2

Architecture of Computer Intrusion Detection Based on Partially Ordered Events

Liberios Vokorokos and Anton Baláž
Technical University of Košice
Slovak Republic

1. Introduction

Information technologies became part of our daily life. Nowadays, contemporary society is dependent on functioning of miscellaneous information systems providing daily community motion. The attack aim is often to disrupt, deny of service or at least one of its parts required for proper functionality, or to acquire unauthorized access to information [Vokorokos (2004)]. Nowadays, solid system assecuration becomes one of the main priorities. Basic way of protection is realized through specialized devices firewalls allowing to define and control permitted communications in boundary parts of computer network or between protected segments and surrounding environment. Present firewalls often detect some unauthorized attack activities but their functionality is limited. Unauthorized intrusion detection systems allow increase of information systems security against attacks from the Internet or organization intranet, by means of passive inform about arising intrusion or active interfere against defecting intrusion.

The existing intrusion detection approaches can be divided in two classes - anomaly detection and misuse detection [Denning (1987)]. 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(IDS) can be categorized into three types - host-based IDS, network-based IDS and hybrid IDS [Bace (2000)]. 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 non-deterministic nature where that same intrusion or attack can be realized through various permutations of the same events. The purpose of this paper is to present authors' proposed intrusion detection architecture based on the partially ordered events and the Petri nets.

Project is proposed and implemented at the Department of Computers and Informatics in Košice supported by VEGA 1/4071/07. (Security architecture of heterogeneous distributed and parallel computing system and dynamical computing system resistant against attacks) a APVV 0073-07 (Identification methods and analysis of safety threats in architecture of distributed computer systems and dynamical networks).

2. State of art

Several intrusion detection systems were designed and implemented till today. Most of these systems are based on statistical methods derived from work of Denning [Denning (1987)]. Some of them, as source of information, use log system of operation system [Anderson et al. (1995)]. Other one, as input data, use network traffic [Zhang et al. (2003)] [Spirakis et al. (1994)] [Servilla (1990)]. Systems, as MADIDS [Guangchun et al. (2003)], extend this network traffic with distribution of intrusion data within single analyzing network systems that perform partial intrusion detection. Among systems not working with statistical methods, there can be inserted system of authors [Teng et al. (1990)] that analyzes single user events and tries to find mutual relations among them. IDS architectures based on misuse detection are systems, as [Ilgun et al. (1995)] [Ilgun (1993)], that search for already known intrusions, derived state of intrusion based on present system state.
According to present state of intrusion detection systems, this work is focused on intrusion detection and system penetration variability, which can reduce time needed to evaluate potential intrusion.

3. Architecture of designed IDS system

Proposed system architecture includes part of planning and matching, figure 1. 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 [Russell & Norvig (2003)] 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 provided by Petri Net automata. System architecture designed on the Department of Computers and Informatics is on the figure 1.

4. Partially ordered state analysis

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 [Axelsson (2000)], 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

