EVENTS PLANNING IN INTRUSION DETECTION SYSTEMS [1]

Liberios VOKOROKOS, Anton BALÁŽ, Norbert ÁDÁM
*Department of Computers and Informatics, Faculty of Electrical Engineering and Informatics, Technical University of Košice, Letná 9, 042 00 Košice
E-mail:liberios.vokorokos@tuke.sk, anton.balaz@tuke.sk, norbert.adam@tuke.sk

SUMMARY

The goal of this paper is to present designed architecture of intrusion detection system based on events planning and intrusion signature. The article describes problematic of the variation of intrusions and intrusion detection systems. The core of the proposed architecture is intrusion signature matching through petri nets that clasify system behaviour and determine potential intrusion of monitored computer system. The result is a new method of intrusions signature detection.

Keywords: *intrusion detection system, architecture, planning, petri nets*

1. INTRODUCTION

The paper considers an area of computer system security based on a nonauthorized access to the computer system. The existing intrusion detection approaches can be divided in two classes - an anomaly detection and a misuse detection [3]. The anomaly detection approaches the problem by attempting to find deviations from the established patterns of usage. On the other hand, the misuse detection, compares the usage patterns to known techniques of compromising computer security. Architecturally, the intrusion detection system can be categorized into three types - host-based IDS, network-based IDS and hybrid IDS. The host-based IDS, deployed in individual host-machines, can monitor audit data of a single host. The network-based IDS monitors the traffic data sent and received by hosts. The hybrid IDS uses both methods.

The intrusion detection through multiple sources represents a difficult task. Intrusion pattern matching has a nondeterministic nature where that same intrusion or attack can be realized through various permutation of the same events. The purpose of this article is to present authors' proposed intrusion detection architecture based on the partially ordered events and the petri nets [2]. Project is proposed and implemented at the Department of Computers and Informatics in Košice supported by VEGA 1/4071/07.

2. ARCHITECTURE OF DESIGNED IDS SYSTEM

Proposed system architecture includes part of planning and matching. The matching means that the system gets into a state of intrusion when a sequence of events leading to the mentioned state occurs. The intrusion is a system state which overtakes previous states represented by particular system events. If there exists such a fine-grained log system, it is possible to detect the states with intrusion. Single attacks to the systems represents mentioned single events that in a final implication leads to the state of intrusion. Characteristic feature of intrusions is their variability; permutation of same events leads to same state of intrusion. Single intrusions are characteristic with their nondetermination. Designed IDS system solves this problem with planning that responds to lay-out of possible sequence of steps leading to the final intrusion. Planning part creates the intrusion plan by first-order logic when it describes known activities and disturber's goals to specify attack sequences. Result of planning is intrusion specification and it's single steps that uses the matching part of the system to the intrusion detection. System architecture designed on the Deparment of Computers and Informatics is in the figure 1.

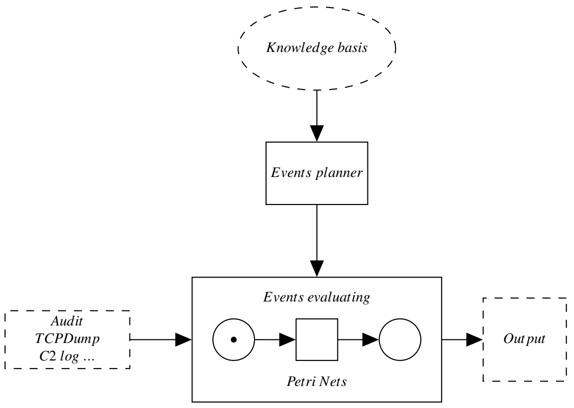

Fig. 1 Designed IDS architecture

2.1. Intrusion signature sequence planning

Intrusion is defined as a set of events with a focus on compromise integrity, confidentiality and resources avaibility. Designed architecture of IDS includes the planning part to construct event sequence plan of which consists the intrusion. Planning includes goals, states and events. According to what is necessary to do in final plans, planning combines actual enviroment state with information depending on the final result of events.

