and the detection engine of the system was implemented using Snort [3].

2. BACKGROUND

In this section, the existing intrusion categories and several previous researches on intrusion detection systems will be introduced. Several issues concerning the design of intrusion detection systems will then be examined. Since the expressions of intrusion patterns are very important for an intrusion detection system, the expressions of intrusion patterns in currently available intrusion detection systems will also be summarized in this section.

2.1 Categories of Intrusion Behaviors

As the network environment has grown rapidly, so has the problem of intrusions. Currently available approaches to dealing with intrusions can be categorized as follows [4-6]:

Reconnaissance/Snooping/Information Gathering (Probing): This kind of intrusion tries to gather useful information, including public or private data, using the powerful computing capability of a computer.

Gaining Access (User to Root): Intruders or hackers try to get access rights from a victim

host, e.g., the access right of the root account.

Remote Control (Remote to Local): The intruder uses a back door program or takes advantage of application vulnerability to control remote victims through a network.

Denial of Service (DoS): The basic goal of DoS intrusion is to overwhelm the victim host with a huge number of requests. DoS intrusion is easy to achieve, and it can cause the host to crash.

In addition to the intrusions mentioned above, other intrusions may use physical or social strategies to intrude into a system by taking advantage of the vulnerability of the system or application.

2.2 Intrusion Detection System

To protect network environments from intrusions, many products, e.g., firewall products, have been made available on the market. Although different systems may provide different functions and mechanisms for intrusion detection, their main purpose of them which is to detect, filter, or prevent intrusions.

When an intrusion detection system, several issues is designed, including the representation of intrusion patterns, the tradeoff between the complexity of the detection process and the system resources required, and the maintenance of expert knowledge, must be considered.

In traditional firewall systems [7-12], each intrusion pattern can be represented by merely using simple rules, the system administrators should establish rules about what kind of packet information should be filtered or noted, and the system must match the information of each individual packet with these rules. Although the dramatic improvement in hardware systems has improved the processing ability of these firewall systems, it is still very hard for them to deal with the increasing number of rules.

Furthermore, some research on intrusion detection systems has focused on the design of efficient and practical representations of intrusion patterns for representing complex situations. Some specific data structures, including complicated rules and Goal Tree [13], have been used in these researches. They may be robust enough to represent more complex knowledge about intrusions, but they still face the problem of knowledge maintenance. Also, the performance of these systems may not satisfy the on-line performance requirements of an intrusion detection system, and the performance of intrusion detection using these mechanisms cannot be efficiently evaluated.

In conclusion, an ideal intrusion detection system should have an efficient detection mechanism and provide good representation of expert knowledge for intrusion patterns, which should be easily understood and maintained.

2.3 The Representation of Intrusion Behavior

In order to perform intrusion detection, the representation of intrusion behavior is a very important issue for a computer based intrusion detection system. According to the results of previous researches [13, 14], the approaches to representation of intrusion be-

havior can be categorized as follows:

Implicit representation of intrusions: Some intrusion detection systems, such as statistics based intrusion detection systems, use their own models to detect specific intrusion behaviors. Such intrusion detection systems may not provide an understandable representation of intrusion behavior since the knowledge needed for intrusion detection is imbedded in the system.

Rule oriented intrusion representation [7-12, 15]: This is the most common approach to representation of intrusion detection knowledge. In an If…Then formatted rule, the condition of the rule records the match criteria for the intrusion, and the action of the rule records the reaction for the intrusion.

Pattern oriented intrusion representation: Many intrusions may not be completed in a single step, and this is also true of intrusion detection. With only a single rule, only intrusions with a single step or intrusions with a significant feature, e.g., a BO intrusion can be represented. Therefore, for intrusions with several steps, a pattern oriented intrusion representation [16, 17] of intrusion behavior is needed. A pattern oriented intrusion representation can represent an intrusion, for example, a state machine [14, 18-21] or a state diagram [16, 17], in a sequence of states.

Specific intrusion representation: Many researches have tried to define a specific model together with a corresponding specific intrusion representation. For example, goal tree [13], which may achieve good performance for some specific target intrusions, is used to represent intrusion patterns. However, the specific representations sometimes lack extendibility since they may be not suitable for all kinds of intrusions.

Each kind of intrusion behavior representation has advantages and disadvantages, but different intrusion behavior representations for different intrusion detection systems make integration of intrusion behavior knowledge hard to achieve. An expressive intrusion behavior description language would help us to accumulate expert knowledge about intrusions. In the next section, an Intrusion Detection Markup Language based on the XML protocol will be proposed to provide a standardized representation of intrusion behavior.

3. IDML-BASED INTRUSION DETECTION MODEL

When an intrusion detection system is designed, several issues must be considered, which are listed in the following:

Pattern representation: Many intrusions need not just a single step [16, 17, 25], but a sequence of steps to finish. The representation of intrusion behavior should have the ability to represent a sequence pattern.

Computability: Corresponding to the representation of intrusion behavior, there must exist an efficient computer mechanism for performing intrusion detection based on

knowledge included in the intrusion expression [14, 16-21].
Understandability: Before an acceptable automatic intrusion discovery mechanism can be designed, knowledge about new intrusion behaviors must be obtained from the experts of the domain. The representation of intrusion behavior must be understandable so that these experts can express their knowledge about the intrusion.

Performance: An ideal intrusion detection mechanism should achieve real-time detection. The performance issue is especially critical for network intrusion detection, since network traffic behaviors change most frequently.

Extendibility and maintenance: To facilitate reuse of expert knowledge about intrusions, the representation of intrusion behavior must be extendible, which means that it can be refined and extended to cover new intrusions. In addition, a standardized expression language will help us to maintain an intrusion representation.

