Intrusion Detection: Introduction to Intrusion Detection and Security Information Management

Hervé Debar and Jouni Viinikka

France Télécom Division R&D,
42 rue des Coutures,
F-14066 Caen Cedex 4
{herve.debar, jouni.viinikka}@francetelecom.com

Abstract. This paper covers intrusion detection and security information management technologies. It presents a primer on intrusion detection, focusing on data sources and analysis techniques. Data sources presented therein are classified according to the capture mechanism and we include an evaluation of the accuracy of these data sources. Analysis techniques are classified into misuse detection, using the explicit body of knowledge about security attacks to generate alerts, and anomaly detection, where the safe or normal operation of the monitored information system is described and alerts generated for anything that does not belong to that model. It then describes security information management and alert correlation technologies that are in use today. We particularly describe statistical modeling of alert flows and explicit correlation between alert information and vulnerability assessment information.

1 Introduction

Information systems security has been a research area for a long time. Initial viruses and worms propagated slowly through the exchange of magnetic containers. With the development of TCP/IP, security problems have become more frequent and taken very different forms, and have lead to the development of new security techniques. Very early in the development of the Internet, vulnerabilities affecting operating systems have allowed attackers to move from system to system. Detecting attackers has been a necessity for military environments. Insufficient access control measures have led to the development of intrusion-detection systems (IDS).

These IDS have been developed to detect abnormal behaviour of information systems and networks, indicating a breach of the security policy. Two families of techniques have been developed, *misuse-detection* and *anomaly-detection*, to analyze a *data stream* representing the activity of the monitored information system. Misuse-based analysis detects known violations of the security policy, explicitly specified by the security officer. Anomaly-based analysis detects deviations from the normal behaviour of the monitored information system.

The objective of intrusion-detection systems today is to inform operators on the security health of information system. This mostly improves accountability but does not protect the information system from attacks. The development of dependable analysis techniques, particularly reduction of false alarms and identification of attack context, should in the future enable the migration to efficient intrusion protection systems, merging access control at the network layer with access control at the application layer. Distributing IDS components onto single workstations should create an efficient multi-layered approach to information security in the near future.

Beyond intrusion detection, security information management (SIM) platforms have emerged to manage alerts created by intrusion detection / prevention systems and other security tools, and provide a global view of the security state of the information system. These platforms are a must-have for large organization concerned with the security of their information systems, and are offering facilities for alert correlation, display and threat management.

In this paper, we will first cover intrusion detection by examining sensors, data sources and analysis techniques. We will then present security information management and alert correlation techniques.

2 Intrusion Detection Sensors

Figure 1 presents a schematic model of an intrusion detection / intrusion prevention system (IDS/IPS) according to the Intrusion Detection Working Group[1] (IDWG) of the Internet Engineering Task Force (IETF). This model contains all the important components of an intrusion detection system. The square boxes represent software or hardware components, while the ellipses represent human roles.

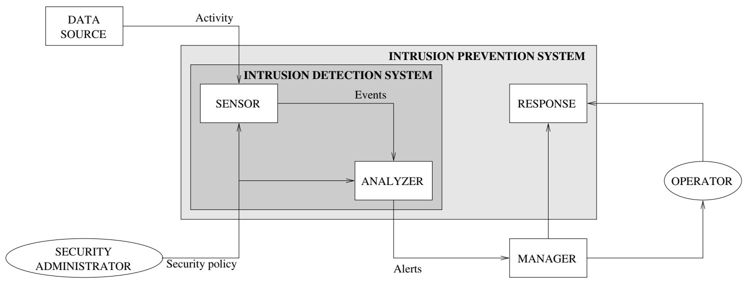

Fig. 1. Schematic model of an intrusion detection / intrusion prevention system according to the Intrusion Detection Working Group

An intrusion detection system observes the activity of the monitored information system through a data source. These data sources are captured and synthesized as events by the *SENSOR* component of the intrusion detection system. While nothing prevents an intrusion detection system to incorporate several

[1] `http://www.ietf.org/html.charters/idwg-charter.html`

sensors simultaneously and feed heterogeneous events to the *ANALYZER* component, current commercial intrusion detection systems acquire and analyze a single data source. Data sources are detailed in section 3.

