ANOMALY AND MISUSE INTRUSIONS VARIABILITY DETECTION

Liberios VOKOROKOS, Anton BALÁŽ, Branislav MADOŠ
Department of Computers and Informatics, Faculty of Electrical Engineering and Informatics,
Technical University of Košice, Letná 9, 042 00 Košice, Slovak Republic,
e-mail: liberios.vokorokos@tuke.sk, anton.balaz@tuke.sk, branislav.mados@tuke.sk

ABSTRACT

In this paper we discuss our research in developing intrusion detection software framework for modeling, simulation and detection computer system intrusion based on partially ordered events and patterns - FEIIDS. The article describes problematic of intrusion detection systems and intrusions detection. We provide concrete design of developed framework based on intrusion signatures - threats are matched through Petri Nets that classify monitored system behavior and determine intrusion of monitored computer system.

Keywords: *model, itrusion, treats, detection, framework, variability*

1. INTRODUCTION

As network-based computer systems play increasingly vital roles in modern society, they have become the targets of our enemies and criminals. Therefore, we need to find the best ways possible to protect our systems. The security of a computer system is compromised when an intrusion takes place. An intrusion can be defined as any set of actions that attempt to compromise the integrity, confidentiality or availability of a resource [1]. Intrusion prevention techniques, such as user authentication, avoiding programming errors, and information protection have been used to protect computer systems as a first line of defense. Intrusion prevention alone is not sufficient because as systems become ever more complex, there are always exploitable weakness in the systems due to design and programming errors, or various socially engineered penetration techniques. For example, after it was first reported many years ago, exploitable buffer overflow still exists in some recent system software due to programming errors. The policies that balance convenience versus strict control of a system and information access also make it impossible for an operational system to be completely secure. Intrusion detection is therefore needed as another wall to protect computer systems. The elements central to intrusion detection are: resources to be protected in a target system, i.e., user accounts, file systems, system kernels, etc; models that characterize the normal or legitimate behavior of these resources; techniques that compare the actual system activities with the established models, and identify those that are abnormal or intrusive. To address present state of intrusion detection systems, this work focused on intrusion detection and system penetration variability, which can reduce time needed to evaluate potential intrusion.

This work was supported by the Slovak Research and Development Agency under the contract no. APVV-0073-07 and VEGA 1/4071/07.

2. DESIGNED INTRUSION DETECTION FRAMEWORK

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 non-determination. Designed IDS system solves this problem with planning [2] 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 its single steps that uses the matching part of the system to the intrusion detection. System framework designed on the Department of Computers and Informatics (Fig. 1).

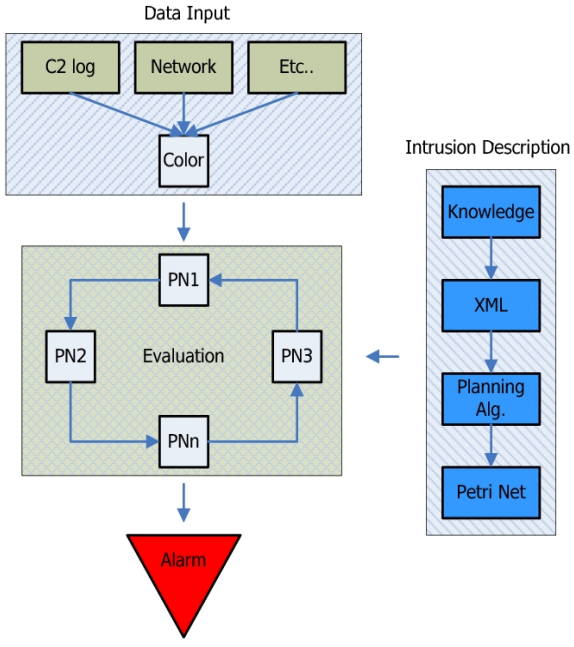

Fig. 1 FEIIDS framework

3. INTRUSIONS VARIABILITY

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 [2], 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. Fig. 2 represents example of the attack that consists of four states of the attack.

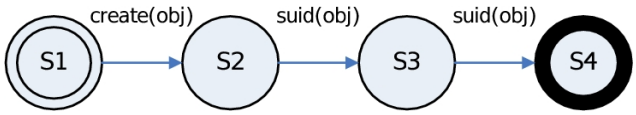

Fig. 2 States transition

Classical method of the state transition [4] strictly analyses intrusion signatures as ordered sequence of states without any chance of overlaying sequence of single events. Designed IDS architecture increases the flexibility 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 allows through the partly ordered plans to operate a broad range of troubleshooting domains in the planner as well as systems of intrusion detection. The partly ordered scheme provides more exact representation of intrusion signs as the completely ordered representation, because only inevitable dependencies are considered within particular events. Fig 3 is the only dependency between operations *touch* and *chmod*.