As mentioned above, the representation of knowledge about intrusion patterns is a very important issue in the design of intrusion detection models. Therefore, an XML based Intrusion Detection Markup Language (IDML), which can be used to express expert knowledge about intrusion patterns, and a corresponding model of an intrusion detection mechanism based on IDML is proposed here.

Since XML is a standard language that is clearly understandable, so is IDML. Thus, the IDML parser can be easily implemented by simply modifying the existing XML parser. We have designed a corresponding intrusion detection model based on IDML. In this model, the intrusion pattern described in IDML can be translated into a finite state machine because the structure of XML is regular expression [1]. Furthermore, IDML documents can be easily reused, and IDML can be extended to describe new intrusion pattern due to the standardized property of XML.

XML provides a more readable and structural format for IDML. Experts can use IDML to express their knowledge about intrusion patterns, and others can understand the meaning of the intrusion patterns easier due to the structure of XML is clear.

In this section, as a first step in defining our intrusion detection model based on IDML, the syntax and corresponding DTD of IDML are first defined. Then the corresponding XML parser which can be used to translate intrusion patterns into a computer-processable finite state machine, will be presented. Finally, the detection system and its detection algorithm will be given.

3.1 IDML (Intrusion Detection Markup Language)

To design an intrusion detection system, determining how to express intrusion behaviors in a computer-processable format is most important. Many different kinds of representations of intrusion behaviors have been proposed [13, 26], but these are limited to currently known intrusion patterns. For newly announced vulnerabilities and intrusions, these mechanisms will not able to extend the restricted intrusion expression model. A more general expression model of intrusions which experts can use to refine the intrusion detection system to deal with new intrusion behaviors is needed.

In this work, an Intrusion Detection Markup Language based upon the XML proto-

col is proposed to provide a general model for representing previously known intrusions and to achieve extendibility so that unknown intrusion behaviors can be dealt with.

General speaking, almost all intrusion patterns can be transformed into sequences of intrusion actions, which will lead the intruding process from the current State to the next State, where the State is used to keep track of the current status of the intruding process. In IDML intrusion pattern representation, a Comparator which connects one State to another state, is defined as the corresponding action of an intrusion performed to trigger a state transition according to some network information, system log information or some specific or user defined information. This information usually contains a set of properties, which will be structured as Event information in IDML, where each property consists of an attribute and a value. For example, a network packet event includes a set of properties extracted from fields in the packet. Fig. 1 shows the structure of IDML.

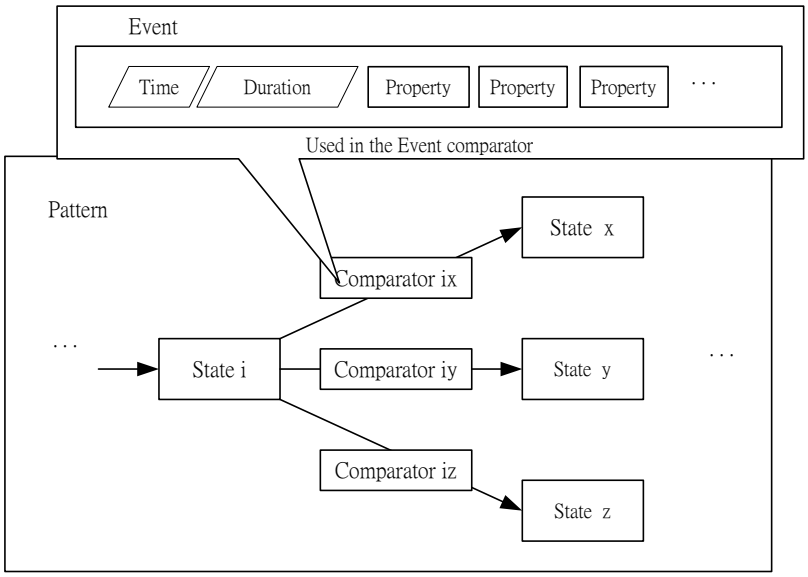

Fig. 1. The structure of IDML.

3.1.1 Property

An intrusion detection mechanism may detect intrusions by considering information obtained from the detection target. In our attribute-based intrusion detection mechanism, the unit in formation, consisting of a property name and a value, is the most basic element in IDML, and the corresponding DTD is declared as shown below:

```
<?xml version="1.0" encoding="Big5" ?>
<!ELEMENT Property (Name, Value, Description?)>
    <!ELEMENT Name (#PCDATA)>
    <!ELEMENT Value (#PCDATA)>
    <!ELEMENT Description (#PCDATA)>
```

In the DTD of Property, three data entries, Name, Value, and Description, are included in a single attribute element. Name is the name of the property, Value is the corresponding value of the property, and Description, which will also appear in other DTDs defined in this paper, is the corresponding comment.

3.1.2 Event

Usually, single property information will not be sufficient to determine an intrusion; this information must include not just a single property, but a set of properties. For example, general network packet information consists of a Source IP Address, a Destination IP Address, etc. An event is defined as a set of properties including time and duration, and the DTD for Event is shown below:

```
<?xml version="1.0" encoding="Big5" ?>
<!ELEMENT Event (Name, Time, Duration, Property+, Description?)>
<!ELEMENT Name (#PCDATA)>
        <!ELEMENT Time (#PCDATA)>
<!ELEMENT Duration (#PCDATA)>
<!ELEMENT Description (#PCDATA)>
```

The corresponding DTD of an event in IDML consists of five kinds of elements in an event, namely, Name, Time, Duration, Property, and Description. Name is used to index one type of event. Time indicates when the event happened, and Duration indicates how long the event lasted. Time and Duration are important attributes of an event and are helpful to express some time serial related behaviors. Property is the information about the event, and more than one Property can be included in an event.

