Uniform approach to mandatory security of event management systems

Pavel Štros

stros@datasys.com

Department of system and network management
Datasys s.r.o.
Prague, Czech Republic

Abstract

The article reveals the specifics of a typical event management system (EMS) with respect to mandatory protection, defines necessary information granularity, provides semantics for associations between sensitivity levels and model constructs and analyzes effects of the classification of a particular construct on the classifications of other constructs.

A typical EMS uses rules to correlate and analyze events. It supplies an engine for event processing that will handle all the filtering and correlation of events. Data are typically stored in a database. Mandatory access policy model presented in this article not only states definitions and rules that control access to event data stored in the database, but supplies mandatory access policy to rules and supplies constraints to the execution of rules.

Keywords: event, class, access policy, security model, and mandatory.

1 Introduction

Event management is one of the central questions in network management. Event management pertains to detection, isolation, classification, filtration, correlation and presentation of events occurring in a network. All of the functional elements of EMS have to deal with large volumes of events [1]. Event management aids in the:

- Reduction in alarm events reported to a management station.
- Quick isolation and possible correction of fault.
- Detection of various composite events or event patterns that are a set of interrelated events.

In a multilevel security policy, every user is associated with a *clearance level* and every piece of information is associated with a *sensitivity level*. A subject is a user or a process running on behalf of a user. Process running on behalf of a user is associated with a security level dominated by the clearance level of this user. Classifications, clearances and security levels are both taken out of a partially ordered set of sensitivity levels. In the example presented in this paper I consider sensitivity levels Secret (S), Confidential (C) and Unclassified (U)

There are two key requirements of the Bell & LaPadula mandatory security policy model [2] that is used in [3] to analyze multilevel security policy in the context of object-oriented databases.

- ”No Read Up” - Subjects are only permitted to read data whose classification is dominated by their clearance.
- ”No Write Down” - Subjects are only permitted to write data whose classification dominates their clearance.

My research is derived from [3].

1.1 Research outline

The process of designing multilevel security policy in the context of EMS involves three issues. First, information granularity has to be defined. The question here is what constructs of the event are subjects to mandatory protection and have to be associated with sensitivity levels. Second, semantics for the association between a sensitivity level and a construct must be provided. In particular, does this association protect the

construct content or the existence of the construct? And third, effects of a classification of a construct on classifications of other constructs require analysis. The knowledge of a low classified data may disclose the knowledge of another higher classified data; in order to control these unauthorized inferences, assignment of sensitivity levels must be carefully analysed.

The objective of this paper is to define a set of principles addressing these three issues in the context of a generic EMS.

2 Multilevel security in the context of EMS

2.1 Terminology and assumptions

Event is a particular fault or incident within the computing environment that occurs on an object. An event usually represents either a change in status or a threshold violation. Every time the status of a managed object changes in any way, an event occurs. If this event is important enough to drive attention, or if it needs to be correlated with events from other sources and therefore cannot be fully processed at its local site, then the event should be forwarded to a central event server. The description of an event is referred to as an *event message*. Event messages are the central unit of information within the EMS. Event messages are structured chains and are typically provided in the form of *attributes*, which are ”name=value” pairs. The term ”event” or “*event instance*” is often used in place of the more appropriate term ”event message” within this article.

My research assumes *event class* hierarchy. Event classes determine the attributes and information that may constitute the event message. Each event is identified by a class name as a result of classification. Multiple inheritances are not allowed within class definitions.

A typical EMS uses *rules* to correlate and analyze events. An engine that runs a set of rules to determine if an action needs to be taken in case of an event performs this. The rule engine is responsible for:

- Finding applicable rules for a given event.
- Managing the execution of applicable rules.
- Storage of the processed event data in a database.

A *rule* describes what should be performed when the event server receives a particular event. Rules are used to assess the received event and to determine appropriate actions to perform and to proactively address situations before they become problems. A *rule definition* is a construct that lets you specify what action to take when a certain event is received. A *rule base* contains all rules and event class definitions that are to be performed against events. A rule is only triggered when the event under analysis has satisfied all of the conditions specified in the rule’s event filter. Rules are run one at a time and are usually applied based on their order within the rule base.

2.2 The information granularity

I have to identify what constructs of an event are subjects to mandatory protection and will be associated with sensitivity levels. There are two possible approaches to that: the Single-level Event and the Multi-level Event approaches. In the Single-level Event approach, every event class and event instance is assigned a sensitivity level. The sensitivity level applies to all the information encapsulated in the construct. In the Multi-level Event approach, every attribute of every event class and event instance is assigned with a sensitivity level. This sensitivity level protects the existence and the value of the attribute.

