Available online at www.sciencedirect.com

SciVerse ScienceDirect

Procedia Computer Science 10 (2012) 968 – 975

Procedia Computer Science

The 2nd International Symposium on Frontiers in Ambient and Mobile Systems (FAMS)

An Efficient Formal Framework for Intrusion Detection Systems

Mohsen Rouached[ab*], Hassen Sallay[b]

[a]*College of Computers and Information Technology, Taif University, Taif, Saudi Arabia*
[b]*Scientific and Technology Unit, AMAN Security Team, IMAM University, Riyadh, Saudi Arabia*

Abstract

Traffic anomalies and attacks are commonplace in today's networks, and identifying them rapidly and accurately is critical for large network operators. Intrusion detection systems are an important component of defensive measures protecting computer systems and networks from abuse.
For an intrusion detection system, it is important to detect previously known attacks with high accuracy. However, detecting previously unseen attacks is equally important in order to minimize the losses as a result of a successful intrusion. It is also equally important to detect attacks at an early stage in order to minimize their impact. To address these challenges, this paper proposes to improve the efficiency of the network intrusion detection process by including an Event Calculus based specification to detect the registered and expected behaviour of the whole network.

Keywords: Intrusion detection systems; High Speed Networks; Formal Specification; Verification and Validation.

1. Introduction

Detecting intrusions in networks and applications has become one of the most critical tasks to prevent their misuse by attackers. The cost involved in protecting these valuable resources is often negligible when compared with the actual cost of a successful intrusion, which strengthens the need to develop more powerful intrusion detection systems.

* Corresponding author. Tel.: +966-272-720-20; fax: +966-272-742-99
E-mail address: m.rouached@tu.edu.sa.

1877-0509 © 2012 Published by Elsevier Ltd. Open access under CC BY-NC-ND license.
doi:10.1016/j.procs.2012.06.132

Today, intrusion detection is one of the high priority and challenging tasks for network administrators and security professionals. For an intrusion detection system, it is important to detect previously known attacks with high accuracy. However, detecting previously unseen attacks is equally important in order to minimize the losses as a result of a successful intrusion. It is also equally important to detect attacks at an early stage in order to minimize their impact. The major challenges and requirements for building intrusion detection systems are:

1. Ability to detect attacks reliably without giving false alarms. The challenge is to build a system which can detect a wide variety of attacks and at the same time which results in very few false alarms.
2. Ability to handle large amount of data without affecting performance and without dropping data. The challenge is to prevent an attack rather than simply detecting it.
3. Ability to link an alert generated by the intrusion detector to the actual security incident is desirable. It is not only necessary to detect an attack, but it is also important to identify the type of attack.

In this context, we present, in this paper, an event calculus (EC) based approach to formally analyze the process of detecting network intrusions. The proposed approach checks that security requirements and assumptions are preserved at run-time by monitoring the satisfaction of EC formulas that formalize them using the detection rules.

The rest of the paper is organized as follows. Section 2 depicts the detection framework to be developed. In Section 3, we introduce the main ingredients of our monitoring approach. We also detail the encodings of the different ingredients. The related work is presented in Section 5. Finally, Section 6 concludes the paper.

2. Detection Framework

Currently, the major objective of our research group[†] is to design and develop an efficient real-time Network Intrusion Detection System (NIDS) for High Speed Networks (HSN). We believe that this could be satisfied through the fulfillment of four main tasks; (1) Design and development of an underlying scalable and adaptive parallel and distributed IDS architecture for High Speed Network (2) Design and development of algorithms and techniques improving the Accuracy of NIDS alerts generation and correlation and real-time malicious attack detection by minimizing the false alerts. (3) Design and development of an efficient and integrated management platform coupling these aforementioned algorithms and techniques to the underlying architecture. (4) Testing and performance study of the system and simulation for large scale scenarios. In this context we developed the $(\varphi|\pi)$ IDS Framework [1]. The framework has four research themes pillars. The first one is *modeling the brain* theme which is targeting to improve the accuracy of detection and attack alert prioritization [2]. The second one is *making the mind* which aims to add some intelligence to the system by introducing a type of reasoning on the alert logs to discover new attack scenarios and therefore make the detection process more efficient The *architectural theme* targets to design distributed architectures performing adaptive traffic load balancing algorithms and splitting schemes to take over the HSN bottleneck caused by the IDS scanning tasks inside the network [3]. Finally *the management theme* tends to manage efficiently the overall self-defense process [4]. In this paper, we focus on the detection process and propose the detection framework depicted in figure 1. The detection model consists of the following elements:

- **Security Requirements**: define properties that should be satisfied to guarantee the security of the network.