3.1.3 Event comparator

As mentioned above, the information for intrusion detection can be expressed by IDML. However, without the ability of expressing the condition matched in an intrusion pattern, an intrusion pattern cannot be properly described. Event comparator is then defined in IDML to describe the conditions of intrusion pattern.

```
<?xml version="1.0" encoding="Big5" ?>
<!ELEMENT EventCmpr (EventName, AttrCmpr+,Description)>
    <!ELEMENT EventName (#PCDATA)>
        <!ELEMENT AttrCmpr (Name, OPValue)>
            <!ELEMENT AttrName (#PCDATA)>
            <!ELEMENT OPValue (#PCDATA)>
        <!ATTLIST OPValue Compare (Equal|NonEqual|Less|Large|Range|ANY) #REQUIRED>
    <!ELEMENT Description (#PCDATA)>
```

In the event comparator (EventCmpr), the EventName element indicates the target event type for this comparator to operate with; e.g., for an intrusion which can be detected by packet log information, the corresponding comparator will operate with Packet Log event. The AttrCmpr elements list the comparing condition for this comparator, where each of AttrCmpr element includes AttrName and OPValue. AttrName is used to index the comparing attribute in the event information, and the OPValue describes the operator with the required attribute Compare and the value to be compared for the comparator.

3.1.4 Intrusion state

Between the sequence of events in an intrusion pattern, states are defined to record information about the situation of the intrusion process. In IDML, state in an intrusion pattern can be expressed in following DTD:

```
<?xml version="1.0" encoding="Big5" ?>
<!ELEMENT State (StateName, Link*, Action?, Description)>
    <!ELEMENT StateName (#PCDATA)>
     <!ELEMENT Link (EventCmpr, NextState)>
          <!ELEMENT NextState (#PCDATA)>
    <!ELEMENT Action (#PCDATA)>
        <!ATTLIST Action Final(TRUE|FALSE)>
```

There are four kinds of data elements in this DTD, including StateName, Link, Action, and Description. StateName element of a state is used to index the state for other states to link, and Link elements indicates the link between the state and other state in the intrusion pattern. Each Link includes an event comparator (EventCmpr) as the condition for this state moving to next state, and the StateName element of Link shows the name of the next state.

3.1.5 Intrusion pattern

One of the important purposes of IDML is to express intrusion patterns in a standardized and computer-processable format. Therefore, in IDML, the intrusion pattern can be expressed in following DTD format:

```
<?xml version="1.0" encoding="Big5" ?>
<!ELEMENT IntrusionPattern (Name, InitialState, State+, TTL)>
    <!ELEMENT Name (#PCDATA)>
    <!ELEMENT InitialState (#PCDATA)>
    <!ELEMENT TTL(#PCDATA)>
```

An intrusion pattern format in IDML includes four kinds of data elements, Name,

InitialState, PatternState, and TTL. The Name element of the intrusion pattern is used to index the pattern. The InitialState element indicates the name of the initial state of the intrusion pattern, when there is more than one state in an intrusion pattern. The TTL of the intrusion pattern is the Time To Live of the pattern, which indicates how long this intrusion pattern will last. TTL is designed to prevent endless detection, and experts can set TTL to express their expert knowledge about the time limit of the intrusion pattern.

Example 3.1

This example is an intrusion pattern of a U2R intrusion which takes advantage of a system vulnerability of some Linux systems. The steps included in this intrusion are: the telnet process, login, finger root, cgi-bin hole, copy shell, chmod shell, and log information retrieval. The corresponding IDML document is shown below:

```
<?xml version="1.0" encoding="Big5" ?>
<IntrusionPattern>
 <IntrusionName>U2R</IntrusionName>
 <Initial>U2RInitial</Initial>
 <State>
  <StateName>U2RInitial</StateName>
  <Link>
   <EventCmpr>
    <EventCmpr_Name></EventCmpr_Name>
    <PropertyCmpr>

<Prop-
eryCmpr_Name>SystemCall</ProperyCmpr_Name>
      <OPValu Compare="Equal">telnet</OPValue>
    </PropertyCmpr>
   </EventCmpr>
   <NextState>
    Create_rhost
   </NextState>
  </Link>
  <Description>
   This state is the initial state of a U2R intrusion.
  </Description>
 </State>
 <State>
  <StateName>Create_rhost</StateName>
  <Link>
   <EventCmpr>
    <EventCmpr_Name></EventCmpr_Name>
    <PropertyCmpr>
<Prop-
eryCmpr_Name>SystemCall</ProperyCmpr_Name>
      <OPValueCompare="Equal">Login</OPValue>
    </PropertyCmpr>
   </EventCmpr>
   <NextState>
    Finger root
   </NextState>
  </Link>
  <Description>
<Prop-
eryCmpr_Name>Content</ProperyCmpr_Name>
      <OPValue
      Compare="Equal">/cgi-bin/nph-test-cgi/*
      </OPValue>
    </PropertyCmpr>
   </EventCmpr>
   <NextState>
    copy shell
   </NextState>
  </Link>
  <Description>
   utilize the http hole
  </Description>
 </State>
 <State>
  <StateName>copy shell</StateName>
  <Link>
   <EventCmpr>
    <EventCmpr_Name></EventCmpr_Name>
    <PropertyCmpr>
     <ProperyCmpr_Name>
          SystemCall
      </ProperyCmpr_Name>
     <OPValue Compare="Equal">cp/bin/sh /tmp/.sh
      </OPValue>
    </PropertyCmpr>
   </EventCmpr>
   <NextState>
    chmod shell
   </NextState>
  </Link>
  <Description>
   copy shell to temp for got root privilege
  </Description>
 </State>
 <State>
  <StateName>chmod shell</StateName>
  <Link>
   <EventCmpr>
```

```
    This state is reached when a telnet connection is built.
   </Description>
  </State>
  <State>
   <StateName>Finger root</StateName>
   <Link>
    <EventCmpr>
     <EventCmpr_Name></EventCmpr_Name>
     <PropertyCmpr>
<Prop-
eryCmpr_Name>SystemCall</ProperyCmpr_Name>
       <OPValueCompare="Equal">finger root</OPValue>
     </PropertyCmpr>
    </EventCmpr>
    <NextState>
     cgi-bin hole
    </NextState>
   </Link>
   <Description>
    wait to root login
   </Description>
  </State>
  <State>
   <StateName>cgi-bin hole</StateName>
   <Link>
    <EventCmpr>

<EventCmpr_Name>Packet</EventCmpr_Name>
     <PropertyCmpr>
     <EventCmpr_Name></EventCmpr_Name>
     <PropertyCmpr>
      <ProperyCmpr_Name>
            SystemCall
       </ProperyCmpr_Name>
      <OPValue Compare="Equal">chmod 4755 /tmp/.sh
       </OPValue>
     </PropertyCmpr>
    </EventCmpr>
    <NextState>
     Log_Info
    </NextState>
   </Link>
   <Description>
    The shell in the temp directory has already
    changed its mode to executable.
   </Description>
  </State>
  <State>
   <StateName>Log_ Info </StateName>
   <Description>
    This state is reached when the intrusion patterns
    have all been matched.
   </Description>
   <Action Final="TRUE">Log </Action>
  </State>
  <TTL>900</TTL>
 </IntrusionPattern>
```