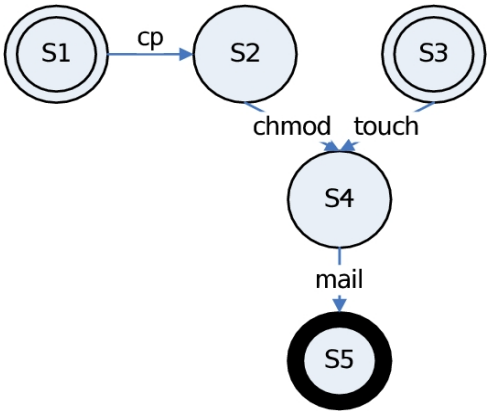

Fig. 3 Partially ordered sequence of events

Fig. 2, it is not clear which dependencies are necessary within single states. Whereas on the figure 3, it is clear which events fore come 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 the sequence of commands:

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

Proposed approach of intrusion analysis outcomes from the demand assumption of identification of minimal set of intrusion signatures and necessary dependencies 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 rights 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 elimination of these commands, it is possible to get minimal set describing attack together with single dependencies within events. Example in form of the first-order logic:

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

4. EVENTS SEQUENCE PLANNING

Intrusion is defined as a set of events with a focus on compromise integrity, confidentiality and resources availability. 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 environment state with information depending on the final result of events.

State transition is characterized as a sequence of events performed by intruders leading from initial state do final compromised state. Planning can be formulated 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 follows:

1. Set of single steps of the plan. Every step represents control activity of the plan.
2. Set of ordered dependencies. Every dependency is in a form of $S_i \prec S_j$, 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.

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

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 consideration of consistence requirements defined in the O set. The key step of this activity is to preserve states of the past conclusions and requirements 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 A_p and the A_c precondition. The expression 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 endangers $A_p \xrightarrow{Q} A_c$ if:

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

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

4.1. Planning algorithm

The core of the planning is represented by the algorithm of planning (Fig. 4), mentioned below, that searches the state environment of the plans. The algorithm begins with a blank plan and performs non-deterministic selection of the event sequence in stages, till all the preconditions are considerate through their casual

$$\begin{aligned}&Planning(\{A, O, L\}, agenda, \Lambda)\\ &1.\ End: If\ agenda\ is\ empty, return\ \{A, O, L\}\\ &2.\ Target\ Selection: Let\ \{Q, A_p\}\ be\ a\ pair \in agenda, A_p \in A\\ &3.\ Event\ Selection: Let\ A_p = be\ event\ selection\ that\ adds\ Q\ one\ of\ the\ new\ events\ from\ \Lambda\\ &\vee\ event\ already \in A\ that\ is\ possible\ to\ order\ regard\ to\ A_p.\\ &Let\ L' = L \cup \{A_{add} \Rightarrow A_p\}, let\ O' = O \cup A_{add} \wedge O' \cup O'\ \{A_0 < A_{add} < A_\infty\}\\ &4.\ Update\ Set\ of\ Targets: Let\ agenda' = agenda - \{Q, A_p\}\\ &5.\ Recursive\ Call: Planning(\{A', O', L'\}, agenda', \Lambda)\end{aligned}$$

Fig. 4 Planning algorithm

connections and till potential threads of the plan are eliminated. Partially ordered dependencies of the final plan are over again represented by only partially 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.

4.2. Events evaluation

Petri Nets represent automates based on events and conditions. Events are actions that are executed and their existence is controlled by system states. Every system state represents set of conditions and their values. In the proposed IDS system, there are these sets in form of first-order expressions presenting fulfilled or not fulfilled conditions. Some of the events only occur on specific conditions where state description represents preconditions for those events. Presence of specific events may terminate validity of one precondition and setup validity of other one. Each intrusion is in the proposed IDS system represented by a Petri Net. Petri Net places represent states or pre - post events conditions. Input for Petri Net creation is plan of partially ordered events forming intrusion. Petri Net transitions correspond with characteristic event pattern. Detection architecture evaluates single intrusions in form of Petri Nets evaluating input events of miscellaneous input data. Initial states represent initial system states and final state represents state that implies intrusion.