[†] http://www.amansystem.com

- **Assumptions**: are additional properties injected to facilitate the verification process. Security properties that we are not sure of and more important, properties that cannot be efficiently monitored will be declared as assumptions. One kind of the verification consists of checking security properties of IDSs together with assumptions, with respect to security policies. Security policies are always satisfied with sufficiently strong assumptions.

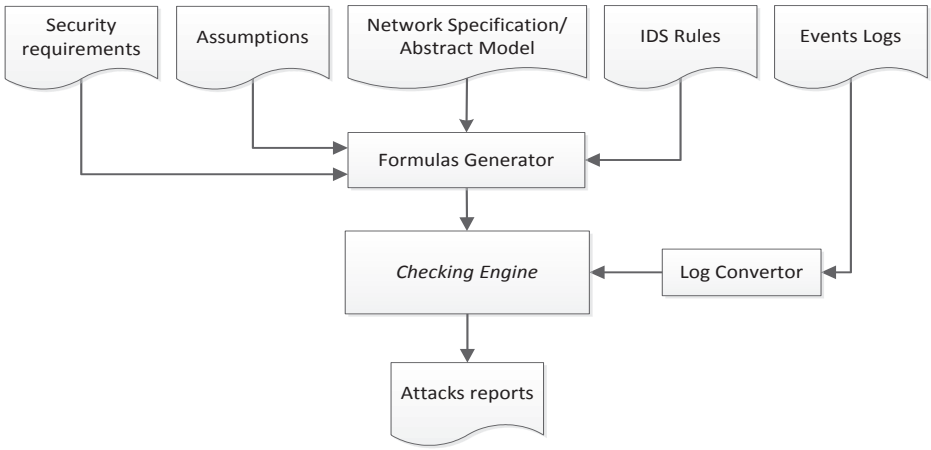

Fig. 1. Detection framework

- **Network specification**: gives an abstract model for security critical entities of the network. One way to develop a specification for a program is to first identify what operations and accesses the program needs to support its functionality. Based on an examination of the code or its behaviours, one writes rules in the specification to cover the valid operations of the program. Writing specifications for a program is subtle and tricky, thus demanding an approach to rule validation.

- **IDS rules**: vary dependent on the IDS. In our case, we will focus on Snort rules. Little research has been done on analyzing intrusion-detection rules. Different approaches [5, 6, 7] have been taken to specify and analyze the intrusion signatures and detection rules primarily for signature-based IDSs.

- **Event logs**: are necessary for the model because almost all IDSs are based on the analysis of the audit trails from operating systems, applications and network components.

- **Verification engine**: The verification can be done either a-priori, i.e., at design time, or a-posteriori, i.e., after runtime to test and repair design errors, and formally verify whether the process design does have certain desired properties. The need for a-priori verification is important for the specification because it can be very complex, and therefore we need to check if the specified behaviour is consistent. The a-posteriori verification is also important to provide knowledge about the context of and the reasons of discrepancies between abstract models and related instances. This kind of verification is necessary since some components that constitute the network may be dynamically specified at runtime, causing unpredictable interactions with other components, and making the a-priori verification method insufficient as it only takes into account static aspects.

3. An event driven model for intrusion detection

In this section, we introduce the formal logic to be used, the ingredients of encoding the different framework components, the verification strategies, and the analysis of the attacks reports.

3.1. Event Calculus

Event calculus (EC) [8] is a formal language for representing and reasoning about dynamic systems, whose ontology consists of (i) a set of time-points isomorphic to the non-negative integers, (ii) a set of time-varying properties called fluents, and (iii) a set of event tpes (or actions). The logic is correspondingly sorted, and includes the predicates *Happens*, *Initiates*, *Terminates* and *HoldsAt*, as well as some auxiliary predicates defined in terms of these. *Happens(a,t)* indicates that event (or action) a actually occurs at time-point t. *Initiates(a,f,t)* (resp. *Terminates(a,f,t)*) means that if event a were to occur at t it would cause fluent f to be true (resp. false) immediately afterwards. *HoldsAt(f,t)* indicates that fluent f is true at t.