3.1.6 Analysis of the properties of IDML

In order to describe the ability of IDML to represent intrusion patterns, we will focus on the following issues:

1. How to collect and analyze the event information required in the pattern.
2. How to transform the intrusion into a state transition pattern.
3. How to express an event comparison between states.
4. What kinds of actions should be in response to an intrusion.

When intrusions originate due to protocol vulnerabilities, system vulnerabilities, or application vulnerabilities, the problems related to detecting such intrusions depend on event information extraction, and intrusion patterning. Although user behaviors in a computer based environment are all recordable and traceable, the properties of the four issues mentioned above must be known in advance so that a corresponding mechanism for analyzing user behavior and collecting event information can be designed.

In [27, 28], computer based intrusions were divided into four categories, namely, Probing, U2R (User to Root), R2L (Remote to Local), and DoS (Denial of Service). Most complex probing intrusions leave log messages since they expend more effort to obtain information about the victim host and they often scan many service ports of the victim host. Connection log information can help us describe intrusion patterns and detect intru-

sions using IDML. On the other hand, probe intrusions which may leave less log information, more detailed information must be obtained and logged from network connections. IDML is able to obtain such detailed log information to perform intrusion detection.

A U2R intrusion tries to obtain access rights to perform system level intrusion. Such intrusions can usually be represented by intrusion patterns since the actions of the intrusion are in the users' behavior logs or system kernel logs. Recognition and detection of the corresponding intrusion pattern can also be achieved by comparing the behaviors of a single user with the intrusion pattern. However, a U2R intrusion which involves fewer steps, it is harder to detect, and more information must be obtained to detect it.

An R2L intrusion is an intrusion that comes from a network, through which the remote intruder connects and controls the local host. Therefore, the intruder can gain information by using preinstalled applications, e.g., backdoor programs, or can gain control by taking advantage of some vulnerabilities of the remote daemon, e.g., an un-patched httpd or ftpd program on a Linux system. This kind of intrusion can be identified and detected by analyzing network connection logs, which can be obtained from the system kernel. It should be noted that most R2L intrusions can be identified using a small piece of information and can be easily detected. For example, many backdoor program intrusions can be detected by checking the port number of the connection.

The basic idea of DoS intrusion is to overwhelm the victim host with a huge amount of requests. For example, a SYN-flood intrusion uses most of the computational resources of the victim host by opening many half-open TCP/IP connections. Detection of this kind of intrusion involves the problem of identifying the intruder. Basically, information about DoS intrusion behaviors is included in network connection logs. However, DoS intrusions do not need to perform any specific actions on the system, which means that DoS intruders can easily hide without leaving obvious clues about themselves; for example, DDoS (Distributed DoS) intrusion coordinates many hosts to intrude the victim host. Since identification of the user is sometimes hard to achieve, intrusion patterns of DoS intrusions are also hard to define. Only if the problem of user identification can be solved, can the pattern of a DoS intrusion be defined and expressed in IDML.

In the future, a Distributed IDS (DIDS) based on both IDML and IDMEF (the Intrusion Detection Message Exchange Format) [29] proposed by IDWG (the Intrusion Detection Working Group) [30] will be proposed. In the architecture of the IDD (the Intrusion Detection Device) and CIDS (the Center of Intrusion Detection System) models for DIDS, IDML can be used to describe intrusion patterns, and IDMEF can be used to facilitate the exchange of information and events between IDD and CIDS. Furthermore, CIDS can perform higher level intrusions according to the events reported by IDD.

As discussed above, using IDML to express intrusion patterns will lead to some problems with information collection, user identification, and intrusion pattern definitions. However, the intrusions performed through social engineering or physically destructive processes are outside our discussion.

3.2 IDML Parser

In order to understand the knowledge and information included in intrusion patterns defined in IDML, an IDML Parser is needed to parse the related information in the

IDML document. Since IDML is based on the XML standard, the parser for IDML can be easily obtained by slightly modifying the original XML parser. In addition to validating the syntax of IDML documents, information included in IDML can be also obtained from the IDML parser, and intrusion patterns can be transformed into a finite state machine for further intrusion detection.

