A Generic Architecture for Fusion-Based Intrusion Detection Systems

Remco C. de Boer

Erasmus University Rotterdam
Rotterdam School of Economics
Master Thesis
Business Informatics (*Bestuurlijke Informatica)*

Supervised by dr. ir. Jan van den Berg

Supported by Ubizen

A Generic Architecture for Fusion-Based Intrusion Detection Systems

Remco C. de Boer
<rcdeboer@xs4all.nl>

October 2002

Contents

List of Figures

Acknowledgments

This thesis forms the conclusion of my study *Bestuurlijke Informatica* at the Erasmus University Rotterdam. It was written over the course of almost a year, with great support of many people. I would like to thank the following people for their efforts, input, and support:

- My supervisor, dr. ir. Jan van den Berg, for his attention on the overall direction as well as the details.
- All the people from Ubizen NL - in particular my coaches Wilco van Ginkel (senior security consultant) and Diederik Klijn (security engineer), for the fun and instructive discussions and their invaluable input and feedback.
- Johan van Oeyen and Yves van de Weyer from Ubizen HQ in Belgium, for providing me with first-hand information about intrusion detection and Intrusion Detection Systems.
- My parents, for always having encouraged me not only to study, but to first and foremost choose a study that's fun.
- My girlfriend, Marjoleine, for showing so much interest in what I was doing she even managed to explain the subject of my thesis to her grandparents.

Thank you.

Remco de Boer
Mijdrecht, 15 October 2002

Chapter 1

Introduction

Since their invention, computers have been playing an ever increasing role in our society. Nowadays, a lot of computers are interconnected by means of networks, the largest of all being without doubt the global Internet. The interconnection of computer systems has numerous advantages. Computer users can communicate with each other simply, for instance via electronic mail, bulletin boards, or so-called 'chat' applications. Businesses can work together more easily, for instance via EDI. And information can be readily accessible from numerous locations, to name but a few.

However, connecting computer systems makes them an easier target for attacks[1]; physical access is no longer needed to manipulate a computer system. With information being a valuable corporate asset, more and more effort is being made to ensure the security of computer systems. Protective security functions aim to prevent an incident from happening. A firewall, for instance, limits the accessibility of a network or host. Only traffic that complies with a certain policy, a number of predefined rules, is allowed to pass.

In practice, however, it is nearly impossible to prevent all incidents from happening. Regardless of the protective security measures that are applied, incidents are likely to happen. The purpose of reactive security functions is to minimize the effects of an incident while it is happening. After an incident has taken place, corrective security functions are used to handle the results of the incident. For instance, by restoring a backup of a compromised system and prosecuting the attacker.

A special and important reactive security function is detection. Without detecting an incident, you never know that you are or were under attack. The result is that any further reactive and corrective security functions are useless, for they rely on prior detection. More and more organizations implement an Intrusion Detection System (IDS) to automate the detection task. An IDS observes events on a network or host and whenever it suspects an attack, it raises an alarm.

[1] An attack is an attempt to compromise the confidentiality, integrity, availability and/or accountability of (the use of) information residing on a computer.

The ideal situation arises when an IDS detects *all* intrusions, and reports *only* real intrusions, i.e. raises no false alarms. In this ideal situation, the output of an IDS is readily usable, without the need of any significant further analysis by the user. This output then gives an instant overview of which attacks occur where in the network, and how severe these attacks are. However, this ideal situation is not yet occurring - and perhaps never will exist. This means that, with respect to the ideal situation, current IDSs suffer from a number of problems. One of the biggest problems is the high number of false alarms. Another big problem is the high number of - legitimate - alarms that tends to result from an attack. A lot of research is needed and carried out to evolve IDSs towards the ideal situation. Part of this research is in the direction of resolving the above mentioned problems with the high number of alarms in general, and false alarms in particular. A research area that seems particularly promising in this direction is the application of multisensor data fusion techniques to intrusion detection. Multisensor data fusion is a framework that enables the combination of data from multiple, possibly heterogeneous, sensors. When applied to intrusion detection, data originating from different types of IDSs are combined. Based on this combined data, hopefully better decisions can be made regarding the (non)existence of an attack, and the current situation as a whole, resulting in less false alarms.

Bass [Bas00] describes a process model for fusion-based Intrusion Detection Systems. It also includes a brief overview of the functionality of a fusion-based IDS. A number of research projects have started to implement IDSs using data fusion techniques. However, a lot of these research projects seem to use a rather ad-hoc approach, and focus on the implementation. Besides the process model and overview of functionality by Bass, there is no clear picture available of what a good, generic architecture of a fusion-based IDS should look like.

Based on these observations, it seems important to start a systematic analysis and to develop a generic architecture for fusion-based IDSs. Such an architecture is an elaboration on the process model and functionality overview from [Bas00], and provides another framework to think and talk about fusion-based Intrusion Detection Systems. While there are numerous advantages of having an architectural description of a system - it enables easy adoption, reusability, and provides an important design step towards implementation - this thesis concentrates on a more specific goal. The goal of this thesis is to solve, at least conceptually, the current problems with the high number of false and legitimate alarms, by deriving a generic architecture of a fusion-based Intrusion Detection System. As indicated earlier, such an architecture can also aid in further development and implementation of a fusion-based IDS.

The actual implementation of IDS fusion algorithms and eventually a complete fusion-based IDS is not covered by this thesis, and is the logical next step - although research is already performed in this area. Instead of focussing on a specific implementation, this thesis focusses on a - more abstract - *generic* architecture for fusion-based intrusion detection. Hence,

the character of this thesis remains analytical. The lack of a concrete implementation of the derived generic architecture in this thesis, is mainly due to time constraints. To enable at least some preliminary evaluation of the architecture, a test case is presented. The architecture is then applied to this test case. This shows on a conceptual level how a fusion-based IDS handles such a case, as well as the result thereof.

This thesis relies heavily on data fusion theory - to which the reader is introduced in Chapter 3 - and uses [Bas00] as a starting point. An elaboration on the functionality overview in [Bas00] provides a detailed functional-level analysis of a fusion-based IDS. The fundamental issues in building a data fusion system, as outlined by Hall and Llinas [HL97][2], are used as a guideline throughout this thesis. Finally, with the help of fusion architecture descriptions from fusion theory, and considering the results of the performed analysis, an architecture for fusion-based IDSs is derived.

The remainder of this thesis is divided into four chapters. Chapter 2 provides an overview of the workings of, and problems with, current IDSs, and lists the state of the art in IDS interoperability and data fusion techniques applied to IDS. In Chapter 3 the reader is introduced to multisensor data fusion theory. In Chapter 4, the above methodology is used to present an architecture for fusion-based Intrusion Detection Systems. That chapter also presents a test case to evaluate the developed architecture. The last chapter, Chapter 5, states the conclusions of this thesis and points to possible further research in this area.

[2]These fundamental issues can be found in Section 3.5.

Chapter 2

Intrusion Detection Systems

This chapter describes the need for Information Security and the roles that Intrusion Detection and Intrusion Detection Systems have. It turns out that current Intrusion Detection Systems suffer from a number of problems. Solutions to these problems are active research topics for numerous researchers. A particular research area is the cooperation between individual Intrusion Detection Systems. An overview of research performed in this area is given and evaluated. This evaluation leads to the statement of the goal of this thesis.

2.1 Information Security

Like other business assets, information has value to an organization. An organization's information should therefore be appropriately protected. The British Standard Institution defines Information Security as the preservation of three features of information [BSI00]:

1. *Confidentiality* - ensures "that information is accessible only to those authorized to have access";
2. *Integrity* - safeguards "the accuracy and completeness of information and processing methods";
3. *Availability* - ensures "that authorized users have access to information and associated assets when required";

Ubizen adds another feature to this list, concerning the usage of information [Ubi00]:

4. *Accountability* - ensures that "an action (can) be linked without doubt to its initiator".

Preservation of these four features of information is the goal of information security. In other words, confidentiality, integrity, availability and accountability are the four *security requirements*. To satisfy the security requirements, *security services* can be used. Stallings [Sta99] identifies six security services:

1. *Confidentiality*: the protection of transmitted data from passive attacks (eavesdropping);
2. *Authentication*: assures that a communication is authentic;
3. *Integrity*: assures that messages are received as sent, with no duplication, insertion, modification, reordening or replays;
4. *Non-repudiation*: prevents sender and receiver from denying a transmitted message;
5. *Access Control*: limits access to systems and applications;
6. *Availability*: prevents or recovers from loss of availability.

Security services are provided by implementations of *controls* [BSI00] or *security mechanisms* [Sta99]. Security services, and hence the controls that provide them, can be classified by means of *security functions* [Ubi00]. A security function is a group of security services related to a main domain of action. Multiple security functions can again be grouped together, distinguishing between *protective*, *reactive* and *corrective* functions. This distinction is based on the time, relative to an incident, that the corresponding security functions apply. An overview of security functions, taken from [Ubi00], is given below:

1. *Protective* security functions apply before an incident has happened:
   (a) *prevention* aims to
      - prepare the means and resources to fight the future possible aggression;
      - prepare the organization to face and enforce the security strategy.
   (b) *deterrence* aims to
      - rise the risk for the aggressor to be detected, identified and prosecuted;
      - lure the aggressor placing 'decoys' or traps he'll fall in;
      - prepare tools to trace the aggressor and possibly attack him back.
   (c) *protection* aims to
      - lower the probability of, and the vulnerability of the system to accidental incidents;

- restrain the access possibilities to the system, its functions and its data;
- increase the required resources and level of experience to break into the system.

2. *Reactive* security functions apply during an incident:
   (a) *detection* aims to
   - immediately allow the most adequate reaction to an incident by detecting it and alerting the appropriate staff.

   (b) *containment* aims to
   - avoid the extension or propagation of the incident;
   - reduce the direct and immediate consequences of the incident.

   (c) *intervention* aims to
   - stop the incident;
   - reduce the direct and immediate consequences of the incident.

3. *Corrective* security functions apply after an incident took place:
   (a) *recovery* aims to
   - reactivate and operate the system even in degraded conditions;
   - reduce the indirect consequences of the incident.

   (b) *restoration* aims to
   - reactivate and operate the system in its normal configuration;
   - recuperate lost data and transactions by using backups.

   (c) *compensation* aims to
   - reduce the long term consequences of the incident.

2.2 Intrusion Detection and its Problems

An intrusion is defined as any set of actions that attempt to compromise the integrity, confidentiality or availability of a resource [Ype01], or accountability of the use of a resource [Ubi00], relative to a security policy. Intrusion Detection (ID) is the process of monitoring computer networks and systems for violations of the security policy [Ype01].

Ypey lists the benefits of Intrusion Detection [Ype01]. In parentheses are the security functions that the benefit subserves:

- deterrence (deterrence);

- detection (detection);
- damage assessment (recovery and restoration);
- attack anticipation (prevention);
- prosecution support (compensation).

In practice, the emphasis of Intrusion Detection lies on the detection function.

The definition of Intrusion Detection does not indicate how the detection should take place. Intrusion Detection can be accomplished manually, for instance by examining log files for signs of intrusions. Intrusion Detection can also be automated. A system that performs automated Intrusion Detection is called an Intrusion Detection System (IDS) [Col] .

Intrusion Detection Systems logically consist of three functional components [A+00]:

- *sensors*: responsible for collecting data;
- *analyzers*: responsible for analyzing data and determining if an intrusion has occurred;
- *user interface*: enabling a user to view the output or control the behavior of the system.

In current IDSs, the analyzer makes a decision - based on the data collected by the sensors - that puts the IDS in one of two states. At any moment the state of an IDS is either *positive*, indicating an intrusion, or *negative*, not indicating an intrusion. The analyzer's conclusion, and hence the state the IDS is in, may be either *true*, meaning the state of the IDS is appropriate, or *false*, meaning the state of the IDS is inappropriate. This means that there are four possible classifications of the state of an IDS:

- *true positive* means the IDS appropriately indicates an intrusion;
- *true negative* means the IDS appropriately doesn't indicate an intrusion;
- *false positive* means the IDS inappropriately indicates an intrusion;
- *false negative* means the IDS inappropriately doesn't indicate an intrusion.

The IDS should maximize the true states, and at the same time minimize the false ones [Ype01].

IDSs can be classified based on the location of the sensor [Ype01, Sch00, MCA00]. An IDS that has a sensor that monitors a host is called a Host-based IDS (HIDS). A HIDS's sensor collects data from sources internal to an individual system, such as operating system audit trails and system logs. When the IDS's sensor monitors a network instead, collecting network packets, the IDS is called a Network-based IDS (NIDS). A NIDS can typically monitor a complete (segment of a) network. A HIDS is per definition confined to a single host.

High-speed networks, switched networks and encrypted traffic can severely limit the view of a NIDS. Since a NIDS looks at all traffic on a network segment, a high-speed network can simply overwhelm it and force it to drop packets. A switched network, unlike a traditional shared network, does not send all data to all hosts. This enhances performance, but also prevents a NIDS from looking at traffic not directed to itself. Some switches provide a so-called spanning port to solve this problem. When enabled, the switch copies all data that passes through it to this spanning port. Encryption in itself does not limit the visibility of the traffic. It is however impossible to analyze it until it has been decrypted on the target host, often within a specific application.

While a HIDS does not necessarily suffer from these problems, Host-based Intrusion Detection has some other disadvantages. Since a HIDS has a limited view of only one host, one must be installed on every system that should be monitored. The presence of a HIDS on a system can affect the system's performance. Furthermore, when a system is successfully compromised the attacker might be able to shut off the HIDS.

Another classification is based on the detection mechanism of the analyzer [Ype01, Sch00, A$^+$00, MCA00]. A distinction is made between *misuse detection* and *anomaly detection*. Misuse detection is the easiest mechanism. It uses pattern-matching techniques to determine if the observed data corresponds to a known attack. This is much like the technique that most anti virus software uses. The number of false positives is relatively low when compared with anomaly detection, but since signatures can match both data belonging to intrusive and data belonging to non-intrusive behavior false positives are still quite common. The main disadvantage of misuse detection is that it can at best detect intrusions that are at least in some way equivalent to a previously known pattern. It can not detect a truly novel attack. This means that the list of signatures, like that of anti virus software, must be constantly updated and the IDS is in general always one step behind the attackers.

Anomaly detection views Intrusion Detection as a pattern recognition problem, rather than a signature matching problem. It assumes that all intrusive actions are necessarily anomalous, and uses a definition of normal activity. The concept of normal activity can for instance be modelled using statistical techniques. Anything that deviates from normal activity is deemed a possible intrusion. An example of such a deviation is a user who normally only logs in during business hours and suddenly starts to log in

at night from a remote location. This might indicate that the user's account has been compromised, although it could just as well be the user himself making legitimate use of his account, for instance because of a project nearing a deadline. With anomaly detection, the IDS can detect new attacks, for in general they will deviate from normal behavior. The amount of false positives, however, tends to be very high with this mechanism. There are some other problems related to anomaly detection. There is for instance the possibility that an intrusion takes place while the IDS is modelling normal behavior. This results in the intrusion being considered normal. Furthermore, an intrusion might look similar to normal behavior and thus be missed. Yet the main problem with anomaly detection remains the high rate of false positives.

2.3 Cooperating IDSs

Section 2.2 outlines some of the intrinsic problems with the different types of current IDSs. These problems include the inherently limited view of a HIDS, the inability of a NIDS to see all traffic, the false negatives that result from misuse detection, and the high rate of false positives - which tends to be even higher with anomaly detection. To be able to solve at least some of these problems, IDSs should be able to cooperate. When there is cooperation between different types of IDSs, the advantages of each type can be combined and the disadvantages minimized [M$^+$01]. Current IDSs are often proprietary, closed systems that are not able to cooperate very well[1].

2.3.1 Interoperability

The lack of interoperability has been recognized, and in 1997 the US government's Defense Advanced Research Projects Agency (DARPA) initiated the Common Intrusion Detection Framework (CIDF) research project [Kot02, A$^+$00]. CIDF contains the Common Intrusion Specification Language (CISL), which is a Lisp-like language that is used to represent intrusion data and communicate this data between IDS components. Although CISL is quite powerful, and vendors were invited to participate in the CIDF project, IDS vendors never showed much interest in it. Development of the project ceased in 1999.

The CIDF project triggered the creation of a new working group within the Internet Engineering Task Force (IETF) [Kot02, IDW, A$^+$00]. The Intrusion Detection Working Group (IDWG) has three output goals:

[1] Of course, there are exceptions to this rule: Snort [Sno], for instance, is a well-known open source Intrusion Detection System.

1. A requirements document, which describes the high-level functional requirements for communication between IDSs and requirements for communication between IDSs and management systems, including the rationale for those requirements;
2. A common intrusion language specification, which describes data formats that satisfy the requirements;
3. A framework document, which identifies existing protocols best used for communication between IDSs, and describes how the devised data formats relate to them.

Unlike CIDF, which was a research project, the IDWG tries to establish a standard for IDS interoperability. At the time of this writing, the IDWG has submitted the requirements, language and transport documents to the Internet Engineering Steering Group (IESG) for consideration as Request for Comments (RFCs). One of the IDWG's proposals is the Intrusion Detection Message Exchange Format (IDMEF). This is a data format that IDSs can use to report alerts. It includes an object-oriented data model and XML based implementation. IDMEF is independent of the communication protocol that is used.

Another IDWG proposal is the Intrusion Detection Exchange Protocol, or IDXP. IDXP is an application-level protocol that provides for the exchange of IDMEF messages, unstructured text and binary data. It is in part specified as a Blocks Extensible Exchange Protocol (BEEP) profile. BEEP is a generic application protocol framework. It does not rely on any particular transport protocol and maps easily to TCP/IP. Through the use of other profiles, BEEP allows features such as authentication, confidentiality and 'tunneling' through firewalls.

2.3.2 Intelligent Attack Analysis and Data Fusion

Allen et al. [A$^+$00] concludes that one of the most serious problems with current IDSs is the high false alarm rate, in other words the high number of false positives. Every declared intrusion requires time to investigate, so a large number of false positives costs a lot of time to investigate. Also, a large number of false positives can distract the attention from a true positive. This means that a true positive might accidentally be ignored. Furthermore, the number of false positives might become so high that system administrators simply ignore all warnings. The conclusion that this is one of the most serious problems is endorsed by the observation of Axelsson that the problem of base-rate fallacy applies to Intrusion Detection [Axe99]. This means that even a small false positive rate has a significant negative impact on the probability that an alarm indicates a real intrusion. See also Appendix A.