Events are processed by an *ANALYZER*. This analyzer decides whether the event is malicious and should trigger the generation of an alert, or whether the event participates in the normal activity of the monitored information system. His decision depends both on the analysis technique and algorithms (see section 4) and on the configuration decided by the security administrator. The security administrator decides what conditions trigger alerts, what information is sent to the manager and what the appropriate response is. When alerts are triggered, the analyzer sends them to the manager. Alerts are described as Intrusion Detection Message Format[1] XML messages, transported over the IDXP[2] protocol.

When the *MANAGER* receives an alert, it displays its content to an operator. It can also decide to respond to the alert by applying countermeasures on one or several response components.

When an operator receives an alert, he is responsible for processing the alert according to the guidelines set forth by the security policy.

The main difference between IDS and IPS lies in the response capability. An IPS is often positioned inline, separating two networks like an application-level firewall. Now, for each packet or session that triggers an alert, the IPS can decide that the session needs to be terminated, the packet dropped or rejected, or possibly other measures. The response capability existed in IDS systems, in the sense that they were able to reconfigure other components such as firewalls to achieve a block, or to emit TCP RST packets to cut a connection. However, these responses depended on external devices or on luck, and they were often inefficient, mostly because the response was too slow and the attack had already propagated.

3 Data Sources

The first differentiator of intrusion detection technologies is the data source. Each data source requires specific processing to obtain event information, and allows the detection of different attacks.

3.1 Network Data Sources

IDS using network data are called Network Intrusion Detection Systems (NIDS). These NIDS observe the network, either on a hub, the SPAN port of a switch, or taps, collect the packets, and reconstruct an image of the activity of the users. NIDS are probably the most deployed systems today, and IPS are mostly NIDS. IPS have also brought in this area dedicated hardware devices capable of analyzing several gigabits of traffic inline, what previous software-based IDS were not able to do. There has been a large body of research related to optimizing NIDS such as the open source snort[3], but the current trend is clearly towards hardware-based or hardware-assisted solutions.

The network is a very attractive information source because it introduces little disturbance on the monitored information system[4]. SPAN ports on switches may imply bandwidth limitations on the observed traffic, but the stability problems that plagued early SPAN implementations are mostly gone. Another good solution is the tap, with a fail-open mode that guarantees service continuity even in the case of hardware failure. Finally, network operators are generally willing to look at the kind of traffic that crosses their wire, and the natural aggregation of traffic allows the monitoring of large IT infrastructures with a few well positioned boxes.

However, obtaining meaningful events from network observation is not very easy. One needs to decompose the traffic according to the various protocol layers. Also, experience has shown [5] that NIDS may suffer evasion, i.e. the attacker may be able to inject traffic in the network that will either render the attack invisible to the intrusion detection system or make it generate false alarms. Examples of these issues are:

Address masking. Network Address Translation (NAT) does not provide the IP address of the endpoint to the NIDS.

Encryption. Encrypted protocols defeat NIDS because most of their detection process requires access to headers (sometimes encrypted) and to payload (always encrypted). Popular protocols such as SSL and SSH render NIDS unusable.

Fragmentation. IP fragmentation is a rare phenomenon normally. It can be used to fragment the payload in such a way that the NIDS will not see the attack in one piece.

Reliability of the source address. Many attacks can be realized while using fictitious IP addresses (*IP spoofing*[5]) when no answer from the target is necessary. An attacker can also use *stepping stones*[6] to mask the origin of the attack. Finally, it can also use *reflectors*[7] to trick unwilling agents into carrying out the attack on its behalf.

Transience of address information. In many organizations, and ISPs in particular, IP addresses are handed out dynamically via DHCP. Identification of a particular customer through the IP address requires log analysis and can therefore be costly, or even impossible if logs have been rolled over.

The interested reader will find more detailed information in [5,8]. While these issues are not new and are fairly well known at the network layer, evasion techniques keep appearing at the application layer. Application protocols encode the information in the packet payload. Hence, exploitation of the payload requires that the IDS recognizes the possible encodings. The attacker can hide its actions through the use of specific encodings that are not understood by the NIDS.