[1]Supported by VEGA 1/4071/07

State transition si characterized as a sequence of events performed by intruders leading from initial state do final compromised state. Planning can be formuled as a problem of state transition:

- *Initial state:* Actual state description.
- *Final state:* Logical expression of concrete system state.
- *Intrusion signatures:* Events causing change of a system state.

Planning is defined as:

1. Set of single steps of the plan. Every step represents control activity of the plan.
2. Set of ordered depedencies. Every depedency is in a form of $S_i < S_j$, where, step S_i is executed before S_j.
3. Set of variable bindings. Every binding is in a $v = x$ form, where v is a variable in some step and x is a constant or other variable.
4. Set of causal bindings. Causal binding is in a form of $S_i \xrightarrow{c} S_j$. From state S_i by auxiliary c state S_j , where c is a necessary pre-condition for the S_j.

Every signature has an associated pre-condition that indicates what has to be completed before it is possible to apply event binded with the signature. Post-condition expresses event result connected to the intrusion signature. A task of the planning is to find events sequence reponsible for the intrusion. The goal of planning in the designed IDS architecture is to find event sequence and their depedencies and construct result sequence of an intrusion.

Partially ordered planning allows to represent plans in which some steps are ordered according to other steps. Instrusion signatures and their nature of nondetermination are suitable for fundamentals of partially ordered planning. Planning consists of database of intrusions and events planner figure 1. Knowledge base includes information about each intrusion signature including pre and post condistions of these events in the form of first-order logic. The planner generates set of events and their's depedencies for each initial and final intrusion state. Futhermore, knowledge base includes state depedencies for each event signature. These information are used by planner for defining partially ordering in between intrusion signature. For instance pre-condition intrusion signature consists of k terms. These are represented in form of symbols

$$\{PS_1, PS2, \dots, PS_k\} \wedge \{PS_j < PS_k\} \wedge \dots \wedge \{PS_l < PS_m\}$$

An algorithm of partially ordered planning begins with minimal plan and in each step this plan is extended through available pre-condition step. This is realized by selecting intrusion signature that fulfill some of the unfulfilled pre-conditions in the plan. For a newly fulfilled pre-conditions of event signatures are causal bingings stored in between them. These bindings are necessary for partially event ordering. An ordering result is represented by set of events and their depedencies in between these event signatures. Let intrusion sequence to consist of n event signatures: $SA_1, SA_2, \dots, SA_n$, then intrusion structure is specified as

$$\{SA_1, SA_2, \dots, SA_k\} \wedge \{SA_j < SA_k\} \wedge \dots \wedge \{SA_l < SA_m\}$$

First part of this term $\{SA_1, SA_2, \dots, SA_k\}$ is a set of event signatures. Next part of the term is ordering depedency between signatures. The intrusion example referring to figure 3 is specified as

$$\{cp, chmod, touch, mail\} \wedge \{cp < chmod\} \wedge \{chmod < mail\} \wedge \{touch < mail\}$$

Each formulation represents an intrusion signature variation that leads to the same compromised states. In the case of the intrusion signatures it is necessary to deliberate this intrusion variability from the view of memory requirements. Futher, if it comes to the alternation of initial state, it may have a consequence of complete intrusion plan alternation. The next advantage of partially ordered planning is that the time between two intrusion signatures does not have an influence on the analysis during capturing system data and state changing.

2.2. Events planning

This chapter represents planning algorithm in the designed IDS that's result is partially ordered events of intrusion signature. It is possible to represent the intrusive plan through the triple $\langle A, O, L\rangle$, where A is a set of events, O is a set of ordered dependencies on the A set, and L is a set of casual connections. The planner starts its activity with a blank plan, and it specifies this plan in stages with being obligated to comsideration of consistence requirements defined in the O set. The key step of this activity is to preserve states of the past conclusions and reqierements for these conclusions. For the provision of consistence within various events, the recording of relations within the events is performed through the casual connections. Casual connection is a structure consisting of two references to plan events (producer A_p and consumer A_c), and Q assertion that is the result of the A_p and the A_c precondition. The expresion is represented by $A_p \xrightarrow{Q} A_c$ and connections themselves are stored in the L set. Casual connections are used for the detection of interference within new and old conclusions. Marked as threats. This means that $\langle A, O, L\rangle$ represents a plan and $A_p \xrightarrow{Q} A_c$ is a connection in L. Let the A_t be another event in A, than the A_t endangeres the $A_p \xrightarrow{Q} A_c$ if:

- $O \cup \{A_p < A_t < A_p\}$ a
- A_t has $\neg Q$ as the result

If the plan involves threats, it cannot suit the scheduled requirements defined in $\langle A, O, L\rangle$. The threats must be considerated by the planner during assembling the final plan. The algorith can add supplementary order dependencies by assurance of performance of A_t before the A_p. This method of the thread removal represents so-called disassembly of plan.