An August 2001 survey by Information Security Magazine [ISM01] restates

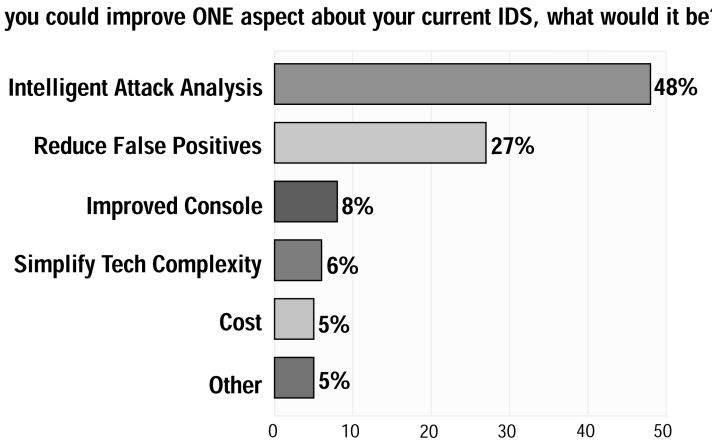


Figure 2.1: Question from an Information Security Magazine survey (from [ISM01]).

the conclusion of Allen et al. When asked to name one aspect of IDSs to be improved, 27 percent of the 300 interviewed information security professionals named the reduction of false positives. Another 48 percent named a more intelligent attack analysis (see also Figure 2.1). This also implies a need for reduced false positives, since a more intelligent attack analysis provides not only a means to detect sophisticated attacks, but also to reduce the false positive rate [A+00]. This means that 75 percent of the interviewed information security professionals either implicitly or explicitly named the reduction of the high rate of false positives to be at least part of their primary wish of improvement of IDSs.

Several advanced research areas have been identified, each of which improves current IDSs by enabling a more intelligent analysis of attacks and attack data [A+00]. Examples are the application of machine learning algorithms - for instance, example classification, neural nets, and genetic algorithms - and the use of state transition diagrams and petri net modelling. Another example is the application of multisensor data fusion techniques. The main idea behind multisensor data fusion is that the combination of data from multiple sensors enhances the quality of the resulting information. A precise definition is given in Chapter 3. That chapter also contains an introduction to multisensor data fusion theory. When applied to intrusion detection, data fusion enables the combination of, and intelligent reasoning with, the output of different types of IDSs. By making inferences from the combined data, a multiple level-of-abstraction situation description emerges. Combination of IDS data is part of the solution to the problem with high false positive rates [A+00, Bas00].

Bass argues that multisensor data fusion provides an important functional framework for building next generation IDSs. Bass also presents an Intrusion Detection Data Fusion model, based on the Joint Directors of Labo-

ratories (JDL) Functional Data Fusion Process Model. The JDL model is described in more detail in Section 3.2. The Intrusion Detection Data Fusion model is shown in Figure 2.2. The four levels correspond to the levels defined in the JDL model. Level 1 fusion results in a collection of objects representing the observed data, the object base. This object base is further analyzed by the Level 2 and Level 3 processes, to form the situation base. Such a next generation, fusion-based IDS would have an inference hierarchy as depicted in Figure 2.3. At the lowest level of inference, a fusion-based IDS indicates the existence of an intrusion. At the highest level of inference, such an IDS presents an analysis of the threat of the current situation [Bas00].

There are a number of (research) projects that have started to implement multisensor data fusion techniques. One of these projects is EMERALD, an acronym for 'Event Monitoring Enabling Responses to Anomalous Live Disturbances' [VS01]. It couples sensors, so the state of one sensor can adjust another. This suppresses false positives and increases sensitivity. EMERALD uses an extension of the IETF/IDWG IDMEF to introduce amongst others the concept of an alert thread. An EMERALD alert thread consists of alerts that originate from the same sensor and belong to the same attack. A second concept that is used is that of a meta alert. A meta alert is composed of one or more alerts that may originate from multiple heterogeneous sensors. A probabilistic alert fusion algorithm looks at the similarity of two alerts, usually a new alert and a meta alert. The overall similarity of two alerts is based on the similarity and expected similarity of their features. Based on the overall similarity the system can decide to fuse the two alerts. This fusion usually involves the merging of the features of the two alerts.

Another project that uses multisensor data fusion techniques is the french MIRADOR (Mécanismes de détection d'Intrusion et de Réaction aux Attaques en DOmaine militaiRe) project [Cup01]. Cooperation between multiple IDSs is achieved by means of a cooperation module. This module has five main functions:

1. *alert base management* receives alerts from multiple IDSs and stores them into a relational database;

2. *alert clustering* accesses the database and generates alert clusters that correspond to the same occurrence of an attack;

3. *alert merging* creates for each cluster a new, global alert that represent the information in that cluster;

4. *alert correlation* correlates global alerts in order to create candidate plans that correspond to the intrusion plan that is executed by an intruder;

5. *intention recognition* extrapolates the candidate plans to anticipate the intruder intentions.

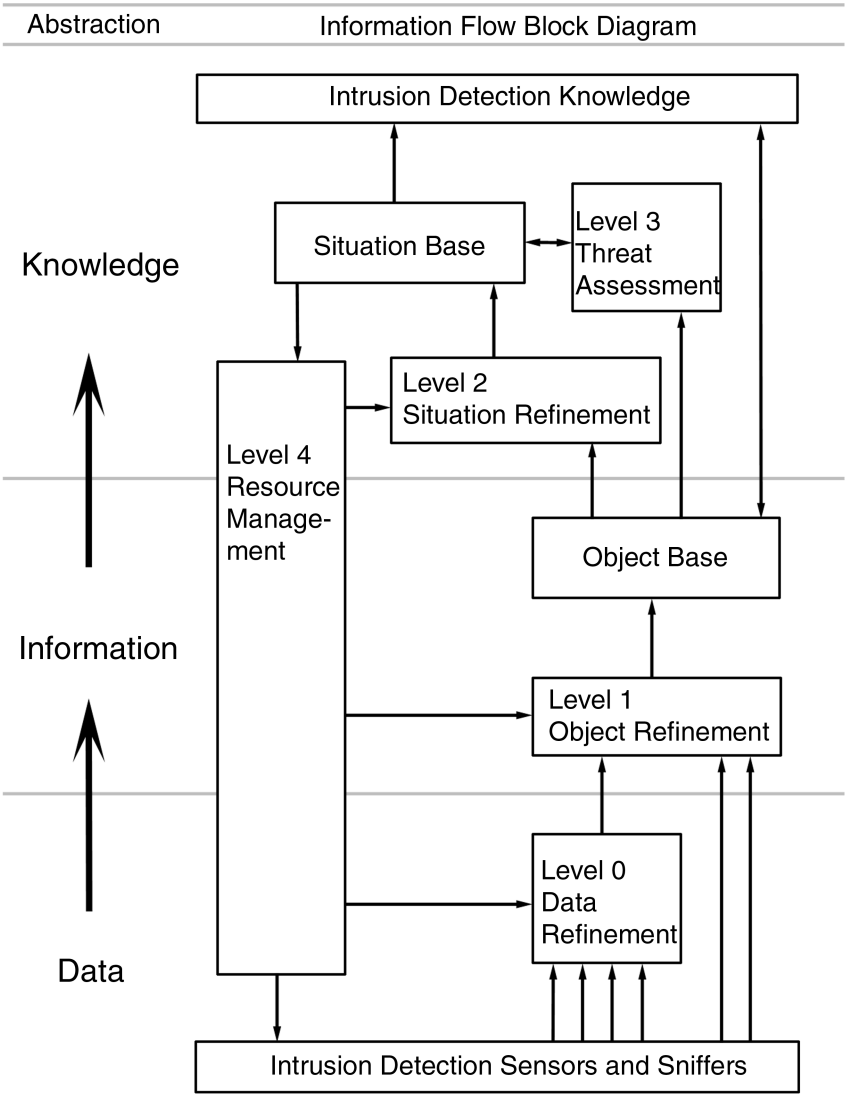


Figure 2.2: IDS Data Fusion Model (from [Bas00])

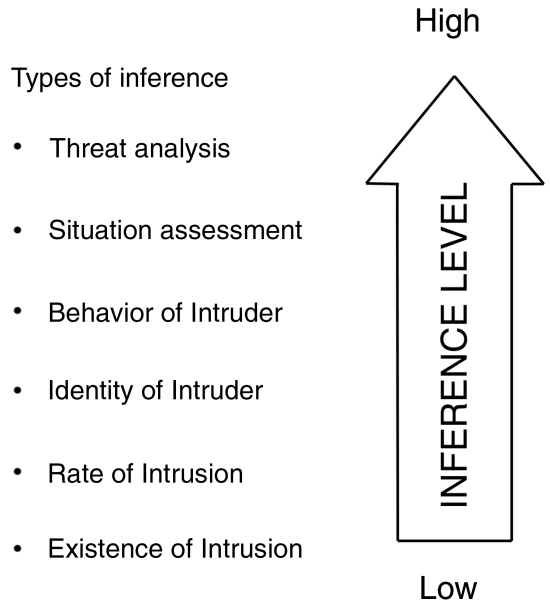


Figure 2.3: IDS Data Fusion Inference Hierarchy (from [Bas00])

The alert clustering function uses a similarity function to determine the similarity between two alerts. Instead of a probabilistic approach like EMERALD, MIRADOR uses an expert system approach in which the similarity requirements are specified using expert rules. The result of the alert merging function is comparable to EMERALD's meta alerts.

Burroughs, Wilson and Cybenko [BWC02] presents yet another approach. It considers intrusions from an attacker centered viewpoint, instead of the more common network centered viewpoint. By using a Bayesian multiple hypothesis tracking algorithm it tracks and identifies attackers.

The projects referred to in this section do not focus on a generic architecture of fusion-based IDSs, but rather on the implementation of data fusion techniques and algorithms.

2.4 Evaluation and Thesis Goal

Currently, IDSs are fairly rigid and rather 'dumb'. They lack the ability to make a distinction between important and less-important events. Together with the large number of false positives this often results in a huge amount of alerts that can easily overwhelm the user.

The application of multisensor data fusion to Intrusion Detection is part of

the solution to these problems. The goal of this thesis is to

- discuss the requirements of a fusion-based Intrusion Detection System;
- describe a good architecture of a fusion-based Intrusion Detection System. In other words an architecture that contributes to
  - elimination of false positives;
  - a more intelligent IDS, in the sense that the IDS partly automates the handling of true positives - thereby reducing the number of alerts the operator of the system has to inspect as well as prioritizing these alerts.

An architecture description leads to further insight in the fundamental design issues of fusion-based Intrusion Detection Systems. Besides, it provides a guideline for the application of multisensor data fusion techniques to the area of Intrusion Detection and the development of fusion-based Intrusion Detection Systems.

2.5 Summary

Information security is defined as the preservation of confidentiality, integrity and availability of information, and accountability of the usage of information. To achieve a certain level of information security, controls must be selected and implemented. Each control corresponds to at least one security function.

Intrusion Detection is the process of monitoring computer networks and systems for violations of the security policy. As a control, Intrusion Detection benefits the security functions deterrence, detection, recovery, restoration, prevention and compensation, with the emphasis on detection. Intrusion Detection Systems are systems that perform automated Intrusion Detection.

Current IDSs logically consist of sensor, analyzer and user interface components. The conclusion of an analyzer component puts the IDS in either a positive state, indicating an intrusion, or a negative state, indicating no intrusion. The state of an IDS can be true (appropriate) or false (not appropriate).

Based on the location of the sensor, a distinction is made between Network-based (NIDS) and Host-based (HIDS) IDSs. Another classification is based on the detection mechanism of the analyzer component. Misuse detection uses pattern-matching techniques and can only detect previously known attacks. Anomaly detection uses pattern-recognition techniques and a concept of 'normal behavior'. It relies on the assumption that all intrusions are

anomalous actions. Anomaly detection tends to have a high false positive rate.

Cooperation between IDSs enables the combination of advantages and minimizes the disadvantages of different approaches. The CIDF project was started by DARPA and addressed the lack of interoperability between IDSs. It triggered the creation of the IETF/IDWG that tries to establish a standard for IDS interoperability.

Multisensor data fusion provides an important functional framework for building next generation fusion-based IDSs. A number of (research) projects implement multisensor data fusion techniques. EMERALD uses a probabilistic approach to specify similarity between alerts, whereas MIRADOR uses an expert system approach to accomplish this. Burroughs, Wilson and Cybenko presents a Bayesian multiple hypothesis tracking algorithm to track and identify attackers.

Unlike other research projects, this thesis does not focus on the implementation of multisensor data fusion techniques. Instead, it discusses the requirements and derives a good architecture of fusion-based Intrusion Detection Systems. In order for the architecture to be considered 'good', it should at least enable the elimination of false positives, and contribute to a more intelligent attack analysis.

Chapter 3

Multisensor Data Fusion

This chapter provides an introduction to multisensor data fusion theory. It describes the history and some application domains of multisensor data fusion, as well as a number of (process) models and architectures.

3.1 History of Multisensor Data Fusion

Multisensor data fusion is about the synergistic use of sensory data from multiple sensors to extract the greatest amount of information possible about the sensed environment [WL90]. It is a rapidly evolving field based on a concept that is hardly new. The fusion of multisensory data is naturally performed by animals and humans. By using multiple senses they improve their ability to survive [HL97]. A barn owl for instance fuses visual and auditory information to accurately locate mice under very low light conditions [Ant95].

There are several definitions of data fusion. The definition that is used throughout this thesis is the definition given by Wald [Wal99]:

> Data fusion is a formal framework in which are expressed means and tools for the alliance of data originating from different sources. It aims at obtaining information of greater quality; the exact definition of 'greater quality' will depend upon the application.

Unlike most other definitions, this definition focusses on a conceptual framework for data fusion, instead of on the means and tools themselves. Note that this definition implies that, although the fusion process itself incorporates decision making, the subject of a data fusion system is not decision making, but rather providing information.

The first reports of the automation of data fusion functions are from the late 1970s. These reports referred to military systems. Throughout the 1980s, a lot of research was done with regard to data fusion. This research

was done mainly by the three U.S. military services, and much of its results were published in open literature. In 1986 the U.S. Department of Defense (DoD) established the Data Fusion Subpanel (DFS), a subpanel of the Joint Directors of Laboratories (JDL) Technical Panel for C3. The establishment of the JDL/DFS was a result of the concern that the three services were duplicating efforts. In the late 1980s a small number of military data fusion systems was operational. Since then, data fusion technology has rapidly advanced. What started as a loose collection of related technologies became an emerging engineering discipline [WL90, HL97, Yan99].

While the first data fusions methods where primarily applied in the military domain, in recent years these methods have also been applied to problems in the civilian domain. Examples of military applications are ocean surveillance, air-to-air defense, surface-to-air defense, battlefield intelligence, surveillance, and target acquisition, and strategic warning and defense. These applications focus on the location, characterization, and identification of entities. Civilian applications include the implementation of robotics, automated control of industrial manufacturing systems, development of smart buildings, and medical applications [HL97]. A more recent idea is the application of multisensor data fusion techniques to the area of information security [Sem99, A$^+$00, Bas00].

3.2 The JDL Functional Data Fusion Process Model

One of the results of the JDL/DFS efforts was the development of a data fusion process model. It provides a high-level functional view of the data fusion process. The JDL Functional Data Fusion Process Model, depicted in Figure 3.1, consists of eight components [WL90, Hal92, Ant95, HL97]:

- *Sources* provide input to the data fusion system. Possible sources are local sensors, distributed sensors, human input and a priori information from databases. Multiple sensors that are from the same type are called commensurate sensors, as opposed to noncommensurate sensors that are of different type.

- *Source Pre-Processing* is sometimes referred to as 'Level 0 Processing' or 'Process Assignment'. It covers initial signal processing and allocates data to appropriate processes. It enables the data fusion process to focus on data that applies most to the current situation and reduces the data fusion system load.

- *Level 1 Processing - Object Refinement* fuses sensor information to achieve a refined representation of an individual entity. It usually consists of four functions:

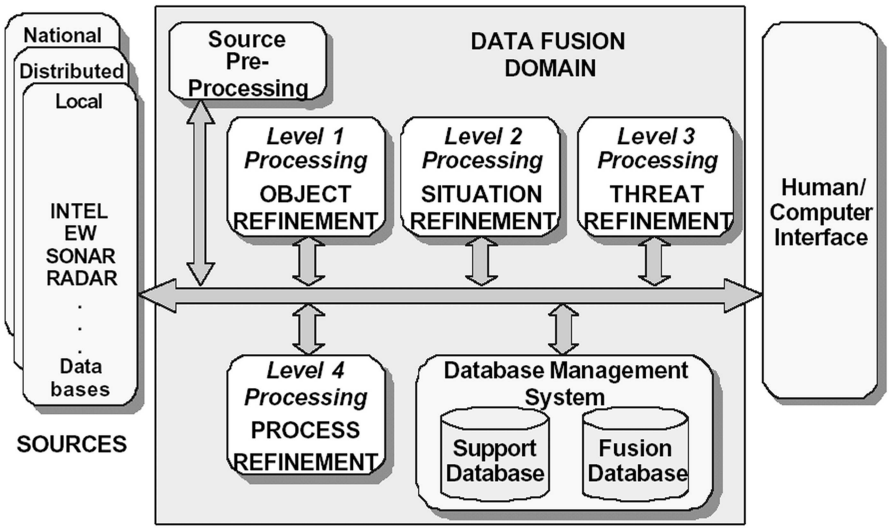


Figure 3.1: JDL Functional Data Fusion Process Model

- *Data Alignment* aligns data received from multiple sensors to a common reference frame;
- *Association* combines, sorts or correlates observations from multiple sensors that relate to a single entity;
- *Tracking* involves the combination of multiple observations of positional data to estimate the position and velocity of an entity;
- *Identification* combines data related to identity to refine the estimation of an entity's identity or classification.

Level 1 fusion benefits from the use of heterogeneous sensors, the employment of spatially distributed sensors and the application of non-sensor derived information.

- *Level 2 Processing - Situation Refinement* develops a contextual description of relations between entities. It focusses on relational information to determine the meaning of a group of entities. It consists of object aggregation, event and activity interpretation and eventually contextual interpretation. Its results are indicative of hostile behavior patterns. It effectively extends and enhances the completeness, consistency, and level of abstraction of the situation description produced by Object Refinement.

- *Level 3 Processing - Threat Refinement* analyzes the current situation and projects it into the future to draw inferences about possible outcomes. It identifies potential enemy intent and friendly force vulnerabilities. Threat refinement focusses on intent, lethality, and opportunity.