For example, the HTTP protocol allows the replacement of any character by its ASCII hexadecimal code, prefixed with the % character [9]. This replacement is mandatory for special characters, obviously % but also space, and others. Ptacek and Newsham have shown that NIDS in 1998 did not understand this encoding and could not detect encoded attacks [5]. This particular problem has been solved for a long time, but others regularly appear.

Also, NIDS can include more intelligence than simple decoding, and incorporate recognition of protocol states, referred to as *protocol awareness.* A protocol aware NIDS recomposes the target protocol finite state machine and applies the detection algorithm to only the appropriate state. A NIDS that does not understand protocol states (often referred to as *network grep*) applies its detection algorithm to the payload of the packet regardless of the protocol state or history, and is much more likely to create false positives. Of course, keeping protocol states is costly, and these NIDS include safeguard mechanisms to avoid memory saturation, falling back to stateless detection if needed. '

3.2 System Data Sources

The system audit trails cover all the data sources that are made available by the operating system. The intrusion detection systems using host data sources are named Host Intrusion Detection Systems (HIDS). Syslog on UNIX systems and the NT event log under Microsoft windows operating systems provide to applications a service for identifying, time-stamping and storing information. Using this kind of facility is very easy for application developers, and they are used by several HIDS as a tool to collect, correlate and present system-related information.

Several operating systems also offer a so-called "C2 audit", to conform to the US government requirement for computer purchases. Such an audit aims at providing a trace of all privileged operations realized on a given computer, usually through the recording of system calls. It offers a strong user identification capability and an extremely fine-grained action description. Unfortunately, this C2 audit system is rarely properly documented and has a strong performance impact, so it has been abandoned by most HIDS.

Recent HIDS have developed specific interception software, similar to antivirus technology. This interception software allows the HIDS to recover only the information that can be analyzed, at lesser cost. Of course, these interception mechanisms are operating-system dependant, which results in a smaller number of offerings in the product space.

The main advantage of host audit data is the precision of the information. On this basis, an HIDS is able to reduce the number of false positives, while providing detailed information about the circumstances of the attack. In particular, actors (both target and perpetrator) are correctly and precisely identified. As such, the counter-measures can be appropriately tailored to the situation. Contextual information, related to the success of the attacker's activity, allows the operator to evaluate the risk and determine the appropriate level of counter-measures.

The main drawback of system audit data is that the HIDS has to reside on the same host, or a large volume of data has to be transported for remote analysis. Performance is such degraded through consumption of either bandwidth or processing power for security. Moreover, the behaviour of HIDS under stress heavily relies on the capabilities of the underlying operating system, and there is a real risk that denial-of-service attacks will either incapacitate the HIDS (if the original application has priority) or be facilitated (if the HIDS has priority).

Moreover, an application needs to be installed on the host. This has a strong impact on server deployment when servers have to be qualified before being placed in a production environment. Also, the signature updates may be problematic if software updates are also included in the signature updates.

3.3 Application Data Sources

Application logs cover all the traces maintained by the applications. The intrusion detection systems using application logs are considered as HIDS, even though an application log could provide information about a distributed environment, spanning multiple machines.

A typical example of application logs is the HTTP server log files storing requests presented to the server (usually in the `access.log` file) and error messages (usually stored in the `error.log` file. Each line stores the request presented to the server, and statistics about the response. The format of these log lines is reasonably easy to parse, making the data source an attractive proposition for developers.

Retrieval of the information generally consists of watching the file and parsing additional information into the data structures of the HIDS. Since these logs may ignore local information, constant information such as host names may be added on the fly to obtain an autonomous message.

Application logs are often more precise and dense than both system audits and network traffic, because they contain information that is atomic from the point of view of the application, while multiple packets or several thousand system calls may be necessary to realize the function.

Also, they provide more accurate information with the inclusion of return codes and error messages. These return and error messages are extremely important for the intrusion detection system, because they provide effective diagnostic of the issue and its impact on the monitored information system.