5. CONCLUSION

Information technology security nowadays presents one of the main priorities of modern society dependent on information. Protection of data access, availability, and integrity represents basic security properties insisted on information sources. Intrusion of one of the properties mentioned above may form penetration or attack on the computer system. One of the main problems of intrusion detection is potential attack variability. From the detection perspective, generation of exact intrusion attribute sequence is deficient. The property of attack sequence non-determination is not be described by the entire sequence of events forming intrusion. One of the goals of this work was to solve attack variability mentioned above. Upon attribute properties and their context research, classificatory hierarchy describing mutual references within attributes and events was formed. The aim of this work was to introduce designed intrusion detection software framework based on partially-ordered events and patterns. Developed method identities possible system intrusions by means of monitoring computer system state patterns. Individual states of the monitoring system are described through performed events, and individual dependencies within the performed events. The resultant detection model is realized by the Petri Nets. Upon designed IDS system framework, system prototype was implemented, tested and verified - FEIIDS. From test results, functionality and practical usability of designed IDS architecture is resulted.

The work is one of reached results within projects VEGA 1/4071/07 (Security architecture of heterogeneous distributed and parallel computing system and dynamical computing system resistant against attacks), and APVV 0073-07 (Identification methods and analysis of safety threats in architecture of distributed computer systems and dynamical networks) being solved at Department of Computers and Informatics, Faculty of Electrical Engineering and Informatics, Technical University of Košice.

ACKNOWLEDGMENTS

Supported by a grant from Iceland, Liechtenstein and Norway through the EEA Financial Mechanism and the Norwegian Financial Mechanism. This project is also co-financed from the state budget of the Slovak Republic.

REFERENCES

[1] DENNING, D. E.: An Intrusion-Detection Model, *IEEE Trans Softw. Eng.*, 1987.

[2] BACE, R. G.: Intrusion Detection, *Macmillan Publishing Co., Inc.*, 2000.

[3] ZHOU, J. – HECKMAN, M. – REYNOLDS, B. – CARLSON, A. – BISHOP, M.: Modeling Network Intrusion Detection Alerts for Correlation, *ACM Trans. Inf. Syst. Secur.*, 2007.

[4] DUAN XUETAO – ZHONG ANMING – LI YING – JIA CHUNFU: Intrusion Detection Method for Program Vulnerability via Library Calls, Wuhan University Journal of Natural Sciences, 12(1):126–130, January 2007.

[5] TENG, H. S. – CHEN, K. – C-Y. LU, S.: Security Audit Trail Analysis Using Inductively Generated Predictive Rules, In Proceedings of the sixth conference on Artificial Intelligence Applications, pages 24–29, Piscataway, NJ, USA, 1990, IEEE Press.

[6] SINGHAL, A.: Data Mining for Intrusion Detection. In Data Warehousing and Data Mining Techniques for Cyber Security, Volume 31, pages 59–67, Springer US, 2007.

[7] MACCABE, A. – HEADY, R. – LUGER, G. – SERVILLA, M.: The Architecture of a Network Level Intrusion Detection System, Technical report, Department of Computer Science, University of New Mexico, 1990.

[8] YOSHINORI OKAZAKI – IZURU SATO – SHIGEKI GOTO: A New Intrusion Detection Method Based on Process Profiling, In SAINT '02: Proceedings of the 2002.

[9] Symposium on Applications and the Internet, pages 82–91, Washington, DC, USA, 2002, IEEE Computer Society.

[10] LEE, W. – STOLFO, S. J.: A Framework for Constructing Features and Models for Intrusion Detection Systems, ACM Trans. Inf. Syst. Secur., 3(4):227–261, 2000.

[11] ILGUN, K. – KEMMERER, R. A. – PORRAS, P. A.: State Transition Analysis: A Rule-Based Intrusion Detection Approach, Software Engineering, 21(3):181–199, 1995.

Received July 8, 2010, accepted November 16, 2010

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. Currently he is dean of the Faculty of Electrical Engineering and Informatics at the Technical University of Košice. 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 (Ing., PhD.) was born in Sobrance, Slovakia, in 1980. He received the master degree in Informatics in 2004 from Faculty of Electrical Engineering and Informatics, Technical University of Košice. In 2008 he received PhD. in area of computer security. Since 2007 he is working as professor assistant at the Technical University of Košice.

Branislav Madoš (Ing., PhD.) was born on 20.5.1976 in Trebišov, Slovakia. In 2006 he graduated (MSc.) with distinction at the Department of Computers and Informatics at the Faculty of Electrical Engineering and Informatics of the Technical University of Košice. He defended his PhD. in the field of Computers and computer systems in 2009; his thesis title was "Specialized architecture of data flow computer". Since 2010 he is working as a professor assistant at the Department of Computers and Informatics. His scientific research is focusing on the parallel computer architectures and architectures of computers with data driven computational model.