- *Level 4 Processing - Process Refinement* is a meta-process that aims to

optimize the overall performance of the fusion system. It consists of four key functions:

- *Performance evaluation* provides information about real-time control and long-term performance;
- *Process control* identifies the information needed to improve the multilevel fusion product;
- *Source requirements determination* determines the source specific requirements to collect relevant information;
- *Mission management* allocates and directs sources to achieve mission goals.

Part of the process refinement, in particular mission management, may be outside the domain of specific data fusion functions. It is therefore partially placed outside the fusion process in Figure 3.1.

- The *Database Management System* provides access to, and management of data fusion databases. It is the most extensive support function for data fusion processing. Its functions include data retrieval, storage, archiving, compression, relational queries, and data protection.

- The *Human-Computer Interface* allows human input into the data fusion process. It is also a means of communicating data fusion results to a human operator.

Antony [Ant95] presents three interpretations of levels 1, 2, and 3 in the JDL model. These interpretations are depicted in Figure 3.2. The first interpretation associates the three levels with input information classes. The second interpretation associates them with the answers to the questions where, when, what, why, how and so what. The third one associates the levels with the information product classes. These interpretations resemble the multiple levels of abstraction of the input, the fusion process, and its products.

Antony compares the multiple level-of-abstraction situation description that results from data fusion with a jigsaw puzzle [Ant95]. Individual puzzle pieces represent low level-of-abstraction information, such as color or texture. When several pieces are combined they might represent an object, such as a tree. The whole puzzle contains several aggregate objects, for instance a forest. When one starts with solving the puzzle, there is very little context, since there are no pieces in place yet. This means that a single piece can be interpreted in multiple ways. A blue piece, for example, could represent sky as well as water. Hence placing the pieces is quite difficult. At this stage, solving the puzzle is analogous to level 1 fusion. With the placement of more and more pieces, objects ('level 1') and groups of objects ('level 2') emerge. This makes placement of new pieces considerably easier.

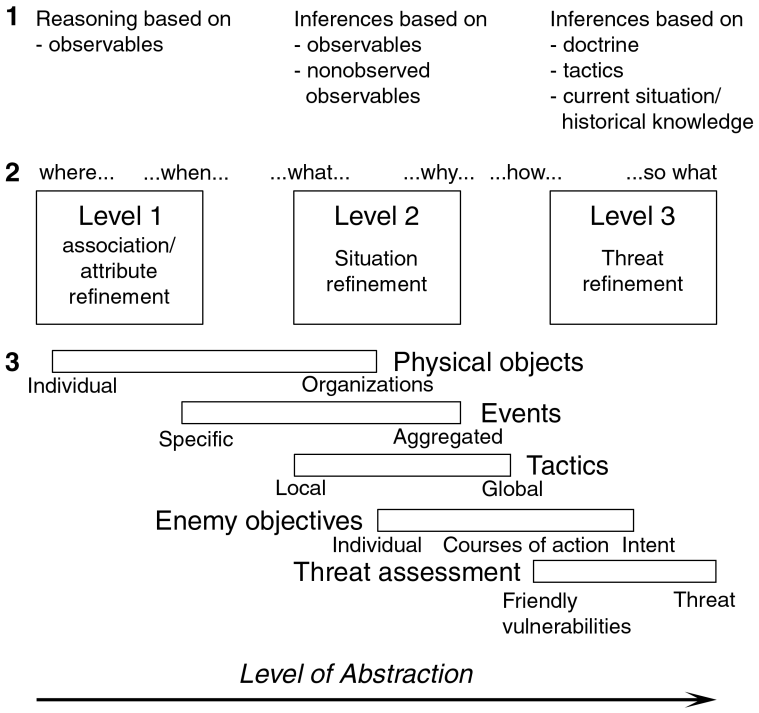


Figure 3.2: Interpretations of the JDL Model (from [Ant95])

One might even start to look for specific missing pieces ('level 4')[1]. Eventually, although some pieces might be missing, the whole scene (situation) is adequately described by the pieces that are in place.

3.3 Antony's Biologically Motivated Fusion Process Model

Antony [Ant95] presents another, biologically motivated, data fusion process model. As mentioned in section 3.1, humans and animals naturally perform data fusion. The biologically motivated fusion process model relies on the distinction between short-term, medium-term, and long-term memory, and associates the primary elements of data fusion with them. Where short-, medium-, and long-term *memory* are associated with the durability of information in *biological systems*, short-, medium- and long-term *knowledge* are associated with the durability of information in *artificial data fusion systems*.

A fusion system's short-term declarative knowledge is represented by the sensor observables. In general short-term knowledge is dynamic, perishable, and highly context-sensitive. This means that it implicitly depends on the current state of the fusion process, the output of nonprimary sensors, or non-sensor derived domain knowledge. This is in contrast with context-insensitive information, which has none of the aforementioned dependencies.

The system's medium-term declarative knowledge is represented by the current situation description at all levels of abstraction. It is the system's current relevant perception[2] of the environment. Although medium-term knowledge is learned and forgotten at a slower rate than short-term knowledge it is still relatively dynamic, perishable and context-sensitive. As a result of their context-sensitivity, medium-term and short-term knowledge require interpretation within context.

The fusion system's long-term declarative knowledge is represented by its factual knowledge base. The system's long-term procedural knowledge is represented by its procedural knowledge-base. Long-term knowledge is relatively non-perishable information that can be either context-sensitive, specialized, knowledge, or context-insensitive knowledge.

Long-term declarative knowledge can be further classified. It is considered to be either specific or general declarative knowledge. Specific declarative

[1]Note that there is no analogy for 'level 3' in this jigsaw puzzle metaphor. This is due to the fact that, once the puzzle (situation description) is complete, there is no need to analyze it any further to assess threats, for instance. The level 2 product is the highest abstraction sought for.

[2]Note that there can be a disparity between perception and reality. This is due to the inherent uncertainties in the data fusion process.

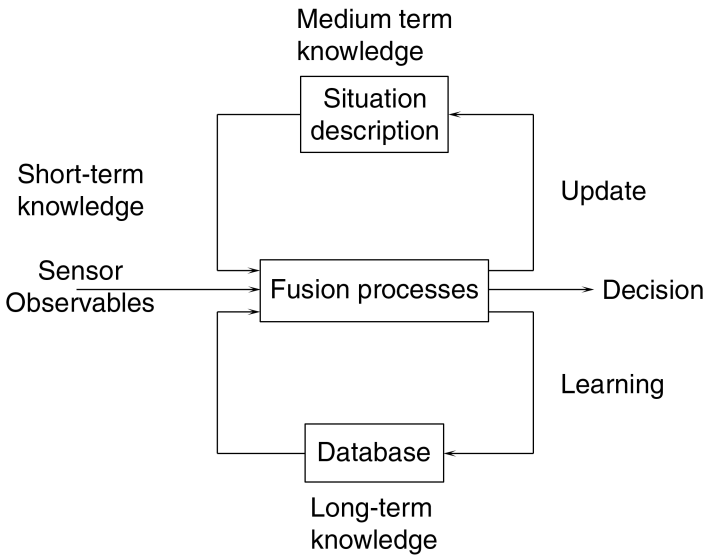


Figure 3.3: Biologically Motivated Fusion Process Model (adapted from [Ant95])

knowledge represents static facts, transformations or templates. General declarative knowledge does not represent a set of fixed attributes, but can characterize the value of, and relationships among individual attributes. Hence specific long-term declarative knowledge supports only non-model-based reasoning, whereas general long-term declarative knowledge supports model-based reasoning.

Figure 3.3 is taken from [Ant95] and provides a schematic overview of the biologically motivated fusion process model. The terms 'Update' and 'Learning' describe analogous operations. Distinct terms are used to indicate the difference of the timescale on which the knowledge is learned and forgotten.

Antony further decomposes fusion processes in three conceptually separate processes. These processes are:

1. Composition of knowledge sources;
2. Evidence accumulation;
3. Decision making.

Figure 3.4 shows this decomposition.

Knowledge composition brings together the knowledge sources - short-term, medium-term, and long-term - using various numeric and symbolic tech-

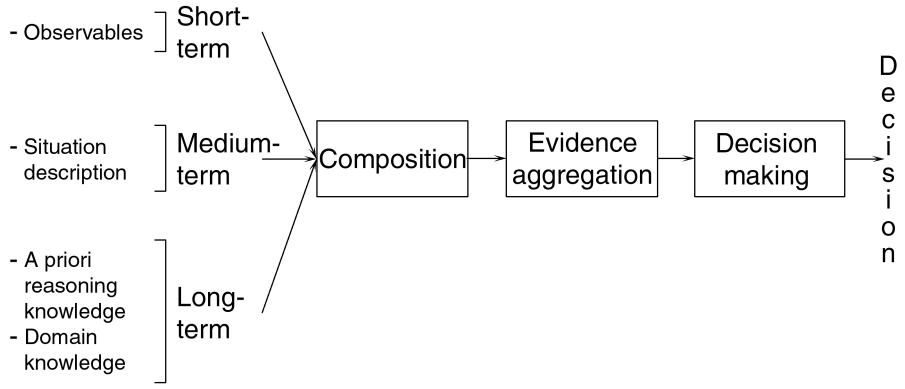


Figure 3.4: Decomposition of Fusion Processes (adapted from [Ant95]).

niques. Evidence accumulation, or evidence aggregation, uses these knowledge sources and combines and reasons with - possibly uncertain - data, rules, and decisions. The decision function uses the accumulated evidence to make a certain decision.

3.4 Data Fusion Architectures

A fundamental issue regarding data fusion systems is the question where in the data flow the fusion must take place, or in other words the choice of architecture. Hall and Llinas [HL97], as well as Hall [Hal92] describe three architectural approaches to the fusion of information at level 1 in the JDL fusion model. The first approach involves the fusion of raw sensor data, and is called centralized fusion (with raw data), or data level fusion (see Figure 3.5). It is the most accurate way of fusing data, but may also require much communication bandwidth since all raw data must be transmitted from the sensors to a central processing facility. The fusion of raw data is possible if commensurate sensors are available.

Another possible architecture is centralized fusion with feature vector data (see Figure 3.6). This is also called feature level fusion. In this architecture, feature vectors rather than raw data are transmitted to the central fusion process. The feature vectors are extracted from the raw data by the sensors. Since the feature vectors are a representation of the raw data, this approach inherently results in data loss. In practice, this is often less problematic than it sounds. Compared with data level fusion, feature level fusion has advantages that might outweigh the disadvantage of data loss. Although there is some data loss, feature level fusion enables the fusion of data from non-commensurate sensors and reduces the required communication bandwidth.

Autonomous fusion, or decision level fusion, is the third possible level 1

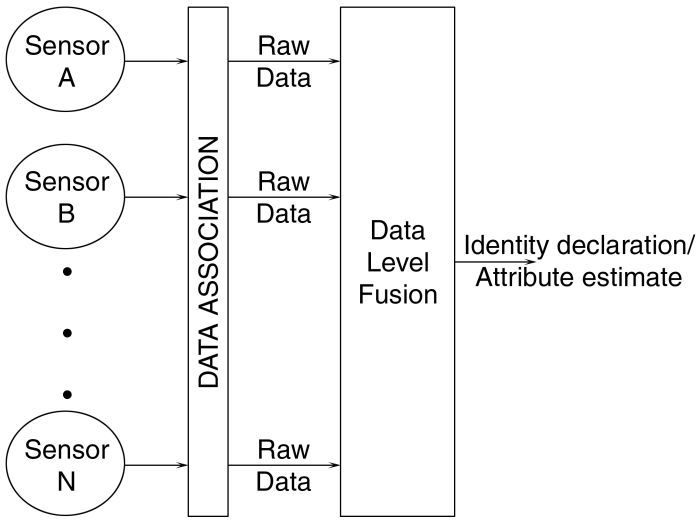


Figure 3.5: Data Level Fusion (adapted from [Hal92, HL97])

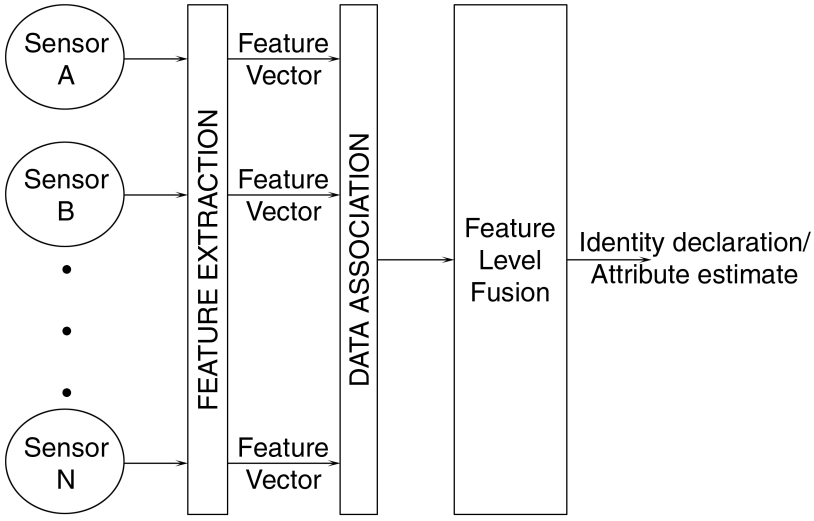


Figure 3.6: Feature Level Fusion (adapted from [Hal92, HL97])

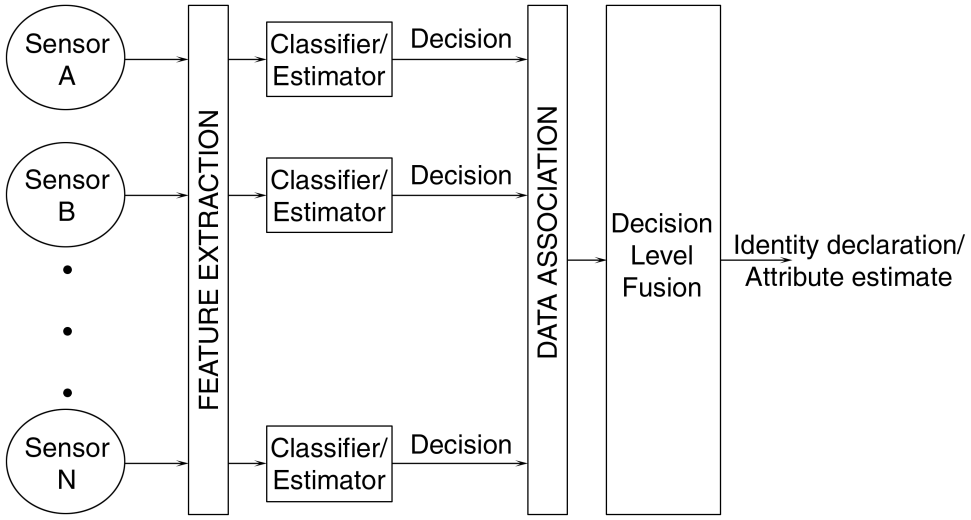


Figure 3.7: Decision Level Fusion (adapted from [Hal92, HL97])

fusion architecture (see Figure 3.7). Instead of outputting raw data or feature vectors, a sensor makes a decision based on its own single-source data. This decision, a declaration of identity or estimation of position and/or velocity, is the input for the fusion process. This too allows data from non-commensurate sensors to be fused. There is significant data loss compared with raw data fusion, and decision level fusion may result in a local rather than a general optimized solution.

There is no 'best' architecture in general. The choice of architecture depends on requirements and constraints, such as the available communications bandwidth, sensor characteristics, etcetera. For each application, the architectural advantages and disadvantages must be weighed against each other.

As described in section 3.2, the result of this level 1 fusion is interpreted in the next levels to achieve situation and threat assessments. The fusion processes involved in this next stages are more complex, since they typically use multiple types of expertise. Waltz and Llinas [WL90] present three problem solving approaches involving multiple experts, or knowledge sources. The three approaches are described as

- *Opportunistic* problem solving (see Figure 3.8(a)). Multiple experts are (usually) collocated. They each monitor the problem state and solution state and asynchronously contribute partial solutions to the overall solution, applying pieces of knowledge at the most opportune time;
- *Communicative* problem solving (see Figure 3.8(b)). The experts must use some type of communication channel in order to share problem

and solution state awareness, as well as to contribute their partial solutions;

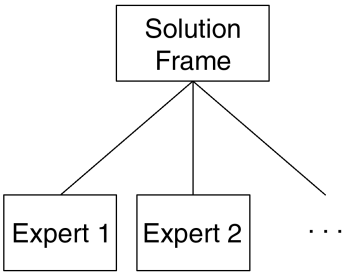
- *Cooperative* problem solving (see Figure 3.8(c)). Together the experts interpret the problem states and develop partial solutions.

The problem solving approach that is probably most used in higher level data fusion applications is 'blackboard processing'. This is a special case of opportunistic problem solving. All independent knowledge sources, or experts, have access to a central blackboard. All communication and interaction between the knowledge sources takes place using this blackboard, which is a global database containing the solution state. Each knowledge source can use, change and create solution state data stored on the blackboard. The knowledge sources respond opportunistically to changes in the blackboard, and are thus self-activating. At any given time a number of knowledge sources might want to respond to the situation described on the blackboard. A monitoring or control function is in general present to mediate between the knowledge sources that want to respond. The control function itself can be on the blackboard, and hence be modified by the knowledge sources. This control function has the ability to violate the opportunistic problem solving characteristic of the blackboard approach. Since the control function selects the knowledge source that may respond, blackboard processing reflects a communicative rather than opportunistic problem solving approach - although the model in itself is a form of opportunistic reasoning [WL90, Nii86]. The JDL model can be readily mapped to the blackboard paradigm [Ant95].