The use of EC to specify security policies and requirements was discussed and proved in several papers such as [9, 10, 11]. [12] presented a method for transforming both policy and system behaviour specifications into a formal notation that is based on Event Calculus. Additionally it describes how this formalism can be used in conjunction with abductive reasoning techniques to perform a priori analysis of policy specifications for the various conflict types identified in the literature.

3.2. Expressing the detection framework components

Security requirements, assumptions, IDS rules, and the system specification are expressed using EC predicates and axioms. The event logs will be presented also as EC predicates.

- IDS rules. As so far mentioned, we consider Snort rules. Snort rules are simple to write, yet powerful enough to detect a wide variety of hostile or merely suspicious network traffic. There are three base action directives that Snort can use when a packet matches a specified rule pattern: pass, log, or alert. Pass rules simply drop the packet. Log rules write the full packet to the logging routine that was user selected at runtime. Alert rules generate an event notification using the method specified by the user at the command line, and then log the full packet using the selected logging mechanism to enable later analysis. For instance, the following rule

  alert tcp any any → 192:168:1:0=24 any (flags : A; ack : 0;msg : "TCP ping detected")

  shows that an alert message will be generated when you receive a TCP packet with the A flag set and the acknowledgement contains a value of 0.The destination of this packet must be a host in network 192.168.1.0/24. This rule can be expressed in EC as follows:

  Happens(receive(TCP,A,ACK: 0; 192:168:1:0/24), t) →HoldsAt(e(Alert;,msg :"TCP ping detected"), t)

- Security requirements. They aim to protect the network from malicious attacks. Our approach checks that such requirements are preserved at run-time by monitoring the satisfaction of EC formulas that formalize them. This can be done by observing the network at run-time and checking observations. As an example of a monitoring property expressed in our EC-based language, consider the formula below:

  Happens(e(sender,receiver,req(op(ParamList))),t1)→Happens(e(receiver,sender,resp(op(ParamList))),t2)

  According to this formula, a network entity (receiver) which receives an event invoking the operation *op(ParamList)* in it (i.e. *e(sender; receiver; req(op(ParamList)))))* should complete the execution of *op(ParamList)* and respond to the caller (sender).

- Assumptions. They are also expressed in EC. Then, a collection of these EC formulas define the monitoring specification. A monitoring specification effectively describes the security policy of the network. When an attack takes place, we consider that the security policy is violated and, therefore, a monitoring rule is breached. Figure 2 shows our assumption editor that enables the network administrator to express the required rules to be added.