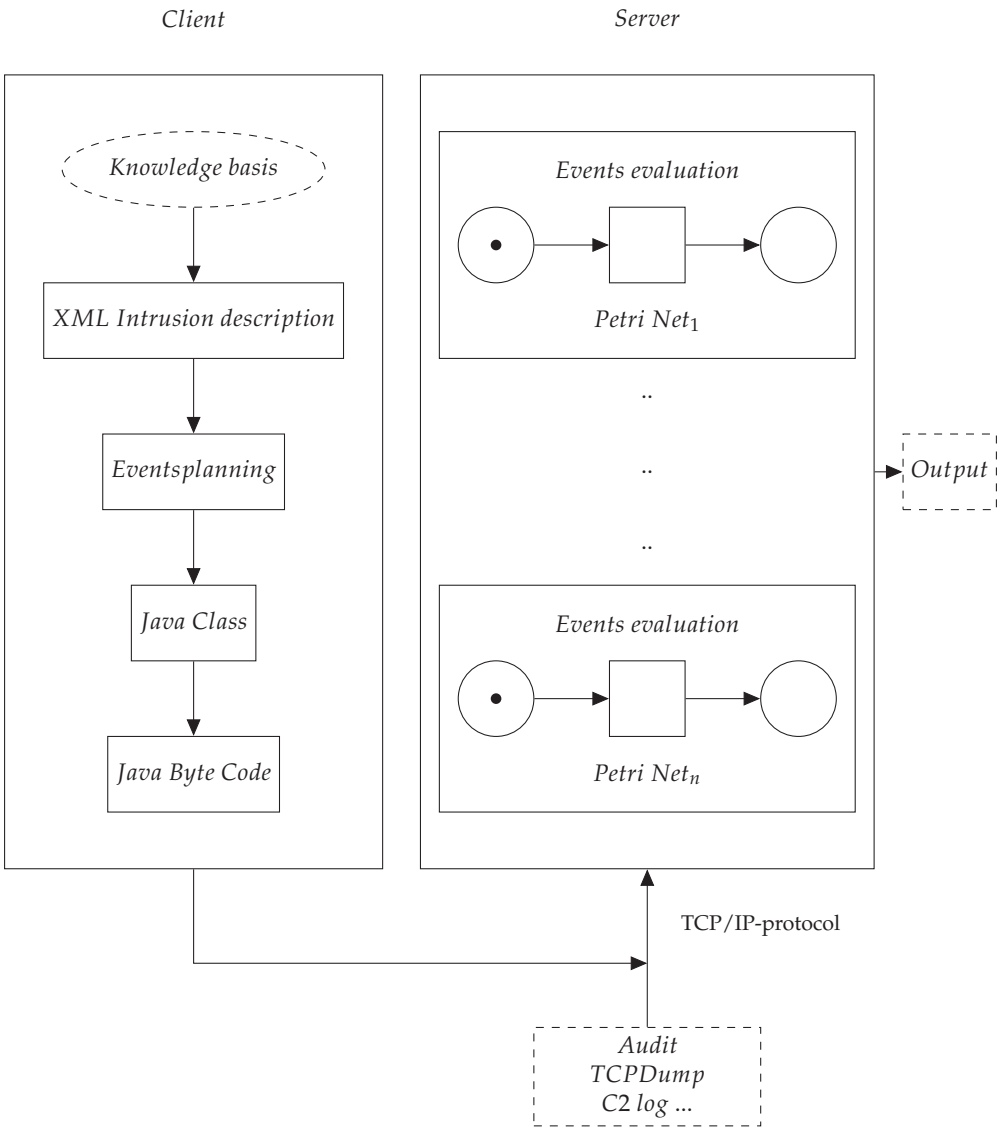

Fig. 1. Architecture of Designed IDS System

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.

Classical method of the state transition [Anderson (1980)] strictly analyzes intrusion signatures as ordered sequence of states without any chance of overlaying sequence of single

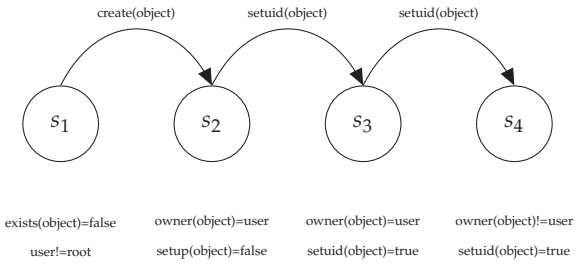

Fig. 2. State Transition Diagram

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 patterns as the completely ordered representation, because only inevitable dependencies are considered within particular events. figure 3 is the only dependency between operations touch and chmod.