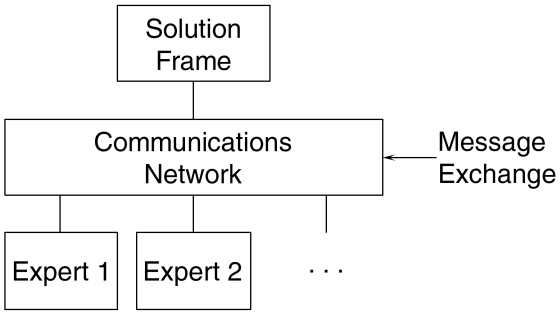
3.5 Development of a Data Fusion System

Hall and Llinas [HL97] remark that, while there are numerous examples of improved system performance by using multisensor data fusion, the implementation of effective data fusion systems is not simple. For instance, the combination of accurate with inaccurate data might produce worse results than the use of the accurate data alone. Seven questions are stated regarding the fundamental issues that should be addressed when building a data fusion system:

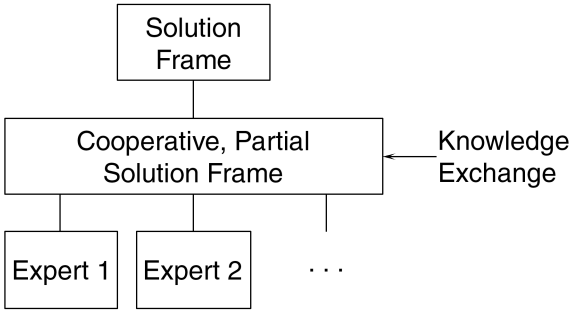
1. What algorithms and techniques are appropriate and optimal for a particular application?
2. What architecture should be used?
3. How should the individual sensor data be processed to extract the maximum amount of information?
4. What accuracy can realistically be achieved by a data fusion process?
5. How can the fusion process be optimized in a dynamic sense?



(a) Opportunistic problem solving



(b) Communicative problem solving



(c) Cooperative problem solving

Figure 3.8: Problem-solving paradigms

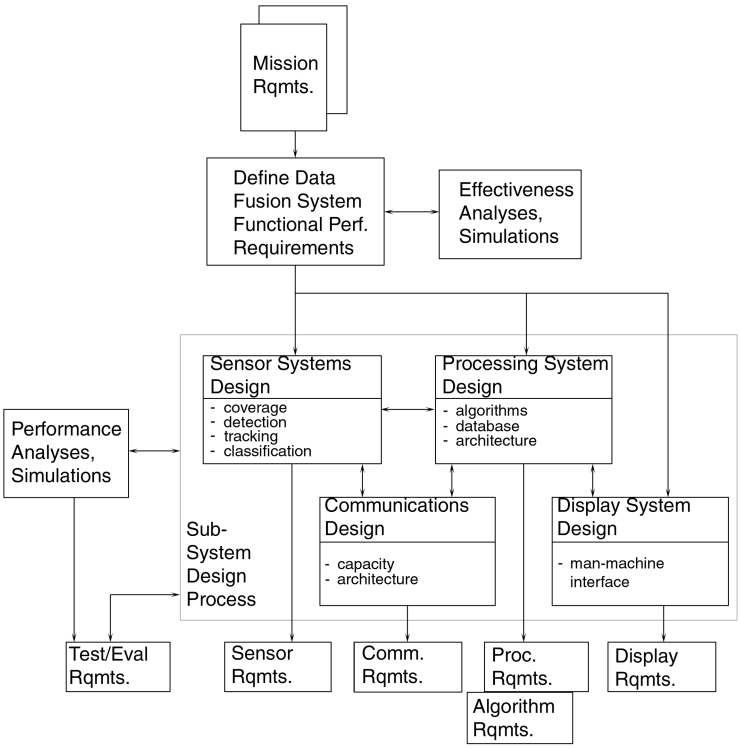


Figure 3.9: Fusion system design process (adapted from [WL90])

6. How does the data collection environment affect the processing?
7. Under what conditions does multisensor data fusion improve system operation?

A general structured fusion system design process is given by [WL90]. A simplified version is depicted in Figure 3.9. Designing a fusion system must start with a definition of the mission requirements. The mission requirements form a high level system specification. Next, the functional requirements of the data fusion system must be defined. Often, this requires decomposition of the system into functional subsystems. The functional subsystems must satisfy the mission-level effectiveness requirements. Simulations can be used to support the analysis of the effectiveness of the subsystems.

After decomposition, the functional requirements for the sensor systems, processing system, communications, and display system are specified. These subsystems must satisfy the functional performance requirements. Design of the subsystems results in the development of various requirements, which must be satisfied during implementation.

This thesis focusses on part of this design process, namely:

- identification of the 'mission' requirements;
- definition of functional requirements;
- development of the processing system architecture;
- effectiveness analysis by using test cases.

This limited focus is mainly due to time constraints.

3.6 Summary

Multisensor data fusion is a rapidly evolving field about the synergistic use of data from multiple sensors. Its use in military systems was first reported in the late 1970s. In recent years, data fusion methods are also applied to problems in the civilian domain.

The JDL/DFS was established to coordinate the data fusion research performed by the three U.S. military services. One of the results of its efforts is the JDL Functional Data Fusion Process Model. The JDL model provides a high-level functional view of the data fusion process.

The Biologically Motivated Fusion Process Model was developed by Antony. It builds on the fact that humans and animals naturally perform data fusion, and relies on the distinction between short-term, medium-term, and long-term memory. Short-term, medium-term, and long-term knowledge are associated with the durability of information in artificial data fusion systems.

There are three types of level 1 fusion architectures, based on the position of the fusion process in the data flow. The three architectures are data level fusion, feature level fusion and decision level fusion. Higher level fusion processes are more complex and make use of multiple knowledge sources. There are three types of problem solving involving multiple knowledge sources: opportunistic, communicative and cooperative problem solving. Probably most used in higher level fusion is 'blackboard processing', a special case of opportunistic problem solving.

The implementation of effective multisensor data fusion systems is not simple, and a number of fundamental issues should be addressed when developing such a system. Hall and Llinas describe these issues.

Chapter 4

Fusion-Based Intrusion Detection Systems

This chapter provides a detailed description of fusion-based Intrusion Detection Systems. It starts with a discussion on the purpose of a fusion-based IDS. Next, a functional-level analysis of a fusion-based IDS is presented. Finally, a generic architecture is derived.

4.1 The Purpose of a Fusion-Based IDS

In Chapter 2, a number of problems with 'traditional' IDSs are outlined. Amongst these problems, the high number of positives in general, and of false positives in particular, are identified as major problems with these IDSs. The application of multisensor data fusion to intrusion detection aids in solving these problems.

Recall the definition of data fusion, stated earlier in Chapter 3:

> Data fusion is a formal framework in which are expressed means and tools for the alliance of data originating from different sources. It aims at obtaining information of greater quality; the exact definition of 'greater quality' will depend upon the application.

This definition implies that

1. a fusion-based IDS should provide information. This means that decision making is not part of the tasks of a fusion-based IDS, apart from decisions made in the fusion process itself. However, decision making - either automated or manual - can be supported by the information provided by a fusion-based IDS.
2. a fusion-based IDS should provide information 'of greater quality' than

information provided by a non-fusion-based IDS. The greater quality of this information lies in

- reduction of the number of positives that must be inspected by the operator of the system;
- elimination of false positives.

The positives that result from 'traditional' IDSs are in general too fine-grained, too low-level. A high-level situation description is required to provide a coarser view. Compare this to the puzzle-solving metaphor described in Section 3.2. The reports of 'traditional' IDSs are comparable with individual puzzle pieces. Sometimes a number of pieces are assembled, to represent a port scan for instance. There may be duplicate pieces, when multiple IDSs report the same observation. There may be pieces that do not belong to the puzzle - false positives. But in general there is no adequate picture of the puzzle, hence no overall understanding of the situation.

Burroughs argues that "it is not the *attack* but rather the *attacker* against which networks must be defended" [BWC02]. Burroughs therefore suggests to perform intrusion detection from an attacker centered viewpoint, as opposed to a much more common network centered viewpoint. Unfortunately, this view ignores the possibility that an attack might be a coordinated effort between multiple, possibly geographically separated, attackers. An example of such an attack is the 1997 *Langley Cyber Attack*, a coordinated global e-mail bomb attack against the Langley Air Force Base e-mail infrastructure [BFGW98]. In the end it *is* the attack against which one should defend, although a network centered view is indeed not sufficient: a single attack might be spread over multiple networks or network segments, or be related to other attacks in other networks - for instance because the other attacks are performed by the same attacker. Tracking the activities of individual attackers through multiple networks is therefore still useful, but is merely a subgoal of a complete fusion-based IDS, not *the* goal.

Hence, it is not the attack, nor the attacker that should be considered most important. What is needed is a *situational view*, involving both. Figure 4.1 depicts an example of such a view. It shows three different attacks, launched against three networks or network segments by three individual attackers. Attacker 1 is involved in a single attack against Networks 1 and 2. Attacker 3 is also involved in a single attack, but this is a coordinated attack in which Attacker 2 is also involved. Furthermore, Attacker 2 has also launched his 'own' attack against Network 1. Hence Attacker 2 is participating in two attacks: one coordinated attack and one individual attack. Note that not all actions of an attacker (light gray) are attacking, or intrusive, actions (dark grey). Some actions that the attacker performs might be legitimate actions. Note also that not all of an attacker's actions - either legitimate or intrusive - are directly observable in the network or network segment. For instance, during an attack, an attacker might look up vulnerabilities known to exist on a default installation of some software on the targeted host. While this is clearly an action that belongs to the attack, it is not directly observable from the network.

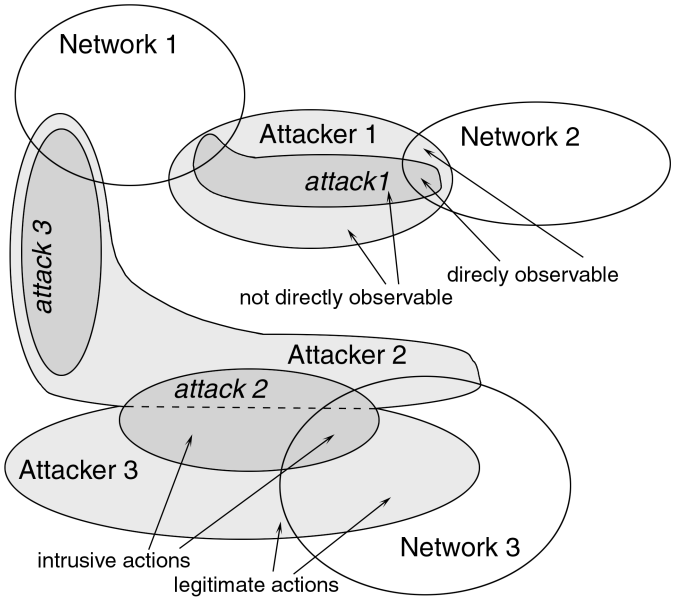


Figure 4.1: Situational view

The information provided by a fusion-based IDS is therefore information concerning the current situation. The greater quality of this information is due to intelligent attack analysis and the use of data from multiple sensors, and results in a reduced amount of false positives. A high-level picture of the attacks that take place, and the attackers that are involved, forms a situational view. This situational view provides a coarser view than 'traditional' IDSs and thereby reduces the, otherwise high, number of positives that result from an intrusion.

4.2 Functional-Level Analysis

Section 4.1 compares the output of traditional IDSs with puzzle pieces, and identifies a fusion-based IDS as a means to solve - at least partly - the puzzle, sorting out pieces that do not belong to the puzzle and joining pieces that belong together. Solving the puzzle requires and results in a multiple level-of-abstraction view of the situation. To build a fusion-based IDS, a framework that supports this kind of abstraction must be used.

The JDL model, described in Section 3.2, inherently supports multiple levels of abstraction. Use of this model therefore provides a natural way for the IDS to coarsen the data and arrive at a high-level situation description. This section presents a functional-level analysis of a fusion-based IDS per level of the JDL model. This is an elaboration on the work in [Bas00], and will be used as a basis for the development of a generic architecture for

fusion-based IDSs.

Bass [Bas00] lists the functions of the five levels in the IDS Data Fusion Model. Since this model is an application of the more generic JDL model, the functions resemble those described in Section 3.2:

- *Level 0*: Filtering of raw data;
- *Level 1*: Alignment and correlation of data and placing the resulting objects in context in an object base;
- *Level 2*: Detection of aggregate sets of objects by their coordinated behavior, dependencies, common points of origin, common targets, correlated attack rates and other high-level attributes;
- *Level 3*: Assessment of current situation and identification of future threats;
- *Level 4*: Refinement of the entire process to further refine detection.

When viewing the JDL model from an object-centered view [Ant95], level 1 is associated with individual objects (events), level 2 with organizations of objects (for instance, attacks) and level 3 with the higher level analysis of those organizations. Hence, each level provides a more abstract view of lower-level information.

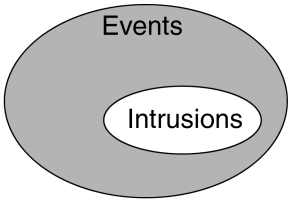
Based on the function descriptions by Bass, and the generic JDL model, a more detailed functional-level requirements analysis can be made. In this section the specific tasks of a fusion-based IDS belonging to each of the five levels are determined. These tasks are described, and the key information requirements for these tasks are identified. The bottom-up approach that is used reflects the increasing level of abstraction associated with each level. While this provides a good description of the per-level functionality that is needed, it is not necessarily true that the information flow follows these levels in ascending order. Section 4.3 treats this in further detail.

4.2.1 Level 0 - Source Pre-Processing/Data Refinement

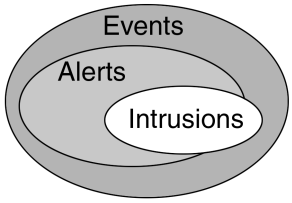
At level 0 the initial signal processing takes place. The raw data that is observed by the sensors consists of *events*[1]. An event is the result of an *action*. An action is defined in [Den87] as an operation performed by some initiator - normally a user - on or with a resource managed by the target system.

These events are typically observed on a host - in the shape of rows in an event log/syslog, application log, etcetera - or 'on the wire' - in the shape of

[1]Note that an event is not an intrusion per se. The arrival of a single network packet, for instance, is an event, as is a failed or succeeded log on. Hence an intrusion or attack consists of one or more events, but not every event is part of an intrusion. See also figure 4.2(a).



(a) Not all events are intrusions.



(b) Not all alerts are intrusions.

Figure 4.2: Events, alerts and intrusions.

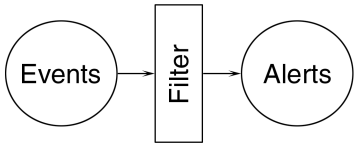


Figure 4.3: Alerts are events that are considered important.

network packets. These two types of observables correspond with two types of non-fusion-based IDSs: NIDSs and HIDSs. An observed event is either important for the other fusion processes to know of, or it is not. The preprocessing of the observed events consists of a filtering function that filters the raw data to exclude as much unimportant events as possible from further processing. Events that are particularly important are events that are part of an intrusion. The filter can be dynamically adjusted by the level 4 process to let through other important events. For instance, during an attack *all* events that originate from the attacker might be considered important, instead of just the events that are part of an intrusion. It can also be adjusted to block events that are no longer important. For instance, when during an attack all events that originated from the attacker were considered important and now the attack has ended. Events that pass through the filter are thus events that are deemed important, and are called *alerts*. See also figures 4.2(b) and 4.3.

To be able to preprocess or filter the raw data, the level 0 process requires the following information:

1. The raw data observed by the sensor;
2. A definition of
   (a) Events that should pass the filter, in other words a definition of important events, or;
   (b) Events that should be blocked by the filter, in other words a definition of unimportant events.

3. A filtering procedure.

Note that the filtering method described by 2a corresponds with the method used by IDSs that perform misuse detection - the important events being intrusions - while the method described by 2b corresponds with the method used by IDSs performing anomaly detection - the unimportant events being 'normal behavior'. If the filtering method of 2a is used, the signature that matched and allowed the data to pass through the filter provides a preliminary estimation of the identity (or type) of the event.

4.2.2 Level 1 - Object Refinement

Level 1 processing refines the objects resulting from level 0 processing. The result of the level 1 process answers the question "*where* and *when* did something happen?" (see Figure 3.2).

The alerts that passed through the level 0 filter are first aligned to a common reference frame. This alignment facilitates the comparison and further processing of multiple alerts that originate from different sources. During the alignment, relevant features are extracted from the raw alert. When combined, these features form an abstract description of the raw alert. Alerts originating from different sensors are likely to possess different kinds of features. A sensor that observes network traffic in general knows little more of the target of an attack than its network address. A sensor located on and observing a single host is more likely to know much more about this host. If the observed host is the target of an attack, the alerts originating from this sensor may include for instance detailed information about the application or process that was targeted by the observed event. The reference frame should therefore provide enough room for these different kinds of features to be represented. The alignment stage requires the following information:

1. Raw alerts, i.e. raw data that passed the filter during level 0 processing;
2. A common reference frame to which the alerts should be aligned;
3. An alignment procedure.

An example of a possible reference frame is the Intrusion Detection Message Exchange Format (IDMEF) data model [IDW], which is part of the results of the IETF/IDWG. A draft[2] version of the IDMEF data model is shown in Figure 4.4. It shows various groups of classes: the core classes, the time classes, the assessment classes and the support classes. The core classes form the core of the model. The Alert class is in the center of it.

[2]The IDMEF data model shown here is based on draft version 0.6.

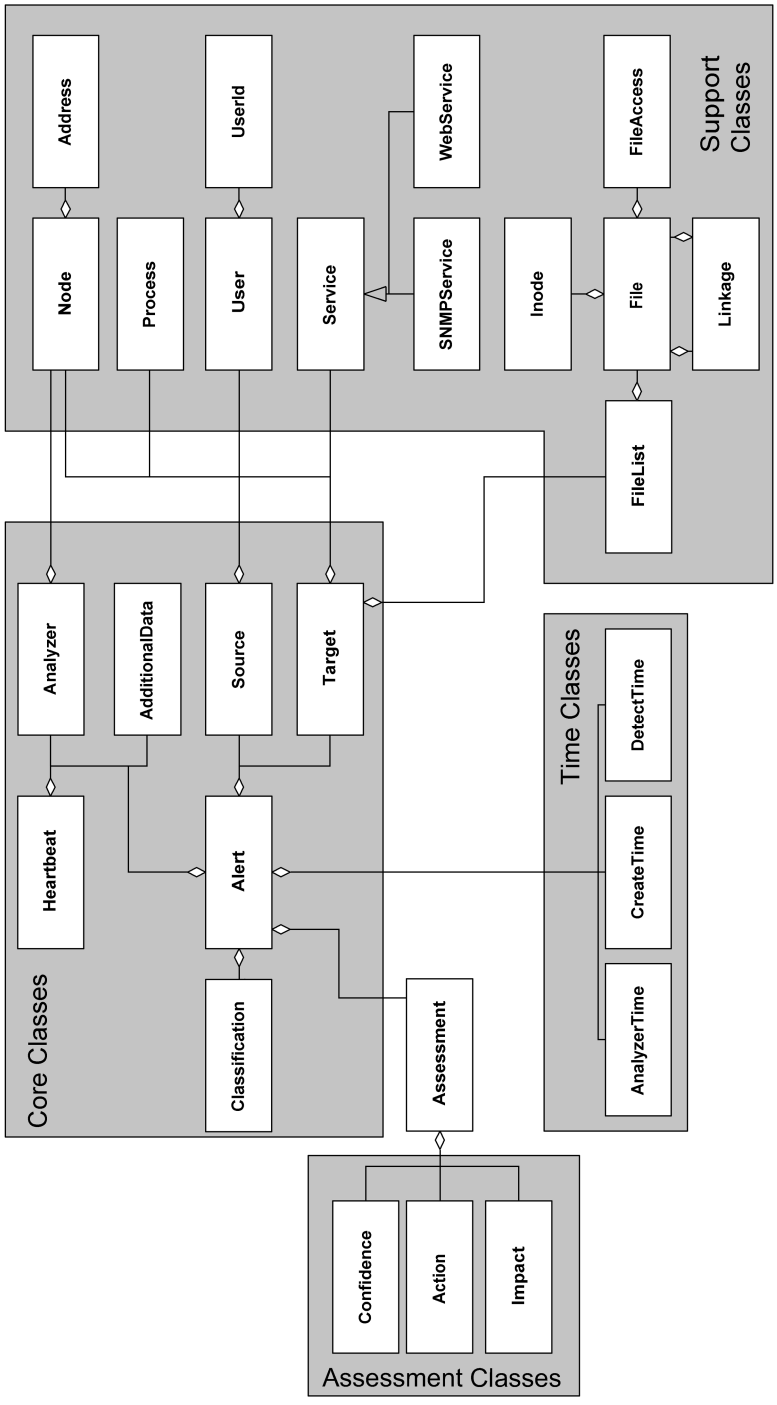


Figure 4.4: IDMEF data model.