As already mentioned with network traffic, applications may use specific encodings. Depending on the log, decoding may also be required to normalize the information.

The biggest issue with application logs is that they are often targeting debugging and abnormal termination cases. As such, these files may not contain enough data for the HIDS. In certain cases, it is necessary to collect the entire transaction log, because even error-free transactions may contain attack-related activity that needs to be analyzed.

4 Analysis Techniques

Misuse detection takes advantage of the body of knowledge related to security vulnerabilities and penetration of information systems and networks. The IDS contains information about these vulnerabilities and looks for attempts to take advantage of them. When such an attempt is detected, an alert is sent to the management console. In other words, any action that is not explicitly identified as an inappropriate usage of the information system is considered acceptable.

Note that misuse detection does cover more than known attacks and vulnerabilities. If a security policy explicitly bans certain activities, these activities can be linked to alerts in an IDS. The best example would be banning IRC activity from a network. Any connection using the IRC port would trigger an alert. SNMP is also banned from certain environments and its inappropriate usage by network management tools can easily be detected. Also, recurrent attack mechanisms have been analyzed to abstract generic attack methods, covering not a single vulnerability but a class of them. These abstract models allow detection of broad attack patterns covering even some unknown vulnerabilities, or at least ensure that the detection mechanism does not rely on specificities of some attack tools.

Signature Description. In a misuse-based approach, one needs to define the trigger that, when found in the event stream, will generate an alert. This trigger is usually referred to as an attack *signature*, although the terms *scenario* and *rule* have been used to describe these triggers as well. The term *signature* will be used throughout this paper.

Initially, trigger description in IDES [10] took the form of facts entered into an expert system. User actions abstracted from the event stream were also represented as facts, while the detection process was described as production rules. This procedure was extremely costly, because of the processing needed to abstract several low level audit events into a single user action. Snapp and Smaha found that instead of abstracting system or network events to the expert system, it was easier to express vulnerabilities as sequences of events found in the event stream, named signatures [11].

A signature is the expression of some sequence of events characterizing the exploitation of a vulnerability. The detection process is thus simplified, and the cost is transferred to the definition and test of the signature for all the possible event stream formats that the IDS intends to support. This is sometimes a costly trade-off, if the event stream does not contain all the data that is being looked for, or if multiple encodings have to be taken into account.

In practice, a signature is expressed by a sequence of bytes being matched in the event stream [12], or in more complex cases by regular expressions [13]. These expressions are easier to write than the initial sequences of bytes proposed by [11], but it is still somewhat difficult to implement. Difficult because even for the same event stream format and the same exploited vulnerability, attacks can show under very different forms: attackers can mask their attempt under specific encodings or change sequencing by introducing irrelevant events in the data stream. Applying signatures to the data stream requires the sensor to remove protocol-specific encodings or operating-system related dependencies. This phase can be complex and costly performance-wise.

Misuse Detection and False Positives. The misuse detection approach should be able to generate very few false positives, if any. This however postulates that the attack is effectively detectable from the data stream, and that at least one signature properly characterizes the exploit.

False positives in misuse detection mainly come from an erroneous characterization of the vulnerability. This erroneous characterization often occurs when the IDS attempts to detect the execution of an application without differentiating between normal usage and the actual malicious attempt. For example, detection of CGI attacks is often based on the detection of the script name in the HTTP request, and identically-named scripts induce false positives.

Moreover, it is often difficult to differentiate the interactions between an attacker and a vulnerable information system from the interactions between normal users and the same information system. It is thus important to analyze alerts with the knowledge of the configuration of the monitored system, to ensure proper evaluation of the severity of these alerts.

Misuse Detection and False Negatives. Clearly, false negatives in misuse detection occur on new attacks, when there is no signature associated to the vulnerability. Collecting vulnerability information of sufficient quality to write adequate signatures is a time-consuming task, and validation of this information is often limited, due to the sheer number of attack combinations possible. Most often than not, IDS vendors obtain sample event stream information containing the attack and ensure that their tools can detect the occurrence of the sample data in the event stream. This is a long and tedious task.