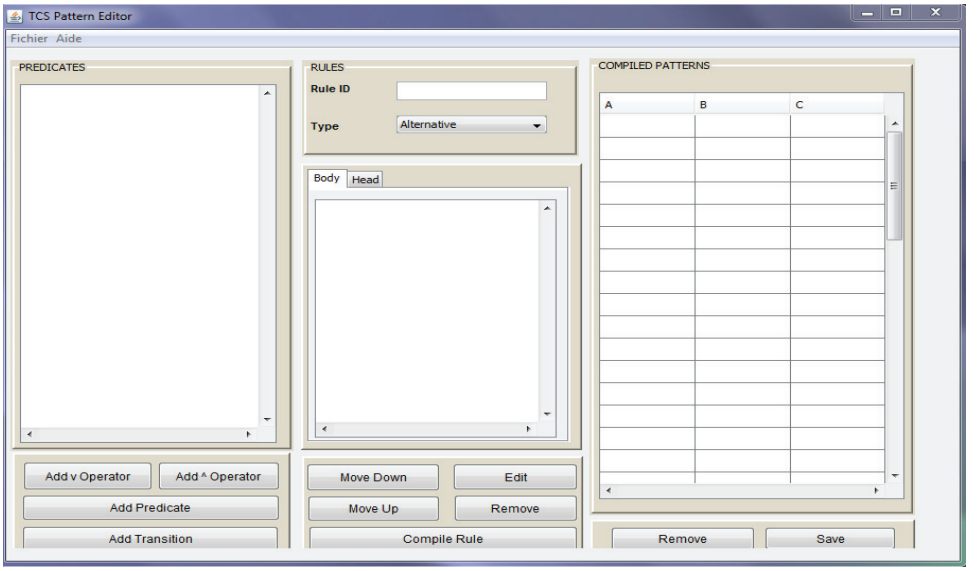

Fig. 2. Assumption editor

3.3. Encoding of EC in SPIKE

For the verification and the analysis of the authorization policies, we have employed the SPIKE [22] induction prover. The SPIKE induction prover has been designed to verify quantifier-free formulas in theories built with first order conditional rules. SPIKE proof method is based on the so-called cover set induction: Given a theory SPIKE computes in first step induction variables where to apply induction and induction terms which basically represent all possible values that can be taken by the induction variables. Given a conjecture (rule or a policy) to be checked, the prover selects induction variables according to the previous computation step, and substitutes them in all possible ways by induction terms. This operation generates several instances of the conjecture that are then simplified by rules, lemmas, and induction hypotheses. The ingredients of our encoding are:

- Data: All data information manipulated by the system is ranged over a set of sorts. This data concerns generally the argument types of events and fluent.
- Events: We consider that all events of the system are of sort Event, where the event symbols are the constructors of this sort.
- Fluent: The sort Fluent represents the set of fluent. All fluent symbols of the systems are the constructors of sort fluent that are also free.
- Axioms: We express all predicates used in EC as Boolean function symbols. For example the signature of the predicate Happens is expressed by *Happens : Event × Nat→Bool*. Then, the EC axioms necessary to do the verification process are also expressed in conditional equations.

In the same way, all detection framework components (security properties, assumptions, IDS rules, logs, and network specification) are formally expressed. Finally, we build an algebraic specification from the EC specification. Once building this specification, we can check all formulas by means the powerful deductive techniques (rewriting and induction) provided by SPIKE.

3.4 Verification process and deviations detection

All the generated axioms can be directly given to the SPIKE prover, which automatically orientes these axioms into conditional rewrite rules. Then, given as inputs the specification of the network expressed in algebraic equations and the security requirements to be checked, when SPIKE is called, either the rules proof succeed, or the SPIKE's proof-trace is used for extracting all scenarios which may lead to potential deviations.

Given the set of security properties R and assumptions A all expressed in EC, one can distinguish between two types of deviations [9]. These types of deviations depend on the recorded or the expected behavior of the network. The recorded behaviour of a network N at time t0, $RL(t0)$, is a set of events, and fluent initiation or termination literals of the forms: $Happens(e,t)$, $Initiates(e,f,t)$, $HoldsAt(f,t)$ which have been recorded during the operation of N and for which $0 \leq t$, and $t \leq t0$.
The expected behaviour of a network N given a subset S of its security properties and assumptions ($S \subseteq (R \cup A)$) at time t is a set of events EL(S,t) that is defined as:

$$EL(S,t) = \{e|\ (f \in S)\ and\ \{RL(t), f\}\ |=nf\ e\}$$

Where $|=nf$ signifies entailment using the normal rules of inference of first-order logic and the principle of negation as failure. The expected behaviour includes events which can be derived from the formulas in S but may have not been generated by the network operation.

Given the previous specifications, we can first detect inconsistencies between the recorded behaviour of a network N and an assumption about the behaviour of N, or any entity interacting with it. This type of deviations is defined as follows: An assumption statement of the form $f: C => A$ is inconsistent with the recorded behaviour of a network N until time t if and only if:

$$\{RL(t)\}\ |=nf\ \neg f$$

We can also detect inconsistencies between a security property or an assumption and the expected behaviour of N. This type of deviations is defined as follows: A security property or assumption of the form $f: C => A$ is inconsistent with the expected behaviour of a network N at time t if and only if:

$$\{RL(t), EL(dep(f), t), ECa\}\ |=nf\ \neg f$$

Where $dep(f)$ is the set of formulas $F : B => H \subseteq (R \cup A) - f$ which f depends on.