The Alert class has a list of attributes that can be used to describe a specific alert. The analyzer, source, and target attributes - all part of the core classes - can be described in further detail by using the corresponding support classes. The target, for instance, can specify the targeted node (device), process, user, service, and/or file(s). Depending on the data reported by the sensor, and its level of detail, some parts of the model are used and filled in, while other parts are left unused. For an in-depth explanation of the IDMEF data model and its various classes, refer to [IDW].

Different observations of events can refer to one and the same event. This is the case if a single event is observed by two or more different sensors. If multiple observations pass through the level 0 filter, this results eventually in multiple (aligned) alerts referring to a single event, thus in fact being a single alert. Recognition of such associated alerts is performed during the next stage in level 1 processing, the association stage. A group of associated alerts is called an *alert track*. A single alert that has no associated alerts forms an alert track on its own.

For each alert that is processed during the association stage there are exactly two possibilities with regard to the event that produced it:

1. The event also produced other alerts that already belong to an alert track. This means that the alert being processed is associated with that particular alert track;
2. None of the existing alert tracks is the result of the event that produced the alert. In other words, the alert is the result of the observation of a new event and forms its own alert track.

These two possibilities are shown in Figure 4.5. In this figure, two sensors are depicted while two events take place. Sensor 1 observes only Event 1, Sensor 2 observes both events. All observations pass the filters, resulting in three alerts (one for Sensor 1, two for Sensor 2). Both Alert 1.1 and Alert 2.1 are the result of an observations of Event 1. Hence Alerts 1.1 and 2.1 are associated alerts. Alert 2.2 is the only alert produced by Event 2.

During data association, data fusion systems often also consider the possibility that the observed signal is a false alarm. A false alarm is usually the result of noise observed by the sensor and interpreted as a detection. At this level of data fusion in intrusion detection there are no such things as false alarms. Unlike for instance some military applications, the sensors do not observe objects (events) that are of no importance at all[3]. Even events that comprise legitimate traffic contain information that is relevant for a situation description. The only problem with reporting events that comprise legitimate traffic is the risk of overloading the system. Therefore the level 0 filter blocks 'unimportant' events - as described earlier - where

[3]Sensors that are used in a military situation for anti-submarine warfare might observe a whale, for example. A whale has clearly nothing to do with anti-submarine warfare, and a reported observation is therefore a false alarm.

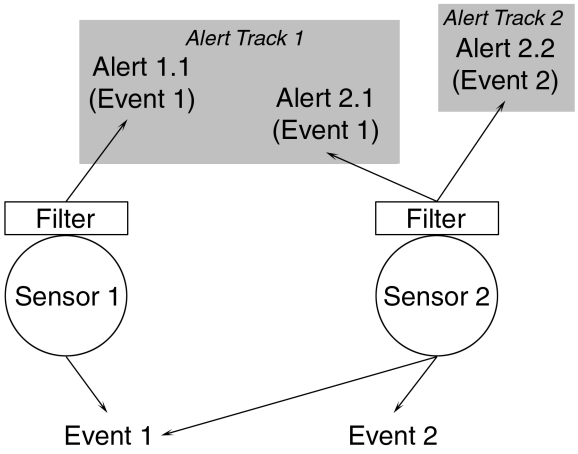


Figure 4.5: Alert association.

'unimportant' is a context-sensitive concept. It is possible that the observed event did not rightly pass the filter[4], and hence is mistakenly considered important. Alerts resulting from such a mistake are considered false positives in traditional IDSs, and will be dealt with later on.

The association process must therefore decide if the alert being processed is due to a new event or is associated with an existing alert track. To diminish the amount of possible associations, the alerts that are unlikely to be associated with the alert being processed are not considered in the decision step. They are not discarded, but merely ignored for the moment. This elimination of unlikely associations, the very first step in the association process, is called *gating* [Hal92]. The gating is performed by selecting alerts from the set of previously processed alerts. The selection criteria are based on the possible values that the alert's attributes could have had in the past, or may have in the future[5], and are therefore the result of a function on the attributes of the alert being processed.

After unlikely associations are eliminated, the remaining alerts are compared with the alert being processed. The comparison results in an indication of the similarity of all possible association pairs. The most likely association is the association of the pair of alerts with the highest similarity. Only if this similarity is 'high enough', the association process decides to assign the new alert to the alert track that the existing alert belongs to. The assignment results thus in either a new alert track or the extension of an existing alert track.

A decision that must be made when implementing a fusion-based IDS is whether the assignment of an alert to an alert track is irrevocable. If so, each alert is only considered once for assignment. If not so, the assignment

[4]In other words, did pass the filter but was in fact unimportant.
[5]Alerts are not guaranteed to arrive in chronological order.

decision made for an alert might be reconsidered after n other alerts have arrived. The value of n is also called the number of scans [Hal92].

Association requires the following information:

1. An alert (new or 'revisited');
2. All alerts in the existing alert tracks;
3. A model or definition of possible earlier attribute values, based on the current attribute values;
4. A gating procedure;
5. A definition of similarity;
6. A similarity computing procedure;
7. A similarity threshold for association;
8. An assignment procedure that decides whether or not the alert should be assigned to an existing alert track.

Remember that the (events underlying the) alert tracks are still not guaranteed to be part of an attack, or intrusion. They are merely deemed to be important for the fusion process as a whole. Every time an alert is assigned to an alert track, however, the confidence in the importance of the alerts that are part of that alert track increases. Based on confidence characteristics of the sensor that reported an alert, the initial confidence in a single-alert track can be calculated.

Confidence calculation requires:

1. An alert;
2. Confidence characteristics for the sensor that created the alert;
3. The alert track that the alert is assigned to;
4. The expectation of the existence of prior observations (alerts) in the alert track;
5. A confidence calculation procedure.

If the level 0 filter made use of a definition of important events, and knows of the reason why an event is considered important, the filter could attach an indication of this reason to the alert, thereby estimating the identity of the observed event. This is what happens amongst others in current misuse detecting IDSs: since a misuse detecting IDS has a list of signatures that are matched against an event, and a signature belongs to a specific intrusion, the IDS can indicate what intrusion it 'thinks' is going on.

The next, and final, step in level 1 processing is refining the estimate of the identity of the observed event. When an event is reported for the first time it may be difficult to unambiguously determine its identity. Subsequent reports of the same event may contain data that is complementary to the data that is already known. These data are combined with the data that was reported earlier. Hence, with every reported alert, the estimate of the identity of the event can be refined. A variety of pattern recognition techniques can be used to estimate the identity of the observed event. [Hal92] lists a number of these techniques.

Combination and identity estimation have the following key information requirements:

1. All alerts in an alert track;
2. An attribute combining/attribute refinement procedure;
3. An identity estimation (pattern recognition) procedure.

The collection of refined alerts that result from level 1 processing forms the Object Base that is used as basis for level 2 and further processing.

4.2.3 Level 2 - Situation Refinement

Unlike object refinement (level 1), situation refinement, as well as threat assessment, is relatively ill-defined. While object refinement is able to directly adapt well-established data association and estimation techniques, situation refinement and threat assessment currently do not have such a well-established basis. There is much less consensus on how to perform situation refinement and threat assessment than there is consensus on how to perform object refinement [WL90, Lam99]. This implies that, compared to object refinement, there are more choices to be made when designing and implementing levels 2 and 3 of a specific[6] fusion-based IDS. Therefore the following sections are necessarily less detailed than the previous section.

The result of level 2 processing answers the questions "*what* did we see?" and "*why* did we see that?" (see also Figure 3.2). Where level 1 processing involves objects, level 2 processing involves relations between, and groups of, these objects. The objects in level 1 are alerts. *What* we did see is expressed by two main types of relationships between alerts, in other words two types of alert aggregations:

1. Alerts that together make up an *attack*;
2. Alerts that together represent the *behavior of a single attacker*.

[6] As opposed to a generic fusion-based IDS, which is the subject of this thesis.

These are different types of aggregations, because a single attacker can be involved in multiple attacks and multiple attackers can be involved in a single attack (see also Figure 4.1). Furthermore, not all attacker behavior is necessarily part of an attack. On the other hand, all attacks are necessarily the result from attacker behavior. Hence alerts that make up an attack are a subset of alerts that represent attacker behavior, and alerts that represent attacker behavior are a subset of the set of all alerts. The set of alerts, again, is a subset of abstract representations of all events.

Note that an alert aggregation (attack or attacker behavior) can consist of a single alert, just like an alert track.

Why we did see this becomes clear when the relationships between attackers and attacks are analyzed. This shows which attackers are involved in which attack.

It is at this level that the notion and concept of false positives becomes important. At the lower levels there were no false positives, since all alerts are representations of events that do indeed exist. However, not all alerts are necessarily part of an attack, or resulting from attacker behavior. From a level 2 point of view, an alert that is neither part of an attack, nor the result of attacker behavior is a false alarm.

Finding alerts that represent the behavior of a single attacker can be viewed as a tracking problem[7]. The actions of an attacker leave a trail of alerts. The tracking procedure that is needed resembles what is done during object refinement at level 1. There, an event is 'tracked' by associating observations, or alerts, that belong together. With attacker tracking, an attacker is tracked by associating alerts that result from (the behavior of) a single attacker. Attacker tracking is different from the creation of alert tracks in that it must also consider false positives. Therefore it must take into account amongst others the belief that the level 1 process has in the importance of an alert track. A low belief in importance at level 1 might be due to a false positive at level 2. The most obvious way to track an attacker is to look at the source of events. Events that have the same source are likely to be the result of actions by one and the same attacker.

The second type of alert aggregations that are considered at this level are attacks. Like an attacker, an attack leaves traces. In fact, these are the traces left by one or more attackers while performing an attack. Traces left by a single attacker, either due to attacks or legitimate actions, provide insight into the identity, intentions, and sophistication of that specific attacker. Traces left by one or more attackers during an attack provide insight into the identity, intentions, and sophistication of that attack. So like tracking an attacker, recognizing attacks can be viewed - at least partly - as a tracking problem.

But there is more to situation refinement than tracking alone. Further

[7]This is also the view that [BWC02] takes.

reasoning is needed to take the context, the current situation, into account, and interpret the meaning of this situation. Hall [Hal92] lists the following types of reasoning:

- Pattern recognition utilizing uncertain, incomplete, and conflicting data;
- Spatial and temporal reasoning;
- Establishing cause-effect relationships;
- Prediction of future events/activities;
- Planning;
- Induction[8];
- Deduction[9];
- Abduction[10];
- Learning.

Reasoning involves the use of both long-term and medium-term knowledge. For instance, at this level the (network) environment must be taken into account. In particular, locations of devices that perform Network Address Translation (NAT) [EF94] can be important information. On a network, a NAT device changes source and/or destination of a network packet. By maintaining an internal table of mappings, the NAT device can translate network traffic back and forth. This means that when two sensors are deployed, one at each side of the NAT device, both sensors might see different source and destination addresses for the same event. Other important environmental information consists of a list of known compromised hosts. When an attacker compromises a host, an action initiated from that host might be initiated by the attacker, and be part of an ongoing attack. Hence the needed environmental information can be both relatively static (e.g. NAT device location) and dynamic (e.g. compromised hosts).

The interrelations between attacks and attackers must also be taken into account. An individual is not an attacker until he actually engages in an attack[11]. Hence the tracking of an attacker relies on the recognition of at least one attack he is involved in. On the other hand, knowing that a

[8] Establishing general concepts based on examples of specific cases or instances.

[9] Reasoning from general principles to specific conclusions about a particular case.

[10] Establishing parallels or analogies.

[11] This raises the question whether someone whose machine or address is unknowingly and unwillingly used in an attack must also be regarded as an attacker. The answer is twofold. When the machine is not in the view of the fusion-based IDS, it cannot be known to the fusion-based IDS that the attacker is in fact the victim of another attack. Hence in this case the victim is considered an attacker. If, however, it is known that the attacking machine was the victim of an earlier attack - for instance because the attacking machine is also monitored by the fusion-based IDS - the attack is contributed to the earlier attacker, not to the (owner of) the attacking machine.

certain individual is an attacker might aid in recognizing attacks in ambiguous behavior by that individual. The mere fact that the source of an alert is actively involved in one or more attacks raises the suspicion that that specific alert is also part of an attack. However, there's still the possibility that the alert results from legitimate behavior, and hence should be considered a false positive.

By making use of (a subset of) the types of reasoning listed above, a human expert is able to make inferences on the current situation. Enabling automation of situation refinement involves enabling a computer to reason like a human expert. In any case, the result of the situation refinement stage is a high-level overview of the situation. In this context, high-level means that the focus is on aggregations of individual events - attacks and attacker behavior - instead of on the individual events alone. By focussing on aggregations instead of individuals, a coarser representation of the situation is attained.

4.2.4 Level 3 - Threat Assessment

Level 3 processing answers the questions "*how* did all this happen?", and "*so what*?". During level 3 processing, the threat of the current situation, i.e. the individual attacks and the attacker behavior identified in level 2 processing, is determined through further reasoning with the situation refinement result. Apart from the possible severity of the result of an attack, the vulnerability of the target must be taken into account. Threat assessment enables prioritizing alert aggregations, thereby aiding the user to focus on the most threatening attacks.

One of the questions that is answered by threat assessment is whether or not a certain attack was successful. The outcome of an attack is not only important environmental information for situation refinement, but is also an indication of the vulnerability of a machine, and potentially the sophistication of the attacker(s) involved. By analyzing the outcome and various other characteristics of the attacks an attacker is involved in, an attacker profile can be created. Such a profile captures the capabilities of the attacker. For example, attacker profiling might classify an attacker as either script-kiddy or cracker. A script-kiddy is generally regarded as a person incapable of performing a truly new attack. Instead, he makes use of an automated script, written by someone else, that provides a step by step 'walk-through' of an attack. Crackers[12] on the other hand, are persons that *do* create and build new attacks, and are therefore likely to pose a larger threat than script-kiddies.

Another aspect of threat assessment is vulnerability assessment. If an attack is taking place against a machine that is known to be vulnerable to

[12]There is a lively discussion going on regarding the difference - if any - between crackers and hackers. Here, the term 'crackers' is used as 'hackers who *misuse* their capabilities'. Depending on your definition of a hacker, these two might be interchangeable.

such an attack, the threat is higher than when the machine is known not to be vulnerable. Vulnerability assessment combines various types of information. Relatively static information, such as a database of known vulnerabilities, but also more dynamic information, such as versions of the operating systems and other software that runs on hosts in the network.

Yet another goal of threat assessment is to project the current situation into the future. This means hypothesizing the outcome(s) of an attack and the intent of an attacker. If an attack is likely to succeed, it poses more threat than if it is likely not to succeed.

Threat assessment involves the same kind of reasoning and inference drawing as situation assessment. Again, automation requires enabling a computer to reason like a human expert.

4.2.5 Level 4 - Resource Management

Level 4 processing performs evaluation of the total fusion process. After evaluation, it adjusts the fusion process and cues the sensors to dynamically improve the quality of the output.

One of the most obvious ways of sensor cuing is adjusting the level 0 filter. By adjusting the filter, the level 4 process effectively affects the data collection process. Since the filter defines what is considered 'important' for the rest of the fusion process, this adjustment involves reasoning with the current situation description - including threat assessments - to derive a notion of importance. For instance, given the fact that a certain attacker is currently heavily attacking one or more hosts, it might be deemed important to collect and analyze *all* events that originate from this specific attacker. This means events that result from legitimate as well as malicious behavior. Or, given the suspicion that a certain attack is going on, further evidence that supports this suspicion might be sought. Sensor cuing does not only involve broadening the filter, but also shrinking it. Again, this depends on the current situation. If, for instance, an attack that was suspected is now *known* to exist, the search for further evidence can be ceased. Hence, sensor cuing provides the context-sensitivity that is needed for the level 0 filter.

Apart from sensor cuing, Resource Management adjusts the fusion process itself. There are numerous possibilities for such adjustments. Based on the threat assessment, for instance, the level 4 process might prioritize the analysis of attacks with a high threat-level. This prioritization is then communicated back to the various processes within the fusion-based IDS. This results in the allocation of more resources to the attacks or attackers that pose the most threat.