Let's take a few numbers to illustrate this fact. An IDS today contains between 500 and 2000 signatures. Public vulnerability databases contain anywhere between 6000 and 20000 different vulnerability reports. Hence, there is roughly a one to 10 factor between what an intrusion-detection system knows about vulnerabilities, and what is publicly available. This ratio seems to be fairly stable; one counts between 100 and 150 new vulnerability announcements per month, associated with 10 to 20 new signatures announced by IDS vendors over the same period.

This difference is the product of two factors:

- Not all vulnerabilities are of interest to IDS users, because they affect only rare, specific environments or tools, or they do not provide the attacker with access to the vulnerable system, only limited denial of service. In addition, some of these vulnerabilities are old and affect very old software revisions that are not available anymore.
- A vulnerability may only leave some tracks in specific event streams. If the IDS does not recover this particular event stream, the vulnerability exploit cannot be detected.

Finally, there is the possibility of generic signatures that trigger on multiple vulnerabilities. This happens because attack code is reused from exploit script to exploit script, or because variations resulting in multiple vulnerabilities affect the same operating system or application and result in a single signature. Vendors are focusing on these generic signatures, hoping to cover not only vulnerabilities but also attack principles. Many products include generic buffer overflow detection, hoping to catch new exploits if they fit the attack technique.

Note that misuse-detection techniques are usually less-well suited for the detection of internal malicious activity. Registered users have access to the information system, and are likely to possess enough privileges on this information system to carry out most malicious activities without resorting to the exploitation of known vulnerabilities.

Misuse Detection and Counter-Measures. Misuse detection allows a contextual analysis of the attack and its effects on the monitored information system. This facilitates the understanding of the problem and the decision-making process for corrective or preventive action. Current research in alert correlation includes correlation between vulnerability assessment tools and intrusion-detection tools. Automated lookup of alert references in vulnerability reports will provide the operator with a mean to rank alert severity not only with respect to the attacker's potential gain, but also with the target's potential risk.

When target machine and target service are identified, it is reasonably easy to detect if the attack has some probability of succeeding, and if its effects are incompatible with the site security policy. The operator is able to evaluate the trade-off between the reliability and business objectives of the service, and the security policy objectives. *This is fundamental in counter-measures*: it could be legitimate to let the information system provide services even if compromised.

Evolution of Misuse Detection. Misuse detection prototypes have been initially implemented using first-order logic and expert systems. Current commercial products follow the so-called "signature-based" approach. There are also Petri-nets-related implementations and state transition analysis implementations.

Signature-based intrusion-detection systems usually rely on string or regular expression matching to detect specific pieces of information occurring in the data source. The matching mechanism is constrained further by specifying additional characteristics of the event stream, such as specific communication ports or protocol states. Each of these characteristics describes a particular facet of the vulnerability.

The expression of the signature depends of the level of detail available in the data source during exploitation of the vulnerability. For example, if a web server stops functioning during an attack before log entries can be written to disk, an intrusion-detection system based on log file analysis will not be able to detect the attack. Product vendors today tend to provide extremely wide signatures that will trigger on anything from normal usage to simple scanning, encouraging the notion that intrusion-detection systems cry wolf without cause.

4.1 Anomaly Detection

The general objective of anomaly detection is to define the correct behaviour of the monitored information system. An alert is generated when an event cannot be explained by the model of correct behaviour. This method assumes that an intrusion will induce a deviation from the normal usage of the information system.

Description of the Correct Behaviour Model. The model of correct behaviour can be constructed either from past samples of observed behaviour, of from explicit policy declarations. When an event occurs, the intrusion detection system compares the current activity with the model. An alert corresponds to the deviation of one or several measures between the current activity and the model.

As such, *anything that does not correspond to an explicitly-defined acceptable activity is considered anomalous.* Of course, the efficiency of such a system strongly depends on the capability of the model to represent the activity of the information system. For example, only using measures of CPU activity to model the normal behaviour of an information system would not allow straightforward detection of denial-of-service attacks filling disks or memory. It also assumes that the measures discriminate normal activity and malicious activity, as postulated by Denning [14,15]. Unfortunately, this