A formula $F : B => H \in dep(f)$, if its head H has a literal L that unifies with: (i) some literal K in the body C of f, or (ii) some literal K in the body $B1$ of another formula $F1$ that belongs to $dep(f)$, and ECa is the set of standard EC axioms . Thus, the check about the inconsistency of a formula with the expected behaviour must, in addition to events which are recorded in $RL(t)$, take into account events which should have been generated according to other formulas in $R \cup A$ and can affect the satisfiability of f.

4. Related Work

Several efforts are driven in the literature in the context of intrusion detection process. [13] presented an extensible IDS management architecture to manage security event and correlation. The validation and correctness of security policies are discussed in [14]. The authors propose a global architecture, based on an original meta-policy approach for access control and intrusion detection, to guarantee global security

properties. Their main idea is to apply verification techniques on the meta-policy while supporting local updates of the security policy so that the system can verify the respect of global security properties after any modification of the policy. In [15] the problem of the IDS rule conflict is involved. Since any conflict that arises between rules in the signature-based IDS detection could put the system in a vulnerable position, the authors address this conflict in host and network-based intrusion detection and response devices.

[16] proposed a specification-based approach for intrusion detection. The idea is to use traces, ordered sequences of execution events, to specify the intended behaviors of concurrent programs in distributed systems. The advantage of this approach is that in theory, it should be able to detect some new types of attacks that intruders will invent in the future. However, in this approach, substantial work is required to specify accurately the behavior of the many privileged system programs, and these specifications will be operating system specific. To address this issue, [17] proposed the use of inductive logic programming to synthesize specifications from valid traces. The automatically generated specifications may be combined with manual rules to reduce the work involved in the specification of valid program behaviors. The basic idea of [18] is to automatically generate the specification of a program by deriving an abstract model of the programs from the source or binary code. Attractive features of this approach are that it has the potential to detect unknown patterns of attacks and it has no false alerts, although it may miss some attacks. LAMBDA language [19] was used to describe attack scenarios as a combination of actions. Each action has conditions or requirements that must be satisfied for the action to succeed, and successful actions affect the network and may satisfy conditions for other actions. Actions can be combined using operators that specify sequencing, parallel unconstrained execution, absence of a condition, nondeterministic choice between multiple equivalent actions, and synchronized execution. [20] described the formalization of a correctness proof for a conflict detection algorithm for firewalls in the Coq Proof Assistant. It gives formal definitions in Coq of a firewall access rule and of an access request to a firewall. An approach to firewall policy specification and analysis that uses a formal framework for argumentation based preference reasoning was proposed in [21]. By allowing administrators to define network abstractions (e.g. subnets, protocols etc) security requirements can be specified in a declarative manner using high-level terms.

5. Conclusion

Intrusion Detection Systems is a powerful technology helping in protecting the precious data and networks from malicious and unauthorized access. As networks become faster there is an emerging need for security analysis techniques that can keep up with the increased network throughput. This paper has proposed an approach to improve the efficiency of the intrusion detection and monitoring processes using a event driven technique. The ingredients of this technique and their encodings using EC logic are also discussed.

Future work will include the refinement and the evaluation of the proposed approach. We are also working on data logs reduction and alerts classification and filtering in order to improve the verification process.

Acknowledgements

This paper is a partial result of a research project granted by King Abdul Aziz City for Sciences and Technology (KACST), Riyadh, Kingdom of Saudi Arabia, under grant number INF 36-8-08.

6. References

[1] H. Sallay M. Rouached ,A. Ammar ,O. ben fredj ,Khalid Al Shalfan ,Majdi Ben Saad :Wild-Inspired Intrusion Detection System Framework for High Speed Networks (φ|π) IDS Framework, IJISP,"Volume 5, Number 4 ,Oct ,2011 .pp.47-58

[2] Adel Ammar ,Hassen Sallay" : *Measuring connection features' relevance to attack detection using Neural Networks*". Journal of Computer Research, (JRC) published by the Federation of Arabic Scientific Councils ,N°1 ,Vol 10, 2011**.**

[3] Hassen Sallay., Khalid. Al-Shalfan and Ouissem. Benfredj, A scalable distributed IDS architecture for high speed networks. Int. J. Computer. Sciences and Network Security, 9: 9- 16.