3.2.1 Intrusion pattern state machine

The concepts involved in constructing an intrusion pattern state machine are shown in Fig. 2:

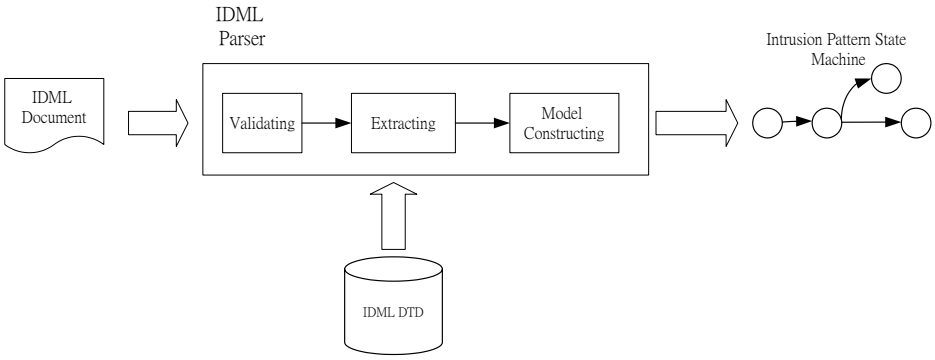

Fig. 2. The concept of IDML parser.

As shown in Fig. 2, the IDML parser first validates the syntax of the IDML document according to the input IDML DTD. The intrusion pattern model can then be constructed by the Model Constructing Module using information extracted from the IDML document. Since the XML document is hierarchically structured, the intrusion pattern can be modeled by a finite state machine in which the set S is the set of States defined in an IDML document, and the set of state machine inputs Σ is the Event defined in IDML. The state transition function δ for an intrusion pattern state machine can be constructed from the Links between states, and for all the events not included in any Link, a self-loop can also be constructed as shown in Fig. 3. The Initial state s is defined in the Intrusion pattern, and the Final states are labeled by the attribute in a state definition.

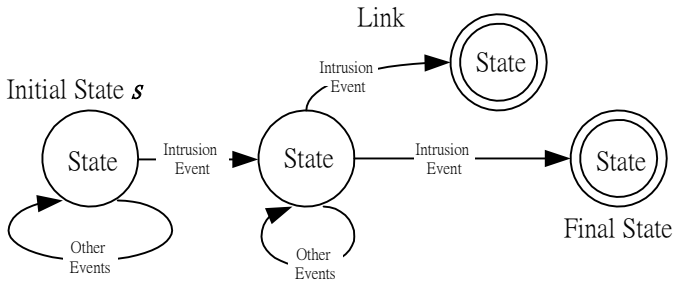

Fig. 3. The finite state machine for IDML documents.

3.3 IDML Based Intrusion Detection System

So far, the IDML parser has the ability to translate IDML based intrusion patterns into a finite state machine. In this section, we will first focus on the problem of how to identify a user, since intrusion detection depends on the behavior of each user. We will also propose a general detection model for IDML-based intrusion detection. Finally, the system architecture for an IDML based intrusion detection mechanism will be illustrated and explained.

3.3.1 User identification

Since the purpose of intrusion detection is to find the person who is attempting an intrusion, user identification is a very important issue. Based on the event information collected, the following models can be used to identify a user from lots of transaction data:

1. IP Address: Since an IP Address is used as an address of a network host, the IP Address of the detection target is widely treated as the identity of the user. However, this approach does not work well for multi-user systems.
2. User Login: Some event information obtained from the system kernel contains the user login and authentication information.
3. Application Log: Some applications or services record their own log information; e.g., web logs and ftp logs.
4. User behavior mining: Based on statistical analysis or machine learning mechanisms, some researches [15, 31-33] have tried to identify a user by mining his behavior profile. This kind of identification mechanism provides a more flexible and variant model for intrusion detection.

Assuming that a proper user identification mechanism can be built for any kind of event information, the problem of identifying the user can be solved.

3.3.2 Intrusion detection model

In our intrusion detection model, all available event information is used to detect intrusions. For each event information, we assume that the user who triggered the event can be identified from the event information. The corresponding user identification mechanism will identify the user, and then the event information will be used as input in each intrusion detection state machine to analyze whether or not any state transition should be performed from the current state of the user in the state machine, where the state information for each state machine is recorded for each user. If the state changed due to the transition, then the action associated with the state in IDML will be performed when the state is reached. And if the final state is reached, then intrusion pattern detection will be triggered, and the state of the intrusion pattern for the user will be reset to the initial state of the intrusion pattern.

We assume that only intrusions which are performed within a certain period of time

are considered, and that each intrusion has a threshold for this period of time. Detecting an intrusion without a threshold will require many system resources since the detecting process will never end. Thus, an intrusion pattern is defined as TTL (Time to Live), which means that if the process of the state machine of an intrusion pattern is idle for a period of time longer than TTL, then the state will be reset to the initial state of the intrusion pattern.

The algorithm for intrusion detection in the IDML based intrusion detection model is shown in the following:

Notations:

e	event information, which is a set of possible attributes, e.g., packet information, system call logs;
P	the set of all intrusion patterns;
S_{ij}	indicates the state in intrusion pattern j for user i;
δ_j (s, e)	transition function of the constructed state machine of intrusion pattern j, where s is the current state in j, and e is the coming event information;
$Init_j$	the initial state of intrusion pattern j.
T_{ij}	the idle time of state machine j for user i;
TTL_p	the TTL for intrusion pattern p.

Algorithm 3.1: Intrusion Detection Algorithm of the IDML based intrusion detection model.

```
While event e needs to be handled
    Identify user i according to e
    For each pattern j in P
        if δ_j ( S_ij , e ) is equal to S_ij then
            T_ij = T_ij + Idle period from the previous event
            If T_ij > TTL_j, then S_ij = Init_j
        else
            S_ij = δ_j ( S_ij , e )
            Perform the action of S_ij if it exists.
            if S_ij is in the final state, then S_ij = Init_j and reset T_ij.
    End for
End While
```

Based on this detection model, the intrusion can be detected using the IDML formatted intrusion pattern.