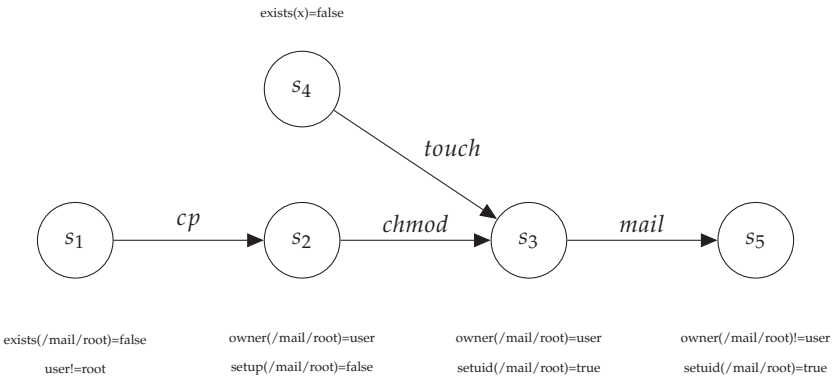

Fig. 3. Partially Ordered Intrusion States

In the figure 2, it is not clear which dependencies are necessary within single states. Whereas in 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:

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

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:

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

5. Intrusion signature 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:

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 $S_i < S_j$, 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. Partially ordered planning allows representing plans in which some steps are ordered according to other steps. Intrusion signatures and their nature of non-determination 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 conditions of these events in the form of first-order logic. The planner generates set of events and their dependencies for each initial and final intrusion state. Furthermore, knowledge base includes state dependencies for each event signature. This

information is 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

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

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 bindings stored in between them. These bindings are necessary for partially event ordering. An ordering result is represented by set of events and their dependencies in between these event signatures. Let intrusion sequence to consist of n event signatures: $SA_1, SA_2, \ldots, SA_n$, then intrusion structure is specified as

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

First part of this term $\{SA_1, SA_2, \ldots, SA_k\}$ is a set of event signatures. Next part of the term is ordering dependency 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. Further, 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.

5.1 Events planning

This session 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 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 the 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 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 considered by the planner during assembling the final plan. The algorithm can add supplementary order dependencies by assurance of performance of A_t before the A_p. 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 performs non-deterministic selection of the event sequence in stages, till all the preconditions are considered 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 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.

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

1. ***Completion:***If the agenda is empty, return $\langle A, O, L \rangle$.
2. ***Target selection:*** $\langle Q, A_{need} \rangle$ is a pair in the agenda (according $A_{need} \in A$ and Q is conjunction of preconditions A_{need}).
3. ***Event selection:*** $A_{need} = event\ selection$ that adds to Q one of the new events from Λ, or the event already in A, possible to be ordered according to A_{need}. If there does not exist any of the mentioned events, return error. Let $L^{'} = L \cup \{A_{add} \xrightarrow{Q} A_{need}\}$, 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 $agenda^{'} = agenda - \{\langle Q, A_{need} \rangle\}$. If A_{need} 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 $A_p \xrightarrow{R} A_c \in L^{'}$ 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.

5.2 Events evaluation

Petri Nets represent automatas 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.

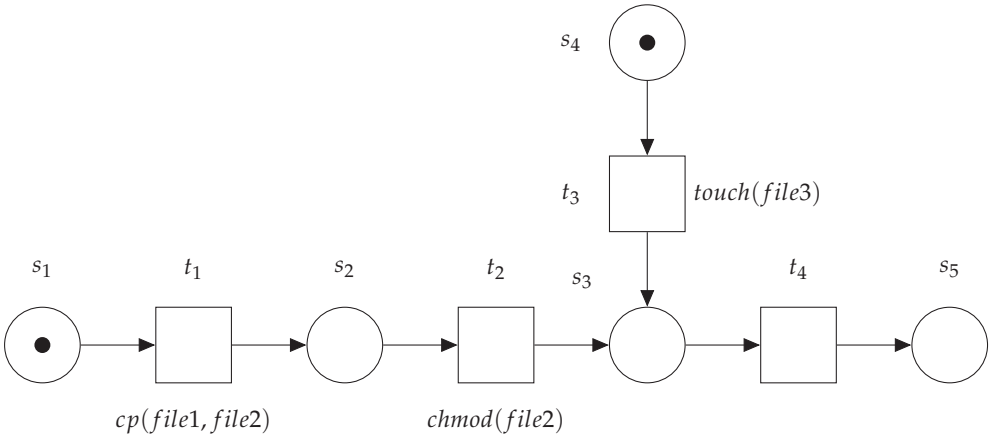

Fig. 4. Petri Net Intrusion Example

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.