[4] Hassen Sallay ,Towards an Integrated Intrusion Detection Monitoring in High Speed Networks ,Journal of Computer Science , Volume 7 ,Number ,7 July 2011 ,pp.2011 ,1104-1094 .

[5] J. Lin, X. Wang, S. Jajodia, Abstraction-based misuse detection: High-level specifications and adaptable strategies, in: Proceedings of the 11th IEEE workshop on Computer Security Foundations, IEEE Computer Society, Washington, DC, USA, 1998.

[6] J.-P. Pouzol, M. Ducass, Formal specification of intrusion signatures and detection rules, Computer Security Foundations Workshop, IEEE 0 (2002) 64. doi:http://doi.ieeecomputersociety.org/10.1109/CSFW.2002.1021807.

[7] M. Roger, M. Roger, J. Goubault-Larrecq, Log auditing through model-checking, in: In Proceedings from the 14th IEEE Computer Security Foundations Workshop (CSFW01, IEEE Computer Society Press, 2001, pp. 220–236.

[8] M. Shanahan, The Event Calculus Explained, 1999, pp. 409+.URLhttp://www.springerlink.com/content/1bxk8gd0n6pajxbq

[9] D. Lorenzoli, G. Spanoudakis, Detection of security and dependability threats: A belief based reasoning approach, in: SECURWARE, 2009, pp. 312–320.

[10] N. Am´alio, G. Spanoudakis, From monitoring templates to security monitoring and threat detection, in: SECURWARE, 2008, pp. 185–192.

[11] G. Spanoudakis, C. Kloukinas, K. Androutsopoulos, Towards security monitoring patterns, in: SAC, 2007, pp. 1518–1525.

[12] A. K. Bandara, E. C. Lupu, A. Russo, Using event calculus to formalise policy specification and analysis, in: 4th IEEEWorkshop on Policies for Networks and Distributed Systems (Policy 2003, 2003, pp. 26–39.

[13] S. Roschke, F. Cheng, C. Meinel, An extensible and virtualization-compatible ids management architecture, in: IAS, 2009, pp. 130–134.

[14] J. B. P. C. M. R. Blanc, M., C. Toinard, A collaborative approach for access control, intrusion detection and security testing, in: Proceedings of the International Symposium on Collaborative Technologies and Systems (CTS'06), 2006, pp. 270–277.

[15] N. Stakhanova, Y. Li, A. A. Ghorbani, Classification and discovery of rule misconfigurations in intrusion detection and response devices, in: Proceedings of the 2009World Congress on Privacy, Security, Trust and the Management of e-Business, CONGRESS '09, IEEE Computer Society, Washington, DC, USA, 2009, pp. 29–37.

[16] C. Ko, Logic induction of valid behavior specifications for intrusion detection, in: Proceedings of the 2000 IEEE Symposium on Security and Privacy, IEEE Computer Society, Washington, DC, USA, 2000, pp. 142–.

[17] D. Wagner, D. Dean, Intrusion detection via static analysis, in: Proceedings of the 2001 IEEE Symposium on Security and Privacy, IEEE Computer Society, Washington, DC, USA, 2001, pp. 156–.

[18] R. W. Ritchey, P. Ammann, Using model checking to analyze network vulnerabilities, in: IEEE Symposium on Security and Privacy, 2000, pp. 156–165.

[19] F. Cuppens, R. Ortalo, Lambda: A language to model a database for detection of attacks, in: Proceedings of the Third International Workshop on Recent Advances in Intrusion Detection, RAID '00, Springer-Verlag, London, UK, 2000, pp. 197–216.

[20] V. Capretta, B. Stepien, A. Felty, S. Matwin, Formal correctness of conflict detection for firewalls, in: Proceedings of the 2007 ACM workshop on Formal methods in security engineering, FMSE '07, ACM, New York, NY, USA, 2007, pp. 22–30.

[21] A. K. Bandara, A. C. Kakas, E. C. Lupu, A. Russo, Using argumentation logic for firewall configuration management, in: Integrated Network Management, 2009, pp. 180–187.

[22] S. Stratulat. A general framework to build contextual cover set induction provers. Journal of Symbolic Computation, 32(4):403–445, 2001.