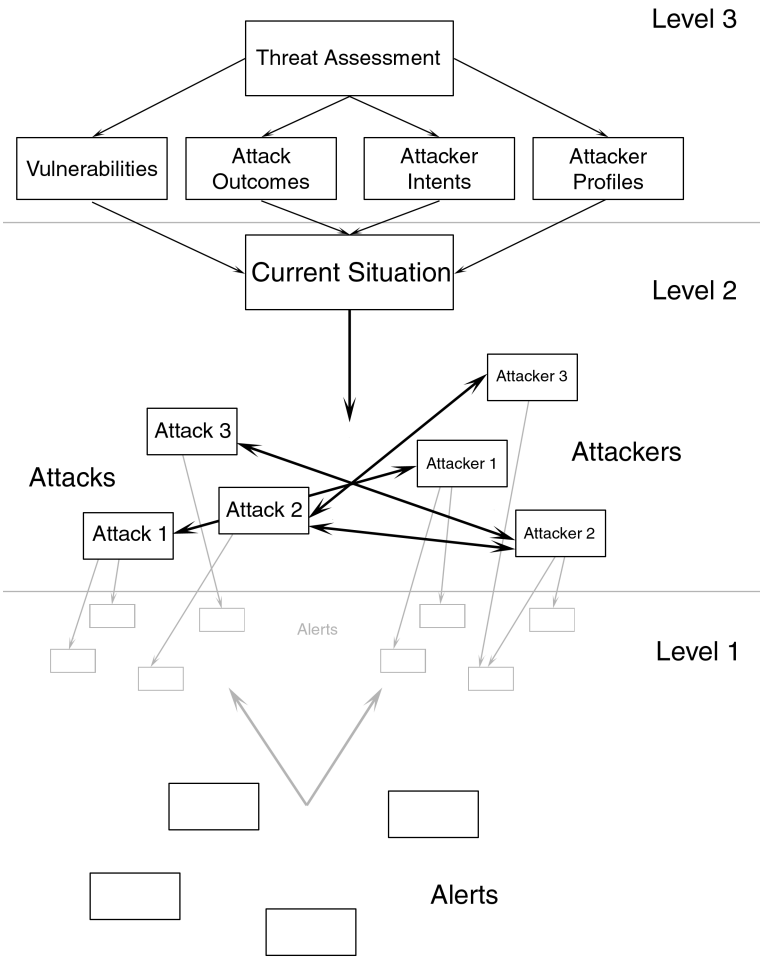


Figure 4.6: Multiple levels of abstraction

4.3 Architecture

In this section, an architecture for fusion-based IDSs is developed. The basis for this architecture is the multiple level-of-abstraction view that was presented in Section 4.2. In other words, the architecture supports the functions described in that section.

The multiple level-of-abstraction situational view that a fusion-based IDS maintains is summarized in Figure 4.6. This figure shows what kind of information is associated with the abstraction levels represented by levels 1, 2, and 3 of the JDL model. It also shows the relationships between the different levels of abstraction. This is all in agreement with Section 4.2. The collection of alerts forms the lowest level of situation description. Analy-

sis reveals two types of objects at a higher abstraction level: attacks and attackers. There is a close relationship between attacks and attackers in that there is always at least one attacker involved in an attack, and there is at least one attack that an attacker is involved in. At yet another level higher, the impact of the current situation is analyzed by taking into account the capabilities and intents of the attackers, and the vulnerabilities of the monitored systems.

The multiple level-of-abstraction view is attained in two different steps:

1. Combination, or fusion, of data originating from multiple sensors (level 1);
2. Further fusion of, and reasoning with, the fused level 1 data (levels 2 and 3).

An overview of the level 1 fusion in a fusion-based IDS, and the level 0 filtering that precedes it, is depicted in Figure 4.7. The figure shows four events, observed by three sensors. Not all sensors see all events, however. Sensor 2 does see all four events, but sensor 1 can see only three. The view of sensor 3 is even more limited; only two events are observed. Each of the sensors has its own filter. Since each of the filters operates independently, events that pass through one filter do not have to pass through another. After the resulting alerts are aligned, they are inspected to find alert tracks, i.e. groups of alerts that are the result of multiple sensors reporting the same event. Based on these alert tracks a refined identity estimate of the events is attained. In this figure, it is assumed that each sensor-filter combination fires exactly one or zero alerts for each observed event. As it turns out, this is a gross oversimplification.

As stated in Section 4.2.1, the sensors of a fusion-based IDS are comparable to the sensors of a traditional IDS. More specific, the two types of observables - events on a host and events on the wire - correspond with what is observed by a HIDS and a NIDS respectively. Furthermore, the filtering function of a fusion-based IDS is comparable with the analyzer component of a non-fusion-based IDS. This means that traditional IDSs can be reused as sensor-filter combinations for a fusion-based IDS (see Figure 4.8). In short, traditional IDSs form single-sensor systems that can be reused as input to a multisensor fusion-based IDS. Input to a fusion-based IDS is not limited to traditional IDSs, however. Information can also be filtered out of log files, such as firewall and router logs, for instance. This requires a separate process that monitors the log file, interprets and aligns the entries and reports events that the corresponding filter allows to pass. This means that multiple heterogeneous sources - different types of IDSs as well as other types of sources - can be used as input to a fusion-based IDS. For clarity, in the rest of this section the sensor-filter combinations are represented only by IDSs.

However, an IDS is not guaranteed to fire one or zero alerts per event. On

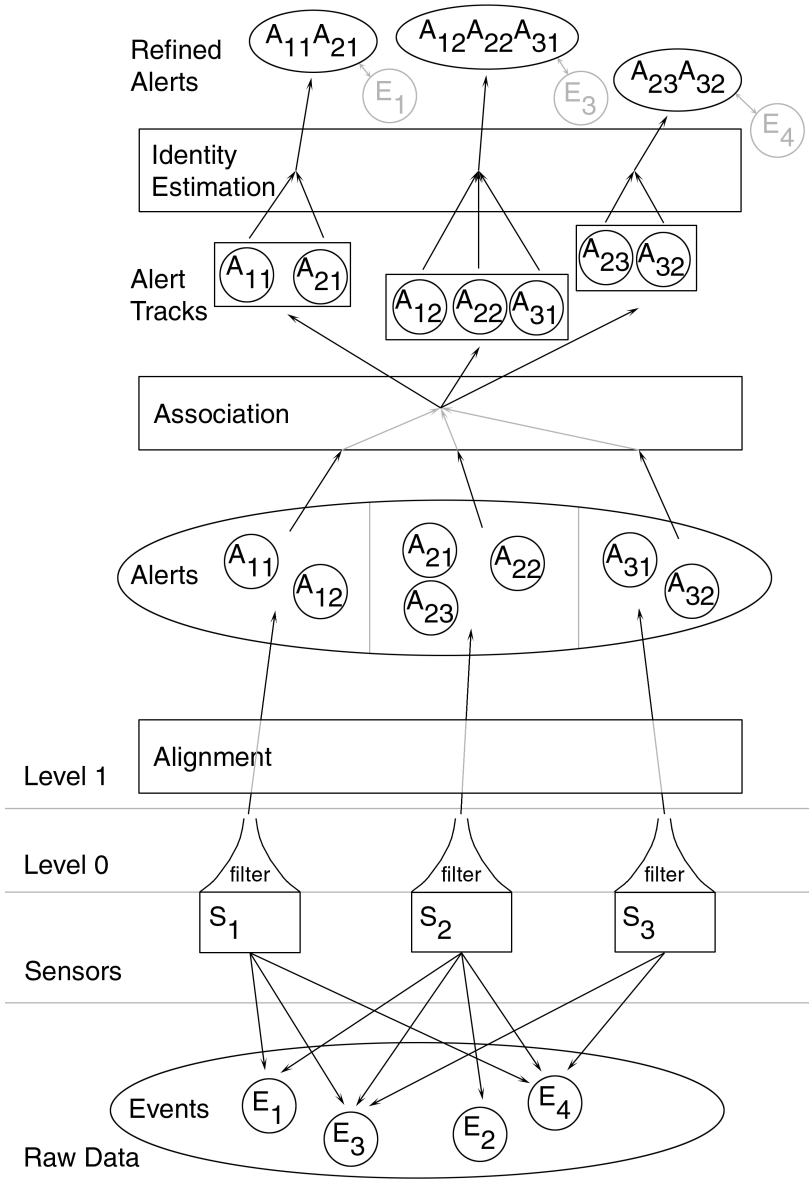


Figure 4.7: Fusion and filtering at levels 0 and 1

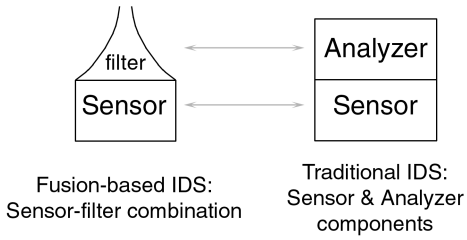


Figure 4.8: Traditional IDS reused as sensor-filter combination

the contrary, it is quite common for an IDS to aggregate the observations of multiple events of a certain type and fire a single alert containing information about all these events. For example, most IDSs will try not to fire an alert for each and every event that is part of a port scan. Instead, an aggregated alert is fired indicating a port scan, and presumably a list of targeted ports. Such an aggregated alert represents a level 2 attack, instead of a level 1 alert. Hence the port scan alert should be directly assigned to the level 2 process, instead of going through the level 1 process first. This identifies the second task at level 0 besides filtering: allocation of data to the appropriate process. While this defies the one-way bottom-up information flow that was presumed in Section 4.2, the functionality that was identified for each level still holds good.

Not only does the reuse of traditional IDSs imply more functionality at level 0, it also affects the fusion process, and hence the fusion architecture, itself. The fusion of data from multiple sensors can no longer be viewed as the one-way bottom-up process depicted in Figure 4.7. Instead, there must be a way to fuse the higher level aggregated data that can be created by an IDS with the lower level alerts, as well as with aggregated data from other IDSs.

Section 4.2.3 remarks that the tracking of attacks and attackers at level 2 resembles the functionality of the level 1 fusion process. This means it also consists of an association stage and a fusion stage[13]. In fact, it can be viewed as level 1 fusion in which the objects are not alerts, but attacks and attackers. The fact that attacks and attackers are level 2 objects rather than level 1 objects, is the reason to place attack and attacker tracking at level 2 instead of level 1. The association stage at level 2 inspects the (refined) alerts that are the output of the level 1 process. Alerts that are associated with the same attack or attacker are combined, and refined estimates of the corresponding attack or attacker emerge. Together with the refined alerts that result from level 1, the aggregated alerts that can be produced by an IDS form the input for this level 2 fusion process.

Section 3.4 presents three possible level 1 fusion architectures:

[13] Alignment already took place at level 1.

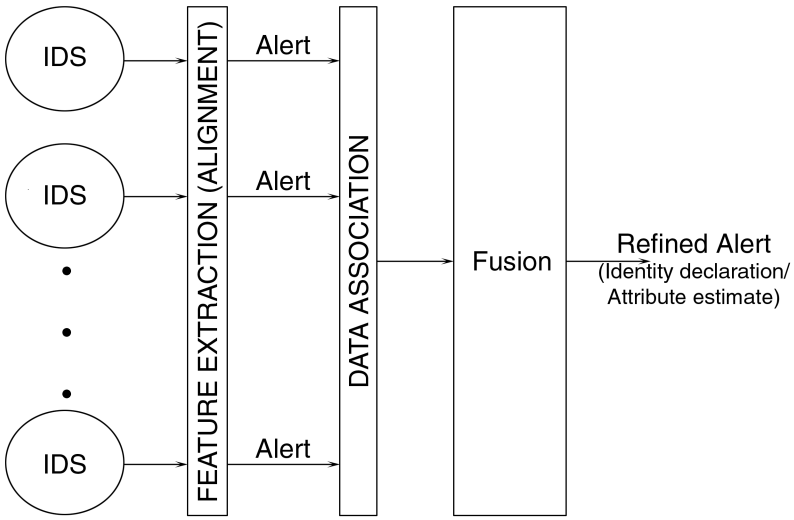


Figure 4.9: A feature level fusion architecture for object refinement

- Data level fusion;
- Feature level fusion;
- Decision level fusion.

In data level fusion, raw data is associated and fused. In feature level fusion, feature vectors are first extracted from the raw data. These feature vectors are then associated and fused. In decision level fusion, each sensor makes a preliminary decision based on its own single-source data. These decisions are then associated and fused.

The alignment stage shown in Figure 4.7 is actually the feature extraction function that belongs to feature level and decision level fusion. As described in Section 4.2.2, the raw data is aligned by extracting relevant features. Noticing this, and comparing Figure 4.7 with Figure 3.6, it becomes clear that Figure 4.7 depicts a feature level fusion architecture. The feature vectors are formed by the alerts that are aligned to a common reference frame. This is also depicted in Figure 4.9.

The next fusion step, to track attackers and attacks, also has characteristics of feature level fusion. Although the refined alerts are regarded as decisions at level 1, they can be viewed as features describing attacks and/or attacker behavior at level 2. Indeed, if all input to this fusion step consisted of these refined alerts, level 2 fusion could be regarded as purely feature level fusion. However, the aggregated alerts that can be output by an IDS also serve as input for this fusion step. Unlike the level 1 output, these aggregated alerts do represent level 2 decisions. If the input to the level 2 fusion process consisted solely of aggregated alerts, level 2 fusion would employ

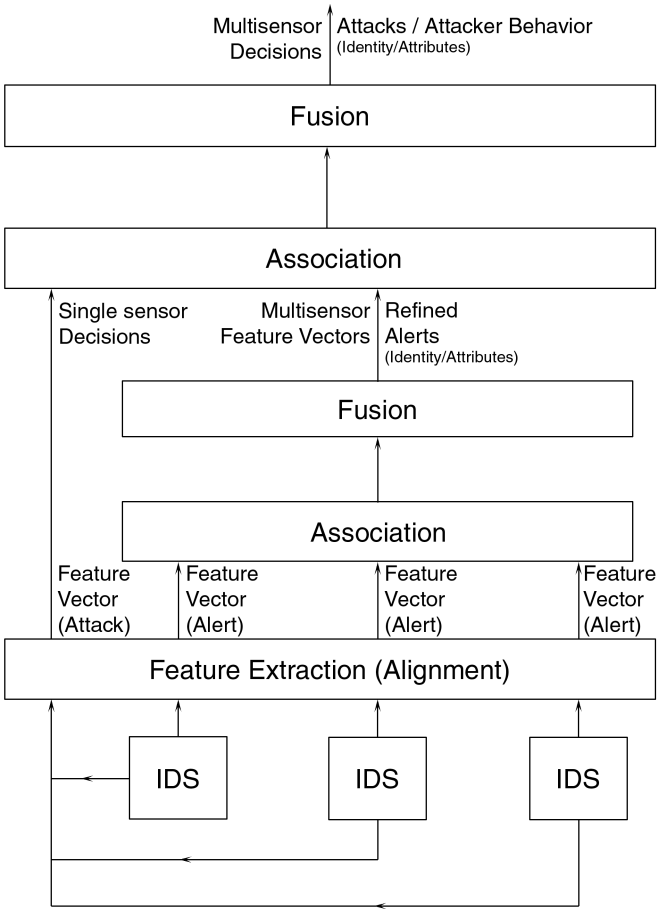


Figure 4.10: A hybrid fusion architecture for object and situation refinement

decision level fusion. Consequently, fusion at level 2 has characteristics of feature level as well as decision level fusion. A hybrid fusion architecture is needed for level 2 fusion. Figure 4.10 shows such a hybrid fusion architecture for object and situation refinement.

Level 2 fusion is much more complicated than level 1 fusion, for it must take the current situation into account. At level 1, comparison of attributes of different events, and calculation of their similarity, is needed for association and consequent fusion. Unlike level 1 fusion, level 2 fusion needs to take the possibility of a false alarm, a false positive, into account. This means that, during the association stage, there is yet another hypothesis to consider. Evidence that supports or refutes the hypothesis that an alert is a false positive must be collected from the refined level 1 alerts. For instance, a refined alert that originated from a single sensor is more likely to be a false positive than an alert that was formed by fusing data from multiple sensors. If not only the number of sensors that reported an event is low, but

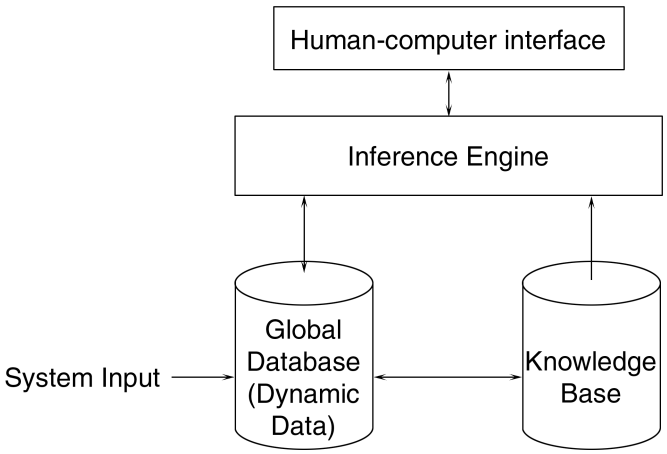


Figure 4.11: Architecture of an expert system (adapted from [Hal92])

there is also no clear sign that it is part of an attack, it might be discarded as a false alarm. Furthermore, as shown in Section 4.2.3, level 2 fusion needs sophisticated context-sensitive reasoning. Consequently, to enable automation of level 2 fusion, it is required that the system is able to reason like a human expert. The use of expert systems accomplishes this.

Hall describes the basic structure of an expert system as comprising four logical parts [Hal92] (see also Figure 4.11):

- A knowledge base, that contains facts, algorithms, and a representation of heuristics;
- A global database that contains dynamic data;
- A control structure or inference engine;
- A human-computer interface.

The knowledge base contains long-term knowledge, the global database medium-term knowledge - or, in other words, the current situation description. The inference engine iteratively decides which knowledge from the knowledge base is applicable to the current situation, and alters, or updates, the current situation description accordingly.

There are various ways to represent knowledge in the knowledge base. Hall presents an overview of these knowledge representation techniques [Hal92]:

- Production rules;
- Networks;

- Frames;
- Scripts;
- Analogical methods.

Production rules are rules of the form IF-THEN. Networks provide a description of hierarchical relationships between generic objects and instances of those objects. Frames are designed to encapsulate multiple attributes of an object, while scripts define sequences of action - and hence incorporate a time element. The direct use of analogical methods is a 'catch-all' category that refers to domain-specific specialized knowledge representations.

All these knowledge representation techniques are in some way or another applicable to fusion-based intrusion detection. If certain conditions are known to imply a false positive, the production rule 'IF (certain conditions are true) THEN (the alert is a false positive)' can be used to identify this type of false positives. Networks are applicable when a distinction must be made between generic classes of attacks, and specific instances thereof. Examples of generic attack classes are port scans and Denial of Service (DoS) attacks. Specific instances of these classes are for instance a RESET scan (port scan), and a smurf attack (DoS)[14]. Frames are used to describe an object, such as an attack. In fact, frames are often used to implement networks [Hal92]. An example of the use of scripts is an attack script. An attack script contains descriptions of sequences of actions that make up known attacks. Hence, together with frames - which contain static stereotypical data - attack scripts aid in the recognition of known attacks. Attack scripts also aid in the 'intent recognition' subprocess of threat assessment.