The core of the planning is represented by the algorithm of planning, mentioned below, that searches the state environment of the plans. The algorithm begins with a blank plan and performes non-deterministic selection of the event sequence in stages, till all the preconditions are considerated through their casual connections and till potential threads of the plan are eliminated. Partially ordered dependencies of the final plan are over again represented by only partialy ordered plan, that resolves the problem of planning. The algorithm arguments are the planning structure, and the plan agenda. Each agenda item is a pair $\langle Q, A\rangle$, where Q is a conjunction of A_i preconditions.

Planning $(\langle A, O, L\rangle, agenda, \Lambda)$

1. ***Completion:***If the agenda is empty, return $\langle A, O, L\rangle$.
2. ***Target selection:*** $\left\langle Q, A_{potr}\right\rangle$ is a pair in the agenda (according $A_{port} \in A$ and Q is conjunction of preconditions A_{potr}).
3. ***Event selection:*** $A_{potr} = event\ selection$ that adds to Q one of the new events from Λ, or the event already in A, possible to be ordered according to A_{potr}. If there does not exist any of the mentioned events, return error. Let $L^{'} = L \cup \{A_{add} \overset{Q}{\rightarrow} A_{potr}\}$, and $O^{'} = O \cup \{A_{add}\}$ and $O^{'} \cup O^{'}\{A_0 < A_{add} < A_\infty\}$ otherwise $A^{'} = A$).
4. ***Update set of events:*** Let $agenda^{'} = agenda - \{\left\langle Q, A_{potr}\right\rangle\}$. If A_{potr} is a new instance, than for each conjunction Q_i of its precondition add $\langle Q_i, A_{add}\rangle$ to $agenda^{'}$.
5. ***Protection of casual connections:*** For each operation A_t, that can threaten the casual connections $A_p \overset{R}{\rightarrow} A_c \in L^{'}$ select consistent ordered dependencies:
   - *Factorization :* Add $A_t < A_p$ to $O^{'}$
6. ***Recursive calling***:
   Planning $(\left\langle A^{'}, O^{'}, L^{'}\right\rangle, agenda^{'}, \Lambda)$

Result of the planning algorithm is the plan of partially ordered events, that considerates possible variations of the described attack.

3. PARTIALY ORDERED STATE ANALYSE

One of the main problems related to the intrusion detection of the system refers to the variability of possible attacks. It is possible to realize the same attack by many ways. Suggested IDS architecture uses the analysis of partially ordered states in a difference from the classical analysis of the transition by the states of the monitored system. In the classical scheme of the state analysis [8], the attacks are represented as a sequence of the transition states. States in the scheme of the attack correspond with the states of the system that have their boolean statement related to these states. These expressions must fulfill the conditions to realize the transition to the next state. The constituent next states are interconnected by the oriented paths that represent events or conditions for the change of the states. Such a state diagram represents the actual state of the monitored system. The change of the states considers about the intrusion as the event sequence that is realized by the attacker. These events start in the initial state and end in the final compromised state. The initial state represents the states of the system before starting the penetration. The final compromised state represents the state of the system which follows from the finished penetration. The transition of the states, that the intruder must do for the achievement of the final result of the system intrusion, are among the initial and final states. In the figure 2 there is an example of the attack that consists of four states of the attack.