Security models based on the Single-level Event approach are easy to implement. However, the expressive power of the multilevel policy is poor. In my opinion, the Single-level Event approach is too restrictive. Thus, the Multi-level Event approach is adopted in my research. As a result, the following construct may by subject to mandatory protection in my model: Event class, Event class attribute, Inheritance link, Event instance, Event instance attribute, Event instance attribute value, Event instance link and Rule definition. However, I will unveil in the next section that not all of these constructs need to be assigned a sensitivity level.

2.3 Semantics of sensitivity level association and inference effects

In this section the principles of semantics are stated as rules that should be applied when designing multilevel EMS. The rules are stated using easy-to-read sentences, without using any formal language. They are derived from the research related to mandatory security in object-oriented databases [3]. A more formal proof presented in [4] provides evidence that these rules are sound and complete. The purpose of this section is to derive the semantics of every association between a sensitivity level and each of the constructs identified in previous section. I also state the inference control rules that must be enforced when assigning security levels to these constructs.

2.3.1 Event classes

Event classes are used for the operational classification of events and determine the attributes and information that may constitute the event message. In other words, they are used for event message formatting. The correctness and availability of class definitions determine successful processing of a newly arrived event. In the proposed system, classes are used as the means of assigning sensitivity level to a newly arrived event. The following rules are relevant to mandatory protection of event classes; detailed reasoning can be found in [3]:

1. Each event class is associated with a sensitivity level. Assigning a sensitivity level to event class *c_ec* protects the existence of class *c_ec* i.e. the fact that *c* is an event class.
2. Each event class attribute is associated with a sensitivity level. Assigning a sensitivity level to an attribute *c_eca* of a class *c_ec* protects its existence, i.e. the fact that *c_eca* is an attribute of *c_ec*.
3. The sensitivity level assigned to an attribute *c_eca* of a class *c_ec* must dominate the security level assigned to *c_ec*.

Event class hierarchy is used for event type definitions. Only single inheritance is allowed. With the exception of the ”root event class” each event class is assigned just one ”parent class”. ”Child event class” contains all the attributes of its parent class but may have additional attributes defined. The definition of the ”root event class” is classified at the lowest sensitivity level and contains all ”basic description of an event” attributes, such as ”date_occured”, ”date_received”, ”received_from”, etc. The following rules are relevant to mandatory protection of inheritance links between event classes; detailed reasoning can be found in [3]:

4. Each inheritance link is associated with a sensitivity level. Assigning a sensitivity level to an inheritance link between a class *c_ec* and a class *c_ec'* protects the fact that *c_ec'* inherits from *c_ec*.
5. The sensitivity level assigned to an inheritance link between two classes *c_ec* and *c_ec'* must dominate the least upper bound of the sensitivity level assigned to *c_ec* and the sensitivity level assigned to *c_ec*'.
6. The least upper bound of the sensitivity level assigned to an attribute *c_eca* of a class *c_ec* and the sensitivity level assigned to an inheritance link between a class *c_ec'* and *c_ec* must dominate the sensitivity level assigned to *c_eca* in class *c_ec*'.

The following constraints are imposed by the assumption that each event class with the exception of the ”root event class” is assigned just one parent class and that a child event class contains all the attributes of its parent class. In particular the constraints have to be enforced when deleting or modifying parent classes.

7. If *c_ec'* is a event class classified at level *l*, there must be at least one parent event class *c_ec* such that the inheritance link between *c_ec* and *c_ec'* is classified at level *l*.[1]
8. If an inherited attribute *c_eca* of an event class *c_ec'* is classified at level *l*, there must be at least one parent event class *c_ec* such that *c_eca* is an attribute of *c_ec* and *l* is equal[2] to the least upper bound of the sensitivity level assigned to *c_eca* in class *c_ec* and the sensitivity level assigned to the inheritance link between *c_ec* and *c_ec'*.