6. Experimental validation of proposed IDS

Presented architecture of IDS system is implemented in Java. The goal of this implementation is to generate a uniform set of classes that can be used for general generation of IDS. By the designed architecture, there exist two critical points of the system that affect the efficiency of entire sample intrusion evaluation. Given points:

1. Time needed for capture and generation of instance of an input event into the object of Java language
2. Time needed for evaluation of the single intrusion represented through the Petri net

On the basis of these two critical points, there were assembled and executed following experiments consisting of network attacks:

Experiment 1 Time needed for generation of input event object and saving already processed input events on the list. The experiment was performed in the network environment with various types of attacks on the TCP/IP protocol. An input event flow included 2500 (5000) packets. In order to elimination of possible external influences and to achieve more objective results, the test was performed with 300 iterations.

The testing environment consists of various computer systems mentioned in table 1. Designed IDS system presents type of host IDS, but from the implementation perspective, single input objects of input events are transferred though the TCP/IP protocol on basis of Client-Server type, where both Server (accepts and handles information) and Client (sends input objects

for evaluation) operate. Therefore, tests containing experiments performed in the network environment on basis of Ethernet were added to testing formation as well. The test results of single configurations are in figures 5 and 6. Average times needed for instance of one input event generation are mentioned in tables 2 and 3 according to:

$$\overline{Time} = \frac{Time\ for\ creating\ (2500\ or\ 5000)\ packets}{2500\ or\ 5000}[ms] \tag{1}$$

Number	Configuration
1.	AMD Duron 800MHz, 512MB SDRAM
2.	Intel Celeron 2.4GHz, 512MB DDRAM
3.	AMD Sempron 2.0Ghz, 512MB DDRAM
4.	Intel P4 2.4GHz HT, 1GB DDRAM
5.	AMD Opteron 2.21GHz, 1GB DDRAM
6.	Ethernet 100Mbit
7.	Ethernet 1000Mbit

Table 1. Testing Configuration of Computer Systems

Description	Average time 2500 packets [ms]
Ethernet 1000Mbit	0,200496
Ethernet 100Mbit	0,815273469
Intel Celeron	0,2646
Amd Duron	1,712097959
Amd Opteron	0,194256
Intel Pentium 4	0,813512
Amd Sempron	0,274432653

Table 2. Average Time Need for Generation of One Instance of Input Event

Description	Average time 5000 packets [ms]
Ethernet 1000Mbit	0,229192
Ethernet 100Mbit	0,94105
Intel Celeron	0,496812
Amd Duron	0,8239
Amd Opteron	0,158636
Intel Pentium 4	0,578310204
Amd Sempron	0,409856

Table 3. Average Time Need for Generation of One Instance of Input Event

Experiment 1 was focused on speed of transformation flow of input events into object instance at Java language. Within simulation, it was detected that the best results are provided by performance the speediest platform AMD Opteron and the weakest performance from the set of testing systems is provided by AMD Duron. To consider input event transfer and transformation into input object, the transfer of packets through the TCP/IP protocol is the