Uncertainty must also be taken into account. As indicated in Section 3.3 there can be a disparity between the current situation description and reality. Furthermore, there can be inherent uncertainty in the relationships described by the data in the knowledge base. For instance, instead of being able to say 'IF (certain conditions are true) THEN (the alert is a false positive)', one might only be able to conclude that 'IF (certain conditions are true) THEN (the alert is *probably* a false positive)'. The term 'probably' indicates there is uncertainty embedded in the conclusion. There are various techniques that can be used to represent uncertainty. Hall lists the following representation techniques [Hal92]:

- Classical probability $P(H)$ of a hypothesis;
- Evidential interval to indicate support and plausibility of a proposition or a hypothesis;
- Confidence factors that describe a degree of belief or validity;
- Fuzzy sets, i.e. sets that are defined by specifying set elements and a membership function.

[14] For more information on these attacks please see [Nor99].

The question which knowledge and uncertainty representation techniques should be used is a question that must be answered when implementing a specific fusion-based IDS.

Recall that the purpose of a fusion-based IDS is to provide information, rather than to make a decision. Decisions based on this information, for instance the decision to restrict or deny access for a certain (attacker) address by adding a rule to the firewall, must be made by another entity. This decision making process could again be (partly) automated. But it is very likely that, given the far-reaching consequences a decision might have, this decision making will be performed, or at least supervised, by a human.

To make a good decision based on the information provided by a fusion-based IDS, it is important that the information at all levels of the fusion-based IDS's situational view is accessible and understandable. As mentioned in Section 3.3 there can be a disparity between the system's perception of the situation (the situational view), and reality (the factual events). It is important that such disparities can be found, especially when the decision that is to be made has far-reaching consequences[15]. In order to find such disparities, the (human) analyzer must be able to 'drill down' from the high-level situational overview to a low-level view of individual events. Furthermore, the relationship between objects on different levels must be clear to the analyzer. In other words: a fusion-based IDS must be a *transparent* system. This implies that the knowledge in the expert system's knowledge base must be comprehensible by the human expert.

Transparency has another advantage. Since the human expert understands the information at all levels, he can reason with the information himself and act as an expert 'system' within the fusion-based IDS. In this way, a human expert can directly influence the situation description within the fusion-based IDS.

Thus, what is needed is an architecture that lets several experts - automated expert systems as well as human experts - work together on the same problem: refining the situation description, and assessing the threat of this situation. Section 3.4 identifies the so-called 'blackboard architecture' as an architecture that enables this cooperation between multiple experts. When using a blackboard architecture, multiple experts work together on a problem. The current problem state is kept on a central 'blackboard' that is accessible by all experts. In this case, the blackboard holds the current, multiple level-of-abstraction situation description, summarized by Figure 4.6.

Based on the functionality description in Section 4.2, the following needed expert systems can be identified:

- Attack recognizer;
- Attacker tracker;

[15] For instance the decision to shut down a server or to prosecute an attacker.

- Attack outcome predictor;
- Attacker profiler;
- Attacker intent predictor;
- Vulnerability assessor;
- Threat assessor (prioritizer).

The attack recognizer and attacker tracker are both operating at level 2. The other expert systems operate at level 3. Since the attack recognizer and attacker tracker operate at level 2, they must perform the level 2 association and fusion steps depicted in Figure 4.10. To do so, they rely on the knowledge in their knowledge bases. *What* knowledge must be included in the knowledge bases, as well as how it is represented, is a (very important) implementation decision.

Using a blackboard architecture, a fusion-based IDS can be modelled as depicted in Figure 4.12. Initially, the blackboard is empty. At a given moment, one or more of the IDSs start reporting alerts. These alerts are fused by the level 1 fusion process, after which the fusion process writes the refined alerts to the blackboard. Aggregated alerts (single-sensor decisions) are written directly to the blackboard. In the mean time, the expert systems are constantly inspecting the blackboard for any condition that activates them.

The attack recognizer is activated by the existence of alerts on the blackboard. The recognition of attacks consists of finding associated alerts, using sophisticated reasoning techniques, and recognizing known attack patterns in these associated alerts. When an attack is recognized, the attack recognizer updates the blackboard to reflect this. This means that it marks the alerts as belonging to an attack, and describes the corresponding attack. This description is the result of fusion of the data obtained from the associated alerts, as well as with a priori knowledge about the recognized attack.

The recognition of an attack activates the attacker tracker. Some alerts are now known to be part of attacker behavior. Based on this information, the attacker tracker starts another level 2 fusion process. Instead of finding alerts that form an attack, however, it searches for all alerts that originate from the attacker involved in the recognized attack. Again, the blackboard is updated to ensure it contains all the relevant information; alerts are marked as comprising attacker behavior, and an attacker object is created.

These are the first steps to fill the blackboard with information regarding the current situation. The expert systems keep responding to relevant changes in the situation description on the blackboard, thereby constantly updating and refining it. Each expert system knows under what conditions it should respond. The attacker profiler, for instance, can react on the first indication of the existence of an attacker, but also on new attack infor-

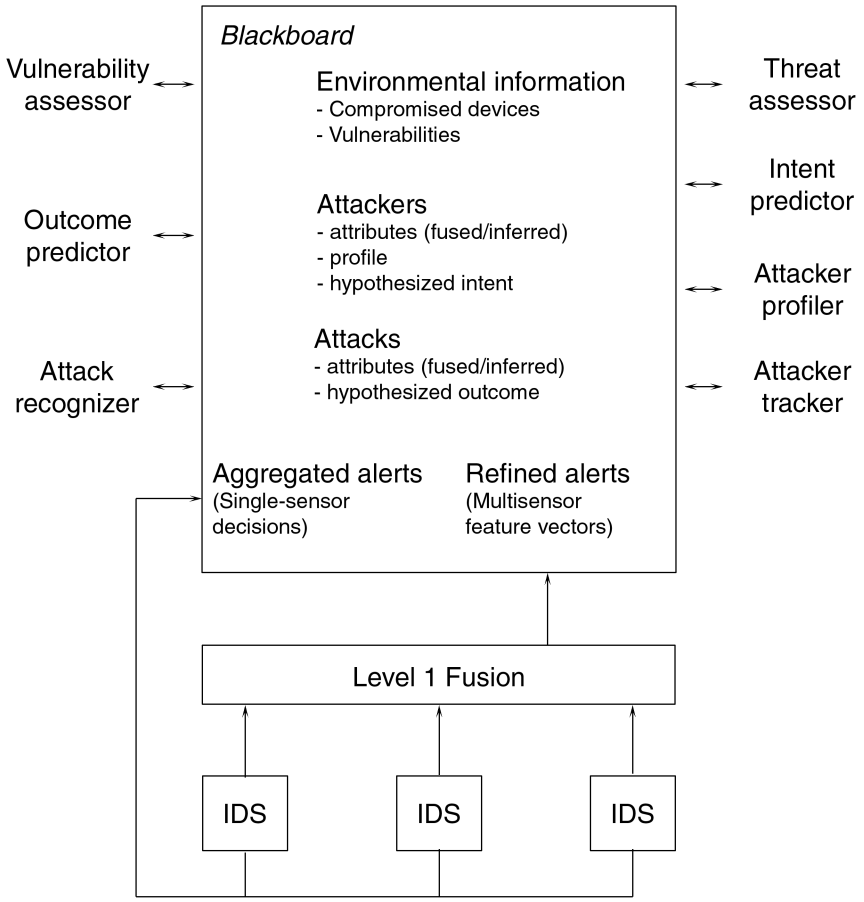


Figure 4.12: Blackboard architecture for a fusion-based IDS

mation regarding that attacker. This new information might provide new insight into the sophistication of the attacker.

The architecture described above reflects a truly opportunistic approach, in which all expert systems operate independently and respond opportunistically to changes on the blackboard. However, as said in Section 3.4, in general a control function is present. This control function mediates between expert systems that want to respond. By adopting a certain selection strategy, the control function can violate the opportunistic character of the system. Such a selection strategy makes it possible to concentrate on specific goals, by enabling the expert systems that directly contribute to achieving this goal to respond. This means that adjusting the control function is part of the level 4 process.

The level 4 process evaluates the current fusion process and dynamically improves it. This means that it influences all levels of the fusion process.

This is depicted in Figure 4.13, in which the control function and the level 4 process are added to the blackboard architecture. In this new situation, whenever an expert system wants to respond it communicates this wish to the control component. Based on its current strategy, and the list of expert systems that want to respond, the control function selects which expert system is allowed to update the blackboard. The current strategy is determined by the level 4 process. It communicates this strategy not only to the control function, but to the other components as well.

The architecture depicted in Figure 4.13 incorporates all functionality described in Section 4.2. It forms a generic architecture for a fusion-based Intrusion Detection System. Due to the multiple levels-of-abstraction, it diminishes the number of alerts that a human analyst using the system has to inspect, by focusing on attacks and attackers instead of individual events. But it also allows the analyst to 'drill down' to the lower level of individual events when needed. Fusing data originating from multiple sensors combined with intelligent reasoning allows the system to ignore false positives. The next section evaluates the architecture by hypothesizing an implementation of a fusion-based IDS and examining how it reacts on a test case.

4.4 Evaluating the Architecture

By now it will be clear that the implementation of a fusion-based IDS based on the architecture from Section 4.3 is not simple. A lot of decisions still have to be made, regarding the fusion techniques that should be used, the kind of reasoning that must be supported, the knowledge that should be put in the knowledge bases, and the representation of uncertainty, to name but a few. Still, the architecture provides a starting point to design and implement a fusion-based IDS.

In this chapter, it is assumed that a fusion-based IDS is implemented according to the generic architecture from Section 4.3. This hypothetical system is used to evaluate the architecture. By using a test case, and showing how the implemented fusion-based IDS would deal with it, it is shown that the architecture indeed does what it is supposed to do:

- support the elimination of false positives;
- reduce the number of positives that must be inspected by the operator of the system.

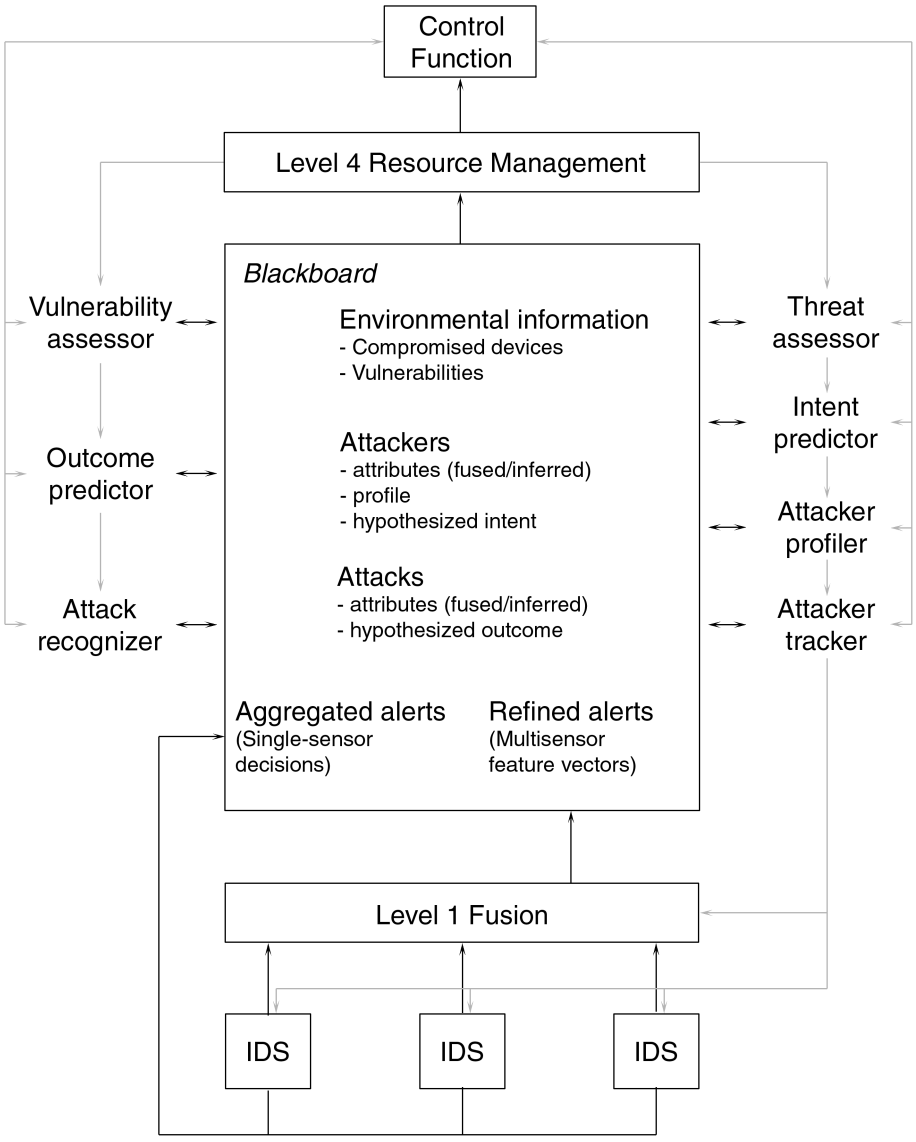


Figure 4.13: Blackboard architecture with level 4 process

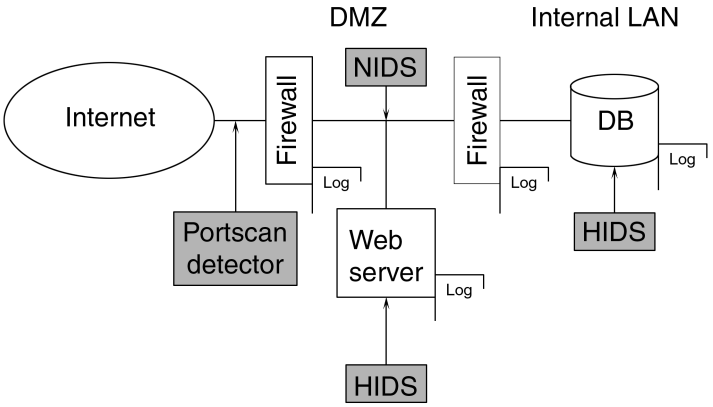


Figure 4.14: Typical network architecture

4.4.1 Scan and Exploit

Often, an attack can be divided into two phases: a 'recon' phase, in which the attacker tries to gain as much information as possible about potential targets, and the actual attack in which vulnerabilities that were found in the recon phase are exploited (see for instance [Nor99]).

A port scan is in general considered part of the recon phase of an attack. The purpose of a port scan is to find out which ports are open on a target system, or on a range of target systems. After a specific port is found to be open, a second scan can be used to search for known vulnerabilities in services that use that port. If, for instance, port 80 is found to be open, one could start to scan for vulnerabilities that are known to exist in various webservers[16]. This results in a list of vulnerabilities that exist, or are likely to exist, on the target. The next step is the actual exploitation of one or more of these vulnerabilities.

This scanning behavior generates a lot of suspicious events, and consequently a lot of alarms are raised by IDSs that observe these events. Figure 4.14 shows a typical network architecture for a web application that is backed by a database (think of an online store, for example). The first firewall limits the traffic that is allowed to enter the DMZ, or De-Militarized Zone. This zone contains the servers that are publicly accessible through the Internet. Since the only server in this zone is a webserver, the only traffic that is allowed from the Internet into the DMZ is traffic destined for port 80. Since the DMZ contains publicly accessible servers, that might be compromised during an attack, a second firewall is placed between the DMZ and the internal LAN. This firewall limits traffic from the DMZ to the LAN to only the traffic that is necessary to access the database server.

[16]The canonical use of port 80 is by a webserver.

There are a number of IDSs placed in the network. Each of the IDSs has its own characteristics. There is a special IDS dedicated to detect port scans[17] placed before the first firewall. Then there is a NIDS observing the DMZ, and two HIDSs; one monitoring the webserver, the other monitoring the database server. Furthermore, numerous devices make use of log files. While the IDSs inspect and report events in 'real-time', the log files provide insight in what happened 'after the fact'. In the example architecture, log files are used by both firewalls, the webserver and the database server.

Port scan

At a given moment, an attacker happens to start such a port scan. Up until that moment, no attacks took place, hence the situation description on the blackboard is still empty. The attack starts with a port scan, but since the firewall blocks all traffic except traffic destined for port 80, the port scan is not seen on the DMZ. The port scan detector, however, is placed before the firewall and therefore does see the port scan. The report of the port scan is comprised of multiple events, and serves as input for level 2 fusion. It is therefore written directly on the blackboard.

This update of the blackboard is a first indication of a change in the situation. It is now the attack recognizer's task to inspect the updated situation to infer whether an attack is taking place. Because no other IDS can see the port scan, there are no subsequent reports of it. Hence there is no further data available for the attack recognizer to use for fusion. However, since the port scan detector's reports tend to be fairly accurate, the attack recognizer infers that indeed there is a port scan going on. It writes its conclusion on the blackboard, after which the attacker tracker marks the source of the port scan as an attacker (after all, the source is now assumed to take part in an attack). There is not yet enough evidence for a clear profile of the attacker. However, the attacker's intentions are likely to be further information gathering and exploitation of a found vulnerability. These hypotheses are written on the blackboard by the intent predictor. Based on the information found sofar, the threat assessor assigns a low threat level to the ongoing attack.

Vulnerability scan

The port scan shows the attacker that port 80 is open. In the next stage of the attack, he uses a script that looks for known vulnerabilities in various webservers. The resulting events are seen by the NIDS that monitors the DMZ. Consequently, it starts reporting these events by generating alerts. Since the HIDS monitors the target webserver, it can also see the events, and thus it too generates alerts. The NIDS's alerts contain different information than the HIDS's alerts, due to the different view both IDSs have.

[17] Even hard to detect stealthy scans that a more generic IDS might miss.

These alerts, that originate from different sources (the HIDS and the NIDS) but can be the result of a single event, are fused at level 1. The first step is alignment of the alerts to a common reference frame, for instance the one presented by the IDMEF model. The alignment can be done by the IDS itself, or by a dedicated process that interprets the IDS's alerts and translates them to the chosen reference frame. Based on common attributes - such as source, target, timestamp, etc. - , associated alerts are identified and assigned to alert tracks. After this, each alert track contains either one alert (from either the HIDS or the NIDS), or two alerts (from the HIDS and the NIDS). The data in the alert tracks are fused, and a refined estimate of the identity of the event is made based on this fused data. Note that in the case of a single alert in the alert track fusion is not possible. In this case, the refined estimate is based only on the single-sensor alert. The IDS, in particular when it uses a misuse detecting analyzer, may have estimated the identity of the event on its own. This single-sensor identity estimation is also used in the refinement step, and - in the case of a single alert in the alert track - might even be reused as the refined estimate.