2.3.2 Event messages

In the proposed system, classes are used as the means of assigning sensitivity level to a newly arrived event. Because the event instance is assigned the same sensitivity level as the corresponding event class, there is no

[1] Notice that it is not explicitly stated, that a parent event class classified at least at level *l* must exist.
[2] Due to paragraph 6, it cannot be greater.

need to assign sensitivity level to the construct ”Event instance link”. Event instance attributes do not suffer from the integrity constraint that is present in the object paradigm. Unlike object attributes, event instance attributes are not required to have a value. Therefore, users cannot guess the existence of a high-classified value if the attribute does not have value. Therefore, in the context of EMS, there is no need to introduce polyinstantiation to attributes. The following rules are relevant to mandatory protection of event instances:

9. Each event instance is assigned with a sensitivity level of the corresponding event class. Assigning a sensitivity level to an event instance *c_ei* protects the fact that *c_ei* exists.
10. Each event instance attribute is associated with a sensitivity level of the corresponding event class attribute. Assigning a sensitivity level to an attribute *c_eia* of an event instance *c_ei* protects the fact that *c_eia* is an attribute of *c_ei*.
11. Each event instance attribute value is associated with a sensitivity level of the corresponding event instance attribute. Assigning a sensitivity level to a event instance attribute value *c_eiav* of an event instance attribute *c_eia* of an event instance *c_ei* protects the fact that the value of *c_eia* in object *c_ei* is equal to *c_eiav*.

2.3.2.1 Remarks to the event data storage

Most existing EMS’s use traditional (relational) database systems that are not tightly integrated with the EMS and are only used as storage for event data. Using an Active Database Management System (ADBMS) to build an EMS system would allow one to specify re-actions to both simple and composite events in the form of declarative ECA (Event-Condition-Action) rules. An advantage of this proposed architecture is that advanced research has been done in the security of ADBMS’ and subset of the ADBMS security research results can be directly applied to the proposed mandatory security model for event management systems.

2.3.3 Rules

A rule is a construct that lets you specify what action to take when a certain event is received. Rules are usually written in a high-level structured language called the rule language that supports the well-known ”event-condition-action” schema. There seems to be no need to apply mandatory protection to parts of the rule structure. Protection on the rule level is only considered in this article. The following rule is relevant to mandatory protection of rule definitions:

12. Each rule definition is assigned with a sensitivity level equal to the security level of the creators’ session. Assigning a sensitivity level to a rule definition *c_rd* protects the fact that *c_rd* exists.

3 Event processing

3.1 Classification of events

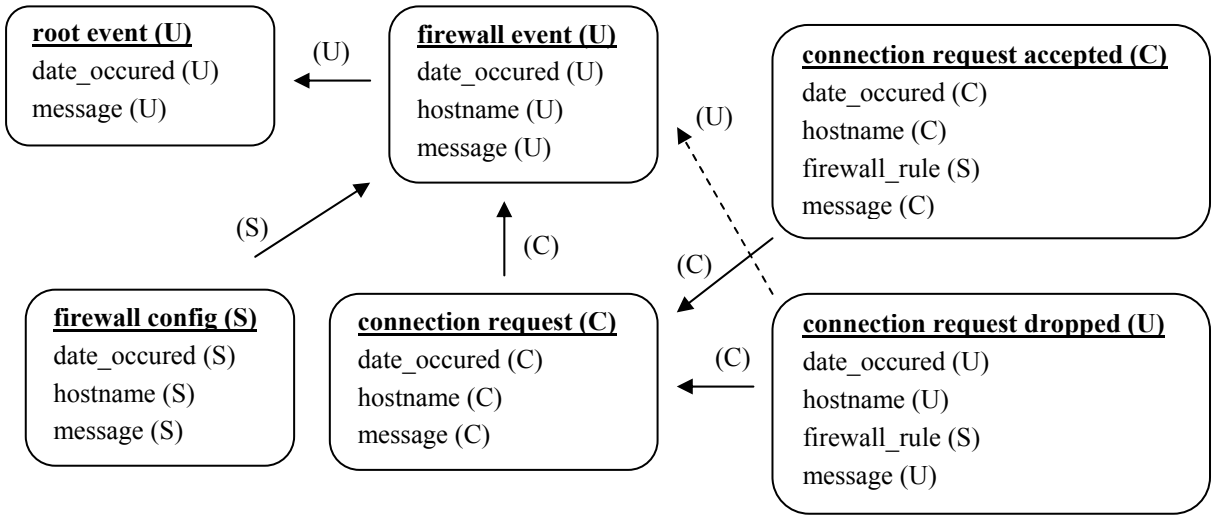
Events received by the EMS are first evaluated against all available event classes by the rule engine. Event classes with higher sensitivity levels are considered first. The most specific matching class is found and event message is assigned with a sensitivity level of the class. The event message is then formatted according to attributes defined for that event class. Each event message attribute and event message attribute value is associated with a sensitivity level of the corresponding event class attribute.