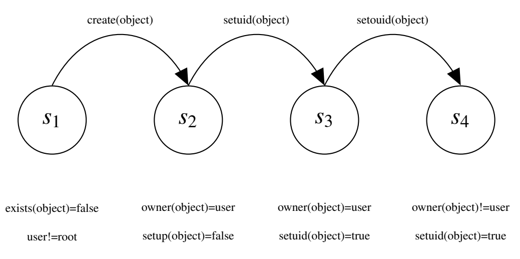

Fig. 2 States transition

Classical method of the state transition [3] stricly analyses intrusion signatures as ordered sequence of states without any chance of overlaying sequence of single events. Designed IDS architecture increases the flexibily of states analysis by using partially ordered events. Partially ordered events specify option when the events are ordered one according to another while the others are without this option of ordering. Analysis of partially ordered states enables several event sequences to form one state diagram. By using partially ordering against total ordering it is possible to use only one diagram to representation permutation of the same attack.

In the proposed architecture partially ordered state transitions are generated by partially ordered planner. Representation by partially ordered plan is more indicating according to total ordered form of states. It enables planner to put off or to ignore unnecessary ordering selection. During the state transition analysis, the number of total ordering increases exponentially with increasing the number of the states. This property of complexity coupled with total ordering is eliminated in case of partially ordered planning. Applying partially ordered notification and its property of decomposition, it is possible to deal with complex domains without any exponential complexity.

Partially ordered planner seeks state space of plans in contrast to state space of cases. The planner begins with a simple, incomplete plan that is extended in sequence by planner till it gets complete plan of solution of the problem. The operators in this process are operators on the plans: addition of steps, instructions appointing order of one step before another, and other operations. The result is final plan of order of particular states based on the dependence within these states.

The acquired representation allowes through the partly ordered plans to operate a broad range of troubleshooting domains in the planner as well as systems of intrusion detecion. The partly ordered scheme provides more exact representation of intrusion signs as the completely ordered representation, because only inevitable dependences are considered within particular events. The figure 3 is the only dependency between operations *touch* and *chmod*.

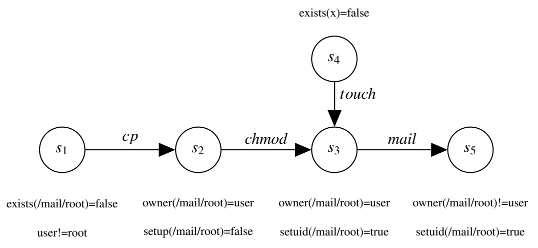

Fig. 3 Partially ordered intrusion states

Figures 2 and 3 depict same system failure of mail agent *sendmail* that provides the root account to the intruder in case of event sequence. From the figure 2 it is not clear which depedencies are necessary within single states. Whereas in the figure 3 it is clear which events forecome which.

Compromised state in the figure 3 is possible represented by the first-order logic as:

$$\begin{aligned}
&\exists\ /var/spool/mail/root\ x\\
&\quad /var/spool/mail/root \in x \wedge\\
&\quad owner(/var/spool/mail/root) = root \wedge\\
&\quad setuid(/var/spool/mail/root) = enable\\
&\quad \Rightarrow compromised(x) = true
\end{aligned}$$

or as a sequence of commands:

$$\begin{aligned}
&\exists\ file1, file2, file3, x\\
&\quad chmod(file2, 4755) \wedge\\
&\quad owner(file1) = x \wedge\\
&\quad touch(file3) \wedge\\
&\quad cp(file1, file2) \wedge\\
&\quad mail(root, file3) \wedge\\
&\quad cp(file1, file2) < chmod(file2) \wedge\\
&\quad chmod(file2) < mail(root, file3) \wedge\\
&\quad touch(file3) < mail(root, file3) \wedge\\
&\quad \Rightarrow compromised(x)
\end{aligned}$$

Proposed approach of intrusion analysis outcomes from the demand assumption of identification of minimal set of intrusion signatures and necessary depedencies within these signatures. Minimal set of signatures assumes the elimination of irrelevant signatures that do not create the intrusion. A possible example of attack, creating a link to file of different owner with different rights with consequential executing link and obtaining rigths of original owner:

1. *ls*
2. *ln*
3. *cp*
4. *rm*
5. *execute*

The first, third and fourth commands do not have an influence on the attack, tendency is to mask the attack. By elimінitation od these commands, it is possible to get minimal set describing attack together with single depedencies within events. Example in form of the first-order logic:

$$\begin{aligned}
&\exists\ file1, file2, x\\
&\quad owner(file1) \neq x \wedge\\
&\quad execute(file2) \wedge\\
&\quad owner(file2) = x \wedge\\
&\quad ln(file2, file1) \wedge\\
&\quad ln(file2, file1) < execute(file2) \wedge\\
&\quad \Rightarrow compromised(x)
\end{aligned}$$

Creation of minimal set of events necessary for creating intrusion description is fundamental problem of specification intrusion. When creating knowledge base about intrusion in a graphic form, figure 3, or in form of the first-order logic, it is possible to transform those representation into form of the petri nets.

4. CONCLUSION

The goal of the paper is to present designed IDS architecture based on partially ordered events. Intrusion detection is based on the intrusion signature matching by petri nets. Input to the system represents description of intrusion events in form of the first-order logic. This description contains mutual relationship between single intrusion events. Planning result is object of ordered events describing intrusion. In form of the petri net this object receives events from external enviroment and compares these events with signatures describing intrusion object. If an object gets to a final state that means system intrusion. Designed IDS system architecture is implemented and tested in the real enviroment.

REFERENCES

[1] Dorothy E. Denning: An intrusion-detection model, IEEE Trans. Softw. Eng., 1987

[2] Štefan Hudák: Reachability Analysis of Systems Based on Petri Nets, 1999, ISBN 80-88964-07-5

[3] Zhuowei Li and Amitabha Das and Jianying Zhou: Theoretical Basis for Intrusion Detection, 2005

[4] Vokorokos Liberios: Digital Computer Principles, 2004, p. 232, ISBN 9639548 09 X

[5] Vokorokos Liberios and Baláž Anton and Chovanec Martin: Intrusion Detection System Using Self Organizing Map, Acta Electrotechnica et Informatica, 2006

[6] Liberios Vokorokos and Alžbeta Kleinová and Ondrej Látka: Network Security on the Intrusion Detection System Level, INES 2006 10th International Conference on Intelligent Engineering Systems 2006

[7] Jingmin Zhou and Mark Heckman and Brennen Reynolds and Adam Carlson and Matt Bishop: Modeling network intrusion detection alerts for correlation,ACM Trans. Inf. Syst. Secur. 2007

[8] Rebecca Gurley Bace: Intrusion detection, Macmillan Publishing Co., Inc. 2000

BIOGRAPHIES

Liberios Vokorokos, (prof., Ing., PhD.) was born on 17.11.1966 in Greece. In 1991 he graduated (MSc.) with honours at the department of Computers and Informatics of the Faculty of Electrical Engineering and Informatics at Technical University in Košice. He defended his PhD. in the field of programming device and systems in 2000; his thesis title was ”Diagnosis of compound systems using the Data Flow applications”. He was appointed professor for Computers Science and Informatics in 2005. Since 1995 he is working as an educationist at the Department of Computers and Informatics. His scientific research is focusing on parallel computers of the Data Flow type. In addition to this, he also investigates the questions related to the diagnostics of complex systems and computer security. Currently he is a member of the State Examination Committee in the field Computing engineering and Informatics. His other professional interests include the membership on the Advisory Committee for Informatization at the faculty and Advisory Board for the Development and Informatization at Technical University of Košice.

Anton Baláž was born in Sobrance, Slovakia, in 1980. He received the engineering degree in Informatics in 2004 from Faculty of Electrical Engineering and Informatics, Technical University of Košice. Since 2004 he is PhD. student at the Department of computers and informatics FEI TUKE and his scientific research is focused on intrusion detection systems.

Norbert Ádám Norbert Ádám was born on 30.08.1980. In 1998 he graduated (MSc.) with distinction at the department of Computers and Informatics of the Faculty of Electrical Engineering and Informatics at Technical University in Košice. He defended his PhD. in the field of programming device and systems in 2007; his thesis title was ”Contribution to simulation of feed-forward neural networks on parallel computer architectures”. Since 2006 he is working as a professor assistant on the Department of Computers and Informatics. His scientific research is focusing on parallel computers architectures.

[2]Supported by VEGA 1/4071/07