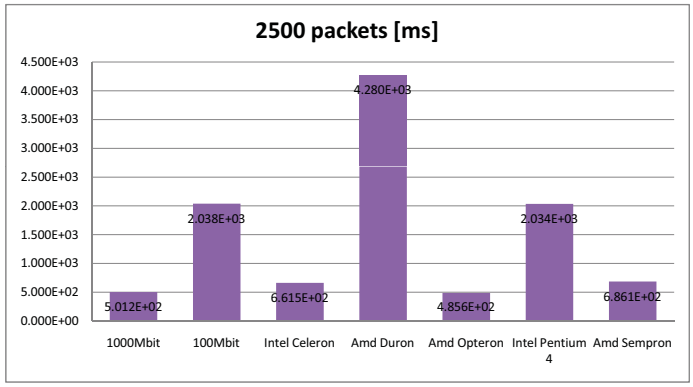

Fig. 5. Results of Experiment 1

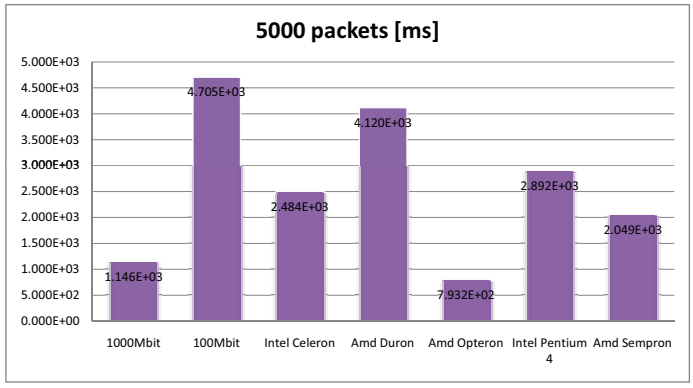

Fig. 6. Results of Experiment 1

most decisive factor. This determination results from comparison of the speediest platform AMD Opteron results and data transfer in Ethernet 1000Mbit network type, where the results of these two simulations are very similar.

Experiment 2 Time needed for attack evaluation at various arithmetic of attacks evaluated at the same time. Time is measured by the object of attack description transfer period till the final time of attack detection.

The experiment was performed on the same configurations mentioned in table 1. The amount of tested attacks is in range of 1 to 20 attacks evaluated at the same time. The testing input flow contains 2500 (5000) packets including packets generating attack. For more objective results acquirement, single tests were performed 300 times repeatedly. The acquired results are displayed in graphs 7, 8, 9, 10 and 11 Summary of experiment 2 results is displayed in graphs 12 and 13.
Results of the simulation experiments were realized on the group of various performance platforms. In order to test performance, the system was implemented in Java language. Development environment was IDE Eclipse, operation system MS Windows XP and MS

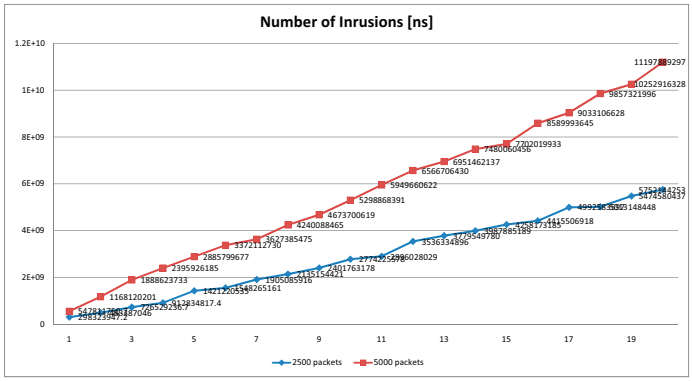

Fig. 7. Experiment 2 - Intel Celeron

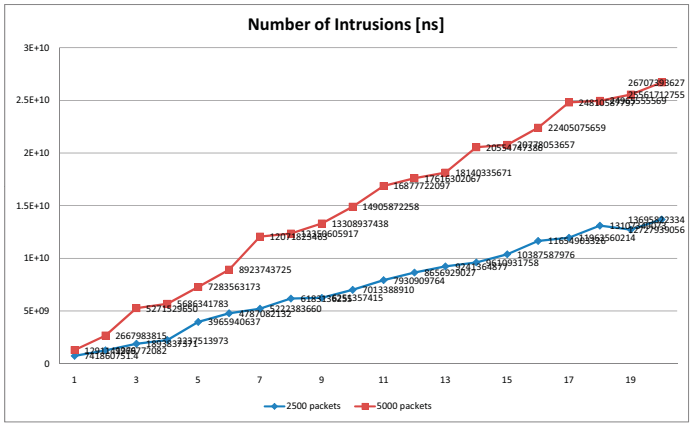

Fig. 8. Experiment 2 - AMD Duron

Windows 2003. The experiments were performed in order to performance evaluation. On the same purpose, a special group of attacks was created, focused on the limitations of the TCP/IP protocol. Single tests were executed 300 times repeatedly in order to elimination of possible fault in case of single measuring.
The results achieved during experiments mean:

- Officially, the most efficient platform AMD Opteron provides better results. The more efficient computing performance, the less the time needed for evaluation.
- At single systems (loopback), the inner interface provides approximately the same permeability as the Ethernet 1000Mbit network.
- Time needed for evaluation of the rising amount of attacks evaluated at the same time, rises linear.

On the basis of the results of the experiments, decisive and main factor of the entire designed architecture is memory subsystem of the tested computer system.
Less affecting speed factors of the architecture:

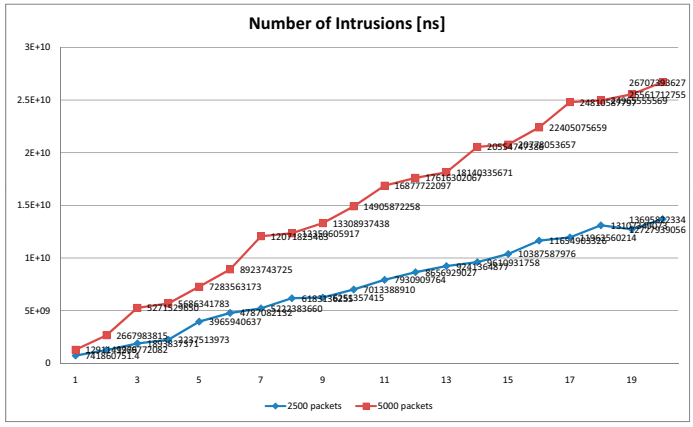

Fig. 9. Experiment 2 - AMD Opteron

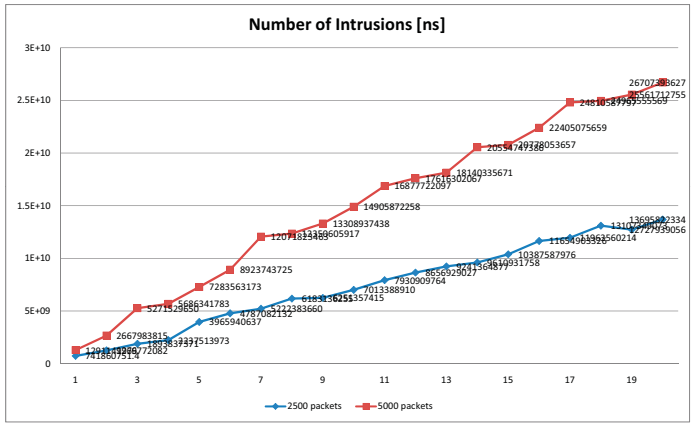

Fig. 10. Experiment 2 - Intel Pentium 4

- *Performance of CPU.* Faster processor presents faster evaluation of input flow. Cooperation customization of the memory subsystem and the processor presents narrow effectiveness socket of the entire architecture.
- *Faster logging system.* More effective retrieval of the input event means continuous processing of objects by the evaluation unit without waiting for write and read. Customization of the logging system and its effectiveness means another important effectiveness role of the entire system.
- *More effective data structures.* The system was designed during its implementation in regard of general IDS, with possibility of another expansion and specification. Effectiveness of some used data structures does not have to be optimal and it requires its profilation in order to force the entire functionality to be more effective.