After level 1 fusion, the blackboard contains lots of refined alerts based on the alerts generated by the HIDS and the NIDS; one for each reported event that was part of the vulnerability scan. The attack recognizer tries to find an attack in the data that are written on the blackboard. One of the things the attack recognizer looks at to find associations in the collection of refined alerts is the source of the alerts (i.e. the attacker, not the IDS that generated it). In this case, the attack recognizer sees that all alerts have the same source, and hence are likely to be associated. Furthermore, the source turns out to be an attacker that is already known, namely the attacker that was involved in the earlier port scan. The blackboard also shows the hypothesized next steps of the earlier port scan: either a scan for vulnerabilities, or a exploitation of a vulnerability. The attack recognizer infers that, due to the high number of alerts - presumably reported by multiple IDSs - and the hypothesis that the port scan would lead to further attacks, the alerts are very likely true positives. Meanwhile, the vulnerability assessor has inspected the alerts. Based on environmental knowledge - amongst others the vendor and version of the webserver software - and a database of known vulnerabilities, it infers that the webserver is not vulnerable to a large part of the reported events. These events try to exploit a vulnerability in another version of the webserver software, or even in software from another vendor. This leads the attack recognizer to believe that this is a vulnerability scan, and not an actual exploit of a vulnerability. The attack recognizer updates the earlier recognized attack to reflect this. At the same time, the attacker tracker updates the track record of the involved attacker to include the newly reported events.

The intent predictor sees one of his hypotheses turn out to be the truth. Based on the new information, it predicts the next step in the attack to be an actual exploitation of a vulnerability, and updates the hypotheses on the blackboard. Its prediction is endorsed by the conclusion of the vulnerability assessor that not all events were the result of actions the webserver is not vulnerable to. This conclusion also leads the outcome predictor to pre-

dict a future successful attack on the webserver. Also, the attacker profiler now has some more clues as to what kind of attacker is involved in this attack: the use of scripts to find vulnerabilities indicates that the attacker is searching for well-known vulnerabilities. It can hence be concluded that the sophistication of the attacker is quite low - his current profile resembles that of a script-kiddy. Based on all this, the threat assessor slightly raises the threat level associated with this attacker and his attack, to indicate a possible succeeding attack by an attacker with a low degree of sophistication.

Exploit

The attacker now knows of the existence of some potential vulnerabilities on the webserver, and decides to pick and exploit one. Again, this results in alerts generated by both the HIDS and the NIDS, and level 1 fusion thereof. The attack recognizer, still looking for attacks, again sees events that result from actions by the known attacker. The attacker tracker updates the attacker's track with the new information. The time between successive events gives an indication of whether the attacker is performing the attack by hand, or by use of a script. This is used by the attacker profiler to further refine the profile. The attack recognizer compares the sequence of events with sequences of events that are known to make up a certain attack, and finds a match. It updates the attack information on the blackboard to indicate which attack is being performed. The outcome predictor, making use of information from the vulnerability assessor, predicts that the outcome of the attack is now a compromised host. It consequently updates the environmental information on the blackboard to reflect this. The threat assessor raises the threat level even more.

4.4.2 False Alarm

In the meantime, a legitimate user uses the webapplication to run a query on the database. The query parameters can be filled in in a search form that is served by the webserver. However, the search parameters provided by the user happen to trigger the NIDS - perhaps some part of the string he used is also found in a known attack. The reported alert is also put in to the system. First, level 1 fusion is applied, however - since the HIDSs did not react - nothing is done in this step: the alert attributes and identity estimate provided by the NIDS are copied onto the blackboard. The attack recognizer notices the alert, and tries to find associated alerts. This fails, because there are none. All this diminishes the believe of the attack recognizer that this event is truly part of an attack. Even if it is part of an attack, presumably more events are needed for the attack to complete. In the end, the attack recognizer decides not to report the event as being part of an attack. It rightly infers this must be a false positive.

4.4.3 Attack on the Application Database

Another attacker decides to try his luck. Unlike the other attacker, this attacker tries to attack the application instead of the webserver. He does so by trying a so-called 'SQL Injection' attack. In such an attack, the attacker manipulates a query string so that it contains characters that are normally used in SQL[18]. If these characters are not filtered out, they can be interpreted as part of the SQL query. This means that the attacker can perform any query he wants, even queries that he is not allowed to perform when legitimately using the application. Such an SQL Injection attack must in general be 'hand-crafted' for the targeted server.

When the attacker starts its SQL Injection attack, again multiple IDSs start generating alerts - this time the NIDS and the HIDS on the database. Level 1 fusion is applied to the alerts and thereafter the attack is recognized by the attack recognizer. This leads amongst others to the identification of a new attacker. Due to the nature of the attack - it is not likely that it is a scripted attack - the attacker profiler indicates that the attacker is probably more sophisticated than the earlier attacker. The threat assessor uses this information to set a preliminary threat level for this attack. The outcome predictor, however, has no clue as to whether the attack succeeded or failed. To find out, it needs more information - for instance from the database log - about the result of the actions comprising the attack. It communicates this need for more information, for instance by writing two hypotheses on the blackboard (attack succeeded and attack failed) that are dependent on the needed information. The level 4 process acts on this by requesting the needed information from the log - provided there is a process able to parse the log that the level 4 process can request the information from. Once the new information is written to the blackboard, it is used by the outcome predictor to update the hypotheses correspondingly.

The example in this section shows how the generic architecture aids in solving the problems with false positives and the low-level view of traditional IDSs. However, the example uses a fairly simple test case. More sophisticated attacks exist, for which the benefit of fusion-based intrusion detection might be even higher. Think, for instance, of the 'Mitnick attack', in which two machines having a trust relationship are attacked, and the attacker is able to hijack the session (see also [Nor99]). Also, the used network architecture is a very simple one. Much more intricate architectures exist and are used. Although this example is a first evaluation step, further evaluation of the architecture is still needed (see also Section 5.3).

[18] SQL (Structured Query Language) is a standard for accessing a database. It can be used to retrieve and manipulate (insert, update, delete) data in a database.

4.5 Summary

The purpose of a fusion-based IDS is to provide information concerning the current situation, i.e. attacks and attackers and their relationships. It provides information of greater quality than traditional IDSs by using multiple sensors and a multiple level-of abstraction situational view.

The multiple levels of abstraction are supported by the JDL data fusion model. This means that the JDL model is applicable to intrusion detection. At each of the levels of the JDL model, functionality of a fusion-based IDS can be identified. At level 0, events (that are the result of actions) are observed, and unimportant events are filtered out. The important events that are reported are called alerts. These alerts are the subject of level 1 fusion. Level 1 fusion finds groups of alerts, called alert tracks, that are the result of the same event. Data of the alerts in an alert track are fused and a refined identity estimate is made. During level 2 fusion, context-sensitive reasoning is applied to find and recognize attacks and attackers. The next level - level 3 - assesses the threat of the current situation, and of individual attackers and attacks. Level 4, finally, evaluates the total fusion process and adjusts it to dynamically improve the quality of the output.

An architecture for a fusion-based IDS must support the multiple level-of-abstraction situational view that is needed. The sensor-filter combinations that provide input to the fusion-based IDS can be formed by - but are not limited to - traditional IDSs. Reuse of traditional IDSs as sensor-filter combinations implies a hybrid fusion architecture is needed for level 2 fusion, because multisensor feature vectors must be fused with single sensor decisions. The reasoning that is needed to take the context into account can be automated by using expert systems. The following 7 needed expert systems can be identified:

- Attack recognizer;
- Attacker tracker;
- Attack outcome predictor;
- Attacker profiler;
- Attacker intent predictor;
- Vulnerability assessor;
- Threat assessor (prioritizer).

These expert systems can work together by using a blackboard architecture.

Evaluation of the developed architecture by using a test case on a hypothetical implementation shows that the architecture does indeed aid in eliminating false positives and reducing the number of positives that must be inspected by the operator of the system.

Chapter 5

Conclusions and Future Research

This chapter revisits the goal of this thesis and states the conclusions. Furthermore, it provides a summary of the major contributions of this thesis. Finally, possible directions for further research are given.

5.1 Conclusions

The goal of this thesis, as stated earlier, is the development of an architecture for a fusion-based Intrusion Detection System that aids in:

1. the elimination of false positives, and
2. the reduction of the number of positives that must be inspected by the operator of the system.

Chapter 4 shows why application of the multisensor data fusion framework can help to solve these problems, through the use of multiple heterogeneous sensors and obtaining a multiple level-of-abstraction view. It also shows an architecture for a fusion-based IDS, and a test case that evaluates the architecture. Although the test case that is used for evaluation is not extremely sophisticated, it does show the potential of the architecture to contribute to the solution of the identified problems. One thing the architecture shows is the importance of the common reference frame that is chosen for alignment; it is at the very basis of fusion-based Intrusion Detection. In summary, the conclusions of this thesis are:

- The multisensor data fusion framework appears to be applicable to intrusion detection;
- The architecture that is developed in Chapter 4 seems promising in

its ability to aid in the elimination of false positives and the reduction of the overall number of positives.

- A blackboard architecture can be used to let multiple parts of a fusion-based IDS work together;
- The quality of the common reference frame that is used for alignment is critical for the functioning of a fusion-based IDS.

5.2 Summary of Contributions

In summary, the major contributions of this thesis are:

- The development of a generic architecture for fusion-based Intrusion Detection Systems;
- Further evaluation of multisensor data fusion as a framework for intrusion detection;
- Providing a starting point for development of fusion-based IDSs.

5.3 Future Research

This thesis still leaves many questions open regarding fusion-based IDSs and the implementation of the developed generic architecture. The following topics could be topics of future research:

- The design of a specific fusion-based IDS based on the generic architecture;
- The selection and development of fusion algorithms for fusion-based intrusion detection. This could benefit from the taxonomies that have been developed by Antony [Ant95];
- The design and implementation of individual expert systems. This involves answering such questions as what knowledge representation techniques to use as well as what knowledge to include in the knowledge base;
- Building and evaluating a prototype implementation;
- Further refinement of the architecture based on experiments with a prototype;
- Identification and incorporation of further requirements into the architecture. Possible requirements are:

- (Near) real-time analysis;
- Secure communication between the various parts of a fusion-based IDS.

Appendix A

Base-Rate Fallacy

This Appendix contains a description of the base-rate fallacy problem by example. This slightly modified example is taken from Axelsson [Axe99].

The base-rate fallacy is a direct result from Bayes' theorem

$$P(A|B) = \frac{P(A) \cdot P(B|A)}{P(B)}$$

which can be rewritten to

$$P(A|B) = \frac{P(A) \cdot P(B|A)}{\sum_{i=1}^{n} P(A_i) \cdot P(B|A_i)}$$

by expanding the probability $P(B)$ for the set of all n possible, mutually exclusive outcomes A.

Now we let I denote intrusive behavior, $\neg I$ non-intrusive behavior, A the presence of an intrusion alarm and $\neg A$ the absence of an intrusion alarm. We can then denote the true positive rate, or *detection rate*, by $P(A|I)$, the true negative rate by $P(\neg A|\neg I)$, the false positive rate, or *false alarm rate*, by $P(A|\neg I)$ and the false negative rate by $P(\neg A|I)$.

We want both $P(I|A)$ and $P(\neg I|\neg A)$ to be as high as possible, for that means that an alarm really indicates an intrusion and the absence of an alarm really indicates the absence of an intrusion. With the use of Bayes' theorem we know that

$$P(I|A) = \frac{P(I) \cdot P(A|I)}{P(I) \cdot P(A|I) + P(\neg I) \cdot P(A|\neg I)}$$

and

$$P(\neg I|\neg A) = \frac{P(\neg I) \cdot P(\neg A|\neg I)}{P(\neg I) \cdot P(\neg A|\neg I) + P(I) \cdot P(\neg A|I)}$$

The incidence rate $P(I)$ is the probability that a single, randomly selected event is part of an intrusion. Note that an event is not the same as an alarm, but an alarm might be raised based on a (number of) event(s).

$$P(I) = \frac{\#events/intrusion \cdot \#intrusions/day}{\#events/day}$$

The number of events per day will be usually very big, especially when related to the number of intrusions per day. Even when multiplied by the average number of events that an intrusion consists of, $P(I)$ will still be very small. Since $P(\neg I)$ is defined in terms of $P(I)$ as

$$P(\neg I) = 1 - P(I)$$

we know that $P(\neg I)$ will be very close to 1.

Now remember that

$$P(I|A) = \frac{P(I) \cdot P(A|I)}{P(I) \cdot P(A|I) + P(\neg I) \cdot P(A|\neg I)}$$

Since $P(A|I)$ is the detection rate and $P(A|\neg I)$ is the false alarm rate, we can rewrite this as

$$P(I|A) = \frac{P(I) \cdot \textit{detection rate}}{P(I) \cdot \textit{detection rate} + P(\neg I) \cdot \textit{false alarm rate}}$$

Since $P(\neg I) = 1 - P(I)$ is much larger than $P(I)$ for small values of $P(I)$, the influence of the false alarm rate on $P(A|I)$ is very big. It completely overwhelms the influence that the detection rate has. Thus even a small false alarm rate will substantially lower $P(I|A)$ while a very large detection rate will have relatively little influence.

For a numeric example please refer to [Axe99].

Bibliography

[A+00] Julia Allen et al. State of the Practice of Intrusion Detection Technologies. Technical report, Carnegie Mellon Software Engineering Institute, January 2000.

[Ant95] Richard T. Antony. *Principles of Data Fusion Automation*. Artech House, Inc., 1995.

[Axe99] Stefan Axelsson. *On a Difficulty of Intrusion Detection*. Department of Computer Engineering, Chalmers University of Technology (Göteborg), August 1999.

[Bas00] Tim Bass. Intrusion Detection Systems and Multisensor Data Fusion. *Communications of the ACM*, 43(4):99–105, April 2000.

[BFGW98] Tim Bass, Alfredo Freyre, David Gruber, and Glenn Watt. E-Mail Bombs and Countermeasures: Cyber Attacks on Availability and Brand Integrity. *IEEE Network*, pages 10–17, March/April 1998.

[BSI00] British Standards Institution. *BS ISO/IEC 17799:2000 (BS 7799-1:2000) Information technology - Code of practice for information security management*, 2000.

[BWC02] Daniel J. Burroughs, Linda F. Wilson, and George V. Cybenko. Analysis of Distributed Intrusion Detection Systems Using Bayesian Methods, 2002.

[Col] Federal Agent Byron S. Collie. *Intrusion Investigation and Post-Intrusion Computer Forensic Analysis*. Australian Federal Police, Directorate of Information Warfare, Headquarters Air Command, Royal Australian Air Force.

[Cup01] Frédéric Cuppens. Managing Alerts in a Multi-Intrusion Detection Environment, 2001.

[Den87] Dorothy E. Denning. An Intrusion-Detection Model. *IEEE Transactions on Software Engineering*, SE-13(2):222–232, February 1987.

[EF94] K. Egevang and P. Francis. The IP Network Adress Translator, May 1994.

[Hal92] David L. Hall. *Mathematical Techniques in Multisensor Data Fusion*. Artech House, Inc., 1992.

[HL97] David L. Hall and James Llinas. An Introduction to Multisensor Data Fusion. In *Proceedings of the IEEE*, volume 85, January 1997.

[IDW] Intrusion Detection Working Group. *http://www.ietf.org/html.charters/idwg-charter.html*.

[ISM01] Information Security Magazine, August 2001. *http://www.infosecuritymag.com/archives2001.shtml#august2001*.

[Kot02] Pravin Kothari. Intrusion Detection Interoperability and Standardization, February 2002. *http://rr.sans.org/intrusion/interop.php*.

[Lam99] Dale A Lambert. Assessing situations. In Robin Evans, Lang White, Daniel McMichael, and Len Sciacca, editors, *Proceedings of Information Decision and Control 99*, pages 503–508, Adelaide, Australia, February 1999. Institute of Electrical and Electronic Engineers, Inc.

[M+01] Ludovic Mé et al. La détection d'intrusions: les outils doivent coopérer. *Revue de l'Electricité et de l'Electronique*, pages 50–55, May 2001.

[MCA00] John McHugh, Alan Christie, and Julia Allen. Defending Yourself: The Role of Intrusion Detection Systems. *IEEE Software*, pages 42–51, September/October 2000.

[Nii86] H. Penny Nii. Blackboard systems: The blackboard model of problem solving and the evolution of blackboard architectures. *The AI Magazine*, VII(2):38–53, Summer 1986.

[Nor99] Stephen Northcutt. *Network Intrusion Detection - An Analyst's Handbook*. New Riders Publishing, 1999.

[Sch00] Bruce Schneier. *Secrets and Lies - Digital Security in a Networked World*. John Wiley & Sons, Inc., 2000.

[Sem99] Tzvetan Semerdjiev. Information Security and Multisensor Data Processing. *Information & Security*, 2, 1999.

[Sno] Snort. *http://www.snort.org/*.

[Sta99] William Stallings. *Network Security Essentials - Applications and Standards*. Prentice-Hall, Inc., 1999.

[Ubi00] Ubizen. *HIZEN - High Level Simplified Risk Management Methodology*, 2000. Internal document.

[VS01] Alfonso Valdes and Keith Skinner. Probabilistic Alert Correlation. In *Recent Advances in Intrusion Detection (RAID 2001)*, number 2212 in Lecture Notes in Computer Science. Springer-Verlag, 2001.

[Wal99] L. Wald. Definitions and Terms of Reference in Data Fusion. *International Archives of Photogrammetry and Remote Sensing*, 32, June 1999.

[WL90] Edward Waltz and James Llinas. *Multisensor Data Fusion*. Artech House, Inc., 1990.

[Yan99] Ronald M. Yannone. Exploring Architectures and Algorithms for the 5 JDL/DFS Levels of Fusion Required for Advanced Fighter Aircraft for the 21st Century, May 1999. Web-accessible abstract at *http://65.105.56.185/master23/category159/A276193.html*.

[Ype01] F. Ypey. Hackonomics & Intrusion Management - Managing Corporate Infosec Risk with Intrusion Detection: A Study of Applying IDS in practice. Master thesis, Erasmus University Rotterdam, 2001.