3.2 Evaluation against the rule base

In the proposed system, rules are evaluated based on their order within the rule base, but rules with higher sensitivity levels have higher priority and are considered first. A rule is only triggered when its sensitivity level dominates the sensitivity level of the event and the event under analysis has satisfied all of the conditions specified in the rule. Only event attributes ”visible” at the rule’s sensitivity level may be evaluated in conditions. Rules are run one at a time.

3.2.1 Event processing example

Consider event class hierarchy that is depicted in the following diagram.



The event class hierarchy may be built from the definition of classes that is listed below. The syntax used for the class definition language is not really important and I will not explain it in detail. However, you can dip that the definition of a class contains a name and sensitivity classification statement, and a link to a parent class in the header part. Further, there are message format constraints and attribute assignment formulas in the body part of the class definition The event class hierarchy created in accordance with this example permits access to information about dropped connection requests to unclassified (U) users, whereas permits access to information about accepted connection requests to confidential (C) and secret (S) users only. Clearance level secret is required to get knowledge about the ”firewall_rule” attribute and to gain access to the firewall rule number (the value of the attribute).

CLASS (”root event”; U) {

(%d %s; date_occured; message);*

date_occured = (date; U);

message = (string; U);

}

CLASS (”firewall event”; U) ISA ”root event”{

(%d firewall at host %s %s; date_occured; hostname; message)*

hostname = (string; U);

}

CLASS (”firewall config”; S) ISA ”firewall event” {

(%d firewall at host %s configuration changed: %s; date_occured; hostname; message);*

}

CLASS (”connection request”; C) ISA ”firewall event” {

(%d firewall at host %s connection request: %s; date_occured; hostname; message);*

}

CLASS (”connection request accepted”; C) ISA ”connection request” {

%d firewall at host %s connection request accepted: %s, rule: %n; date_occured; hostname; message; firewall_rule);*

firewall_rule = (integer; S);

}

CLASS (”connection request dropped”; U) ISA ”connection request” {

%d firewall at host %s connection request dropped: %s, rule: %n; date_occured; hostname; message; firewall_rule);*

date_occured : U;

hostname : U;

message : U;

firewall_rule = (integer; S);

}

The class hierarchy is built in accordance to rules 1-8. In particular, rules 7-8 are satisfied for the class ”connection request dropped” due to existence of the parent class ”firewall event”. Rules 9-12 do not apply to event classes.

When the event message

”2003/02/28 16:03:11 firewall at host guardian connection request: src 192.168.1.15 dst 192.168.2.7 svc ssh”

is received by the rule engine, event classes at secret sensitivity level are evaluated first but no matching class is found. Thus event classes at confidential sensitivity level are evaluated. The matching class ”connection request” is the most specific matching class at the evaluated sensitivity level. The event message is formatted according to the ”connection request” class definition.

EVENT (”connection request”; C) {

date_occured = (” 2003/02/28 16:03:11”; C);

hostname = (”guardian”; C);

message = (”src 192.168.1.15 dst 192.168.2.7 svc ssh”; C);

}

Should there be a rule within the rule base at sensibility level secret or confidential, whose condition part is satisfied by the attribute values of this event, the rule is triggered.

4 Conclusions

In the context of EMS the following constructs should be subject to mandatory protection: Event class, Event class attribute, Inheritance link, Event instance, Event instance attribute, Event instance attribute value, Rule definition. With respect to semantics presented in this article and the subsequent inference analysis there is no need to introduce polyinstantiation.

There are some key processing principles that must be introduced to the rule engine in order to ensure that the mandatory protection can’t be mitigated. Rules are processed not only with respect to their position within the rule base, but higher sensitivity level rules are evaluated first.

Constraints applicable to deletion and modification of parent event classes should be subject to further research and the corresponding conflict between unauthorised inference and usability of such event management system should be solved.

References

[1] Hasan, M. Z.: The management of data, events, and information presentation for network management, *Thesis of the University of Waterloo*, Ontario, Canada, 1996.

[2] Bell, D., and LaPadula, L. Secure Computer Systems: Unified Exposition and Multics Interpretation, *Technical Report ESD-TR-75-306, MTR 2997, MITRE*, Bedford, Mass. 1975.

[3] Cuppens, F., and Gabillon, A.: Rules for Designing Multilevel Object-Oriented Databases, *Fifth European Symposium on Research In Computer Security (ESORICS)*, Louvain la Neuve, Belgium, Springer-Verlag, September 1998.

[4] Gabillon, A.: Sécurite multi-niveaux dans les bases de données à objets, *Ph.D. dissertation*, ENSAE, 1995.