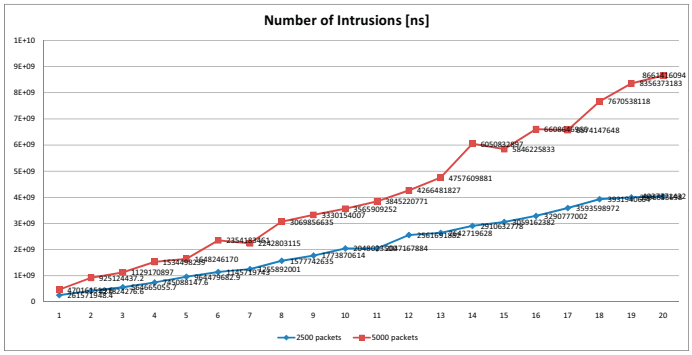

Fig. 11. Experiment 2 - AMD Sempron

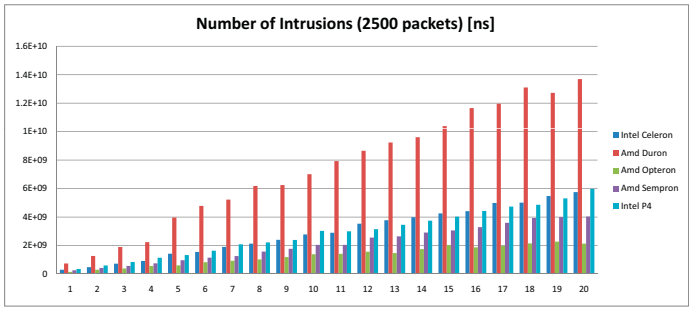

Fig. 12. Experiment 2 - Summary 2500 packets

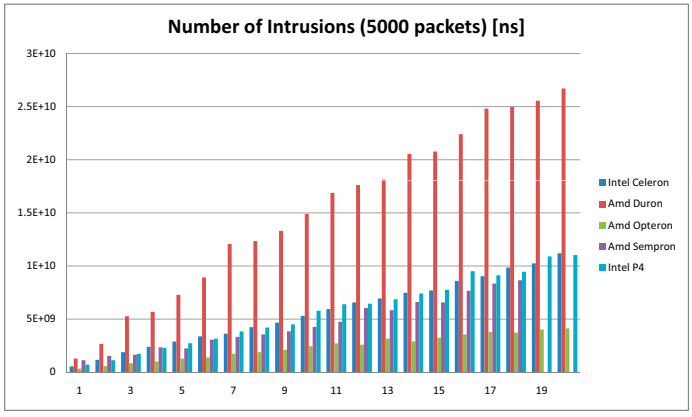

Fig. 13. Experiment 2 - Summary 5000 packets

7. 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. Within protection mechanisms, various methods providing security rules relating individual levels of possible behavior are classified. Other protection mechanisms are diversified systems detecting suspicious behavior. Intrusion detection systems belong to these systems as well.
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. Analysis result is a new method of penetration representation in form of partially-ordered events scheme enabling generation of some dependencies only, within the whole set of events describing intrusion. The resultant event scheme is transformed into Petri Nets that evaluate input event flow and detect possible attributes of represented intrusions.
The aim of this work was to introduce designed intrusion detection architecture based on partially-ordered events and patterns. The main work goal was production of intrusion detection and its alternatives method. Produced 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 architecture, system prototype was implemented and tested. From test results, functionality and practical usability of designed IDS architecture is resulted. From experimentation conclusions, interactivity of central processing unit CPU and memory subsystem is the determining factor with influence on entire intrusion detection effectiveness. The resultant system represents live system with possibility of dynamic addition and removal of other detected intrusions.
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.

8. References

Anderson, J. P. (1980). Computer security threat monitoring and surveillance, *Technical report*, James P. Anderson Co.