3.3.3 System architecture

So far, we have provided a solution for intrusion pattern representation and a corresponding intrusion detection mechanism. DTD is used to provide a meta description to the description language and forces the IDML description to be more structured. Due to the nature of XML language, more complex patterns or rules can be expressed using IDML and the maintenance of IDML will be better than some specific data structures, for example, trees or rules. In other words, the IDML will be more structural than other data structures.

The architecture of the intrusion detection process based on IDML is shown in Fig. 4.

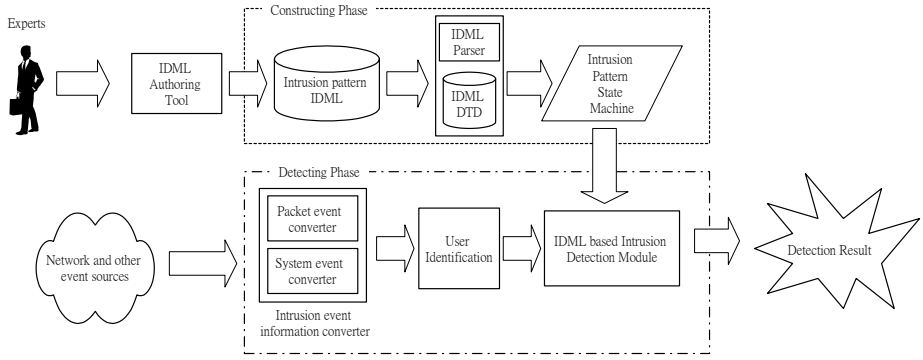

Fig. 4. The architecture of IDML based intrusion detection system.

There are two phases in this process, including a construction phase and a detection phase. In the construction phase, the intrusion pattern IDML documents, which are written by experts on intrusion using an authoring tool, are stored in data storage. The IDML parser is used to validate the intrusion pattern document using the corresponding intrusion pattern DTD. If the pattern is valid, the intrusion pattern will be translated into intrusion pattern state machines for further use in the detection process.

In the detection phase, the state machines generated in the construction phase are used to detect intrusions. An event information converter transforms the information from the network or other event sources into an event, and the event converter can be enhanced by including new event types. The IDML based intrusion detection module then detects intrusions from event information based on the intrusion detection state machines. Finally, the results of intrusion detection can be obtained.

4. IMPLEMENTATION AND EXPERIMENT

To evaluate the performance of our model, the architecture of an IDML-based intrusion detection system was implemented using existing tools and technologies. As shown in Fig. 5, our experimental system is based on Snort, which is an Open-source project, and a subset of general functions provided by Snort [3], including packet log collecting and packet decoding, is used. The IBM XML4C parser [2] was used to validate each IDML document using IDML DTD and to extract the information in each IDML document to construct intrusion pattern state machines. Many rules about current intrusions are also defined in the Snort rule base. In order to use this accumulated knowledge about intrusions, a translator was designed to translate Snort rules into IDML formatted intrusion patterns.

4.1 Experiment

To evaluate the capability of the IDML based intrusion detection model proposed in this work, an experiment was designed. In the experiment, two IDS systems were run in the same subnet of the Microelectronics and Information Building of National Chiao

Tung University, which is a 10/100 based Ethernet environment. One of the IDS systems was the original Snort system, version 1.5, running on Linux Debian knerel 2.4.6. The other IDS system was our IDML based intrusion detection system, which was a modified version of the same Snort system and was run under the same operating system.

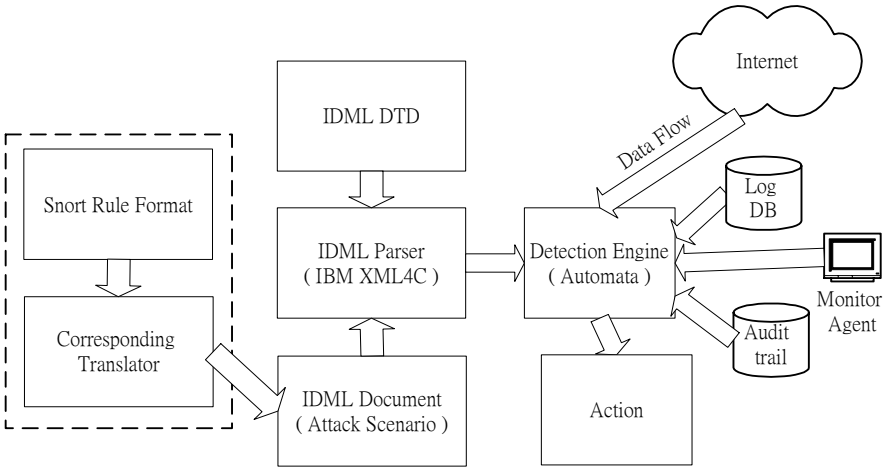

Fig. 5. An IDML-based intrusion detection experimental system.

All of the rule sets included in the original snort system were used to perform detection and transformation necessary to translate these rules into IDML described intrusion patterns. Rules in the Snort system can be transformed into two-state IDML intrusion patterns, which consist of the initial state, final state, and the *Comparator* between these two states, including the rule condition of the original snort rule.

In our IDML based IDS, additional patterns were included. All of these patterns involved several steps and could not be represented or detected in Snort 1.5. These patterns included the examples described in the previous sections and several Probing and DoS intrusion patterns.

The experimental results for the number of alarms launched by these two IDSs are shown in Table 1.

Table 1. The alarms announced by the IDML based IDS and Snort.

	Snort with the IDML based intrusion detection model	**Original Snort**
Probing	3264	2642
U2R	31	26
R2L	167	123
DoS	10562	7620
Total	14024	10411

From the intrusion alarms generated by both systems, the logs were traced, and false alarms were pruned. The numbers of intrusions which occurred in both systems during the period of the experiment are shown in Table 2.

Table 2. The numbers of real intrusion alarms of both systems.

	Snort with the IDML based intrusion detection model	Original Snort
Probing	2371	1763
U2R	21	16
R2L	117	73
DoS	7812	4931
Total	10321	6783

Through analysis of the results shown in Table 2, the false alarms included in the reports of both systems could be pruned, and the accuracy of these two systems for detecting intrusions occurring in the same environment during the same period is shown in Table 3.

Table 3. The detection accuracy of both systems.

	Snort with the IDML based intrusion detection model	Original Snort
Probing	0.73	0.67
2R	0.68	0.62
R2L	0.70	0.59
DoS	0.74	0.65
Average	0.736	0.652

In order to test the ability of both systems to detect intrusions, some experimental intrusion behaviors were examined to verify if IDS successfully detected these intrusions. However, the huge number of Probing and DoS intrusions may have occurred for the following reasons:

1. Verbose alarms: Since Snort is a packet level network intrusion detection system, alarms frequently occur due to the huge number of network connections and packets.
2. Bad Settings: Many network applications will take DoS-like behaviors if the settings are wrong. For example, without properly set DNS related settings, some applications will try to ask DNS frequently as a DoS intrusion.
3. Damaged Equipment: Damaged network equipment can also cause DoS-like intrusions, e.g., half-opened connections.

Based on Tables 2 and 3, it seems the IDML based intrusion detection model could be used to detect a wide variety of intrusions, and that the number of detected intrusions did not decrease due to an overload caused by detecting additional intrusion patterns in the real network environment.

To maintain the states for each network access, additional memory space will probably be required to record the state information, and the cost of this memory overhead will increase. The memory usage of these two systems is shown in Table 4.

Table 4. The memory usage of both systems, measured in KBs.

	1	2	3	4	5	6	7	8	9	10	11	12
Original Snort	4052	4052	4064	4032	4052	4056	4060	4060	4060	4064	4064	4072
Snort with IDML based intrusion detection mode	5632	5684	5692	5640	5624	5648	5684	5664	5704	5720	5692	5720

The memory usage in both systems remained stable, and burst or overload situations did not occur in the experiment. It seems that the IDML based intrusion detection system did not heavily affect performance and or memory usage.

5. CONCLUDING REMARKS

The Intrusion Detection Markup Language (IDML) has been defined to provide a standardized representation of intrusion patterns and to solve problems related to intrusion detection systems, including pattern representation, computability, performance, extendibility and maintenance.

In addition to IDML syntax, an IDML parser has been also designed, which is responsible for translating the IDML described intrusion patterns into a computer- processable format for intrusion detection. The obtained intrusion pattern state machines can then be used to efficiently detect intrusions by means of the corresponding intrusion detection mechanism. An intrusion event converter has been also designed to provide extendibility of the intrusion detection system. An IDML based intrusion detection experimental system, which uses the IBM XML4C parser as the IDML parser and Snort as the detection engine, has been implemented.

In the experiment, the IDML based intrusion detection system was implemented, and the experimental results show that our IDML based intrusion detection model provides expressive power for detecting complicated and multi-phase intrusion behaviors without increasing the cost of resources.

Since intrusion pattern authoring would be very helpful for experts who need to describe their knowledge about intrusions in IDML, we are building a visualized authoring tool to help them write patterns. Also, because each intrusion pattern expressed in IDML can be transformed into one state machine by the IDML parser, we are designing a merging algorithm which can merge several intrusion pattern state machines into one to reduce the number of state machines and improve the efficiency of the intrusion detection process.

In the future, a Distributed IDS (DIDS) based on both IDML and IDMEF [33] proposed by IDWG [34] will be designed. IDMEF has been proposed to exchange information between intrusion related applications. In the architecture of IDD (Intrusion Detection Device) and the CIDS (Center of Intrusion Detection System) model for our DIDS, IDML can be used to describe intrusion patterns, and IDMEF can be used for exchanging information and events between IDD and CIDS. In addition, CIDS may perform higher level detection of intrusions according to the events reported by IDD.

REFERENCES

1. W3C, "XML page," http://www.w3.org/XML/, 2000.
2. IBM, "XML4C, " http://www.alphaworks.ibm.com/tech/xml4c/, 2001.
3. R. Marty**, "**Snort– the open source network IDS," http://www.snort.org/, 2001.
4. M. Stuart and S. Joel, "InfoWorld security sweet 16 (ISS16)," http:// www.infoworld. com/cgi-bin /displayNew.pl?/security/links/security_iwss16.htm.
5. D. Newman, T. Giorgis and T.I. Farhad, "Intrusion detection systems: suspicious," http://www.data.com/lab_tests/intrusion.html, Aug., 1998.
6. Y. L. Cheng and C. S. Laih, "The design and implementation of a distributed network intrusion detection system with the reconnaissance ability," Master's thesis, Department of Electrical Engineering, National Cheng Kung University, Taiwan, June 2000.
7. Cisco, "Cisco PIX firewall manual," http://mail.ht.net.tw/~erik/doc/cisco/pix.zip, 1999.
8. CLDP, "CLDP firewall how to," http://freebsd.ntu.edu.tw/cldp/Firewall-HOWTO. html, 2000.
9. SYSWARE CORPORATION, "CheckPoint 2000," http://firewall.sysware.com.tw/, 2000.
10. ADCOM Technology Inc, "Sonic wall," http://www.adcom.com.tw/product/sonicw/ index.htm, 2000.
11. FEYA TECHNOLOGIES CO., "Border ware 6.0," http://www.feya.com.tw/security/ borderware.html, 2000.
12. Megasoft Corporation, http://www.taipeisoft.com/Products/WinR/winr.html, 2000.
13. M. Y. Huang, R. J. Jasper, and T. M. Wicks, "A large scale distributed intrusion detection framework based on attack strategy analysis," *Computer Network*, Vol. 31, No. 23-24, 1999, pp. 2465-2475.
14. G. Vigna and R. A. Kemmereer, "NetSTAT: A network-based intrusion detection approach," in *Proceedings of IEEE International Conference on Computer Security Applications*, 1998, pp. 25-34.
15. U. Lindqvist and P. A. Porras, "Detecting computer and network misuse through the production-based expert system toolset (P-BEST)," in *Proceedings of IEEE Symposium on Security and Privacy*, 1999, pp. 146-161.
16. S. W. Shieh and V. D. Gligor, "A pattern-oriented intrusion-detection model and its applications," in *Proceedings of IEEE Computer Society Symposium on Research in Security and Privacy*, 1991, pp. 327-342.
17. S. P. Shieh and V. D. Gligor, "On a pattern-oriented model for intrusion detection," *IEEE Transactions on Knowledge and Data Engineering*, Vol. 9, No. 4, 1997, pp. 661-667.
18. R. A. Kemmerer, "NSTAT: A model-based real-time network intrusion detection system," *Technical Report TRCS-97-18*, Department of Computer Science, University of California, Santa Barbara, Nov., 1997.
19. K. Ilgun, R. A. Kemmerer, and P. A. Porras, "State transition analysis: A rule-based intrusion detection system," *IEEE Transactions on Software Engineering*, Vol. 21, No. 3, 1995, pp. 181-199.
20. P. A. Porras, "STAT – A state transition analysis tool for intrusion detection," Mas-

ter's thesis, Computer Science Department, University of California, Santa Barbara, June 1992.
21. K. Ilgun, "USTAT: A real-time intrusion detection system for UNIX," in *Proceedings of IEEE Computer Society Symposium on Research in Security and Privacy*, 1993, pp. 16-23.
22. C. E. Kahn Jr, "Standard generalized markup language for self-defining structured reports," *International Journal of Medical Informatics*, Vol. 53, 1999, pp. 203-211.
23. E. Guerrieri, "Software document reuse with XML," in *Proceedings of the 5th International Conference on Software Reuse*, 1999, pp. 246-254.
24. W3C, "XSL page," http://www.w3.org/Style/XSL/, 2000.
25. CERT, *http://www.cert.org/*, 2000.
26. K. Richards, "Network based intrusion detection: A review of technologies," *Computer and Security*, Vol. 18, No. 8, 1999, pp. 671-682.
27. R. P. Lippmann, D. J. Fried, I. Graf, J. W. Haines, K. R. Kendall, D. McClung, D. Weber, S. E. Webster, D. Wyschogrod, R. K. Cunningham, and M. A. Zissman "Evaluating intrusion detection systems: the 1998 DARPA off-line intrusion detection evaluation," in *Proceedings of DARPA Information Survivability Conference and Exposition*, Vol. 2, 2000, pp. 12-26.
28. K. R. Kendall, "A database of computer attacks for the evaluation of intrusion detection systems," Master's thesis, Department of Engineering and Computer Science, University of Massachusetts Institute of Technology, June 1999.
29. D. Curry and H. Debar, "Intrusion detection message exchange format data model and extensible markup language (XML) document type definition," http://www.ietf.org/internet-drafts/draft-ietf-idwg-idmef-xml-03.txt, 2001.
30. M. Erlinger and S. C. Stuart, "Intrusion detection working group (IDWG) charter," http://www.ietf.org/html.charters/idwg-charter.html, Oct., 2000.
31. S. Manganaris, M. Christensen, D. Zerkle, and K. Hermiz "A data mining analysis of RTID alarms," *Computer Network*, Vol. 34, No. 4, 2000, pp. 571-577.
32. P. G. Neumann and P. A. Porras "Experience with EMERALD to data," in *Proceedings of the 1st USENIX Workshop on Intrusion Detection and Network Monitoring*, 1999, pp. 73-80.
33. P. A. Porras and P. G. Neumann, "EMERALD: Event monitoring enabling responses to anomalous live disturbances," http://www2.csl.sri.com/emerald/concepts.html, 1999.

Yao-Tsung Lin (林耀聰) was born in Taichung, Taiwan, on November 5, 1975, he graduated with a B.S. degree from the Department of Computer Science, Nation Tsing Hua University, Taiwan in 1997. He received the M.S. degree from the Department of Computer and Information Science, National Chiao Tung University, Taiwan in 1999. Currently, he is a Ph.D. student at National Chiao Tung University, Taiwan. His current research interests include Internet-based applications, knowledge engineering, expert systems, and data mining etc.

Shian-Shyong Tseng (曾憲雄) received his Ph. D. degree in Computer Engineering from the National Chiao Tung University in 1984. Since August 1983, he has been on the faculty of the Department of Computer and Information Science at National Chiao Tung University, and is currently a Professor there. From 1988 to 1992, he was the Director of the Computer Center National Chiao Tung University. From 1991 to 1992 and 1996 to 1998, he acted as the Chairman of Department of Computer and Information Science. From 1992 to 1996, he was the Director of the Computer Center at Ministry of Education and the Chairman of Taiwan Academic Network (TANet) management committee. In December 1999, he founded Taiwan Network Information Center (TWNIC) and is now the Chairman of the board of directors of TWNIC. His current research interests include parallel processing, expert systems, computer algorithm and Internet-based applications.

Shun-Chieh Lin (林順傑) was born in Taipei, Taiwan, on January 5, 1977, he graduated with a B.S. degree from the Department of Computer Science and Information Engineering, Tamkang University, Taiwan in 1999. He received the M.S. degree from the Department of Computer and Information Science, National Chiao Tung University, Taiwan in 2001. Currently, he is a Ph.D. student at National Chiao Tung University, Taiwan. His current research interests include network security, knowledge engineering, and data mining etc.