Anderson, Lunt, Javits, Tamaru & Valdes (1995). Detecting unusual program behavior using the statistical components of nides.
URL: *http://www.csl.sri.com/papers/5sri/*

Axelsson, S. (2000). Intrusion detection systems: A survey and taxonomy, *Technical Report 99-15*, Chalmers Univ.
URL: *citeseer.comp.nus.edu.sg/axelsson00intrusion.html*

Bace, R. G. (2000). *Intrusion detection*, Macmillan Publishing Co., Inc., Indianapolis, IN, USA.
Denning, D. E. (1987). An intrusion-detection model, *IEEE Trans. Softw. Eng.* **13**(2): 222–232.
Guangchun, L., Xianliang, L., Jiong, L. & Jun, Z. (2003). Madids: a novel distributed ids based on mobile agent, *SIGOPS Oper. Syst. Rev.* **37**(1): 46–53.
Ilgun, K. (1993). USTAT: A real-time intrusion detection system for UNIX, *Proceedings of the 1993 IEEE Symposium on Research in Security and Privacy*, Oakland, CA, pp. 16–28.
URL: *citeseer.ist.psu.edu/ilgun92ustat.html*
Ilgun, K., rd A. Kemmerer & Porras, P. A. (1995). State transition analysis: A rule-based intrusion detection approach, *Software Engineering* **21**(3): 181–199.
URL: *citeseer.ist.psu.edu/ilgun95state.html*
Russell, S. & Norvig, P. (2003). *Artificial Intelligence: A Modern Approach*, 2. edn, Prentice-Hall, Englewood Cliffs, NJ.
Servilla, R. H. G. L. A. M. M. (1990). The architecture of a network level intrusion detection system, *Technical report*, Department of Computer Science, University of New Mexico.
Spirakis, P., Katsikas, S., Gritzalis, D., Allegre, F., Darzentas, J., Gigante, C., Karagiannis, D., Kess, P., Putkonen, H. & Spyrou, T. (1994). SECURENET: A Network Oriented Intrusion Prevention and Detection Intelligent System, *Proceedings of the 10th International Conference on Information Security, IFIP SEC94*, The Netherlands.
Teng, H. S., Chen, K. & Lu, S. C.-Y. (1990). Security audit trail analysis using inductively generated predictive rules, *Proceedings of the sixth conference on Artificial intelligence applications*, IEEE Press, Piscataway, NJ, USA, pp. 24–29.
Vokorokos, L. (2004). *Digital Computer Principles*, Typotex, Budapest.
Zhang, Y., Lee, W. & Huang, Y.-A. (2003). Intrusion detection techniques for mobile wireless networks, *Wirel. Netw.* **9**(5): 545–556.

Petri Nets Applications

Edited by Pawel Pawlewski

ISBN 978-953-307-047-6
Hard cover, 752 pages
Publisher InTech
Published online 01, February, 2010
Published in print edition February, 2010

Petri Nets are graphical and mathematical tool used in many different science domains. Their characteristic features are the intuitive graphical modeling language and advanced formal analysis method. The concurrence of performed actions is the natural phenomenon due to which Petri Nets are perceived as mathematical tool for modeling concurrent systems. The nets whose model was extended with the time model can be applied in modeling real-time systems. Petri Nets were introduced in the doctoral dissertation by K.A. Petri, titled "„Kommunikation mit Automaten" and published in 1962 by University of Bonn. During more than 40 years of development of this theory, many different classes were formed and the scope of applications was extended. Depending on particular needs, the net definition was changed and adjusted to the considered problem. The unusual "flexibility" of this theory makes it possible to introduce all these modifications. Owing to varied currently known net classes, it is relatively easy to find a proper class for the specific application. The present monograph shows the whole spectrum of Petri Nets applications, from classic applications (to which the theory is specially dedicated) like computer science and control systems, through fault diagnosis, manufacturing, power systems, traffic systems, transport and down to Web applications. At the same time, the publication describes the diversity of investigations performed with use of Petri Nets in science centers all over the world.

How to reference

In order to correctly reference this scholarly work, feel free to copy and paste the following:

Liberios Vokorokos and Anton Balaz (2010). Architecture of Computer Intrusion Detection Based on Partially Ordered Events, Petri Nets Applications, Pawel Pawlewski (Ed.), ISBN: 978-953-307-047-6, InTech, Available from: http://www.intechopen.com/books/petri-nets-applications/architecture-of-computer-intrusion-detection-based-on-partially-ordered-events

INTECH
open science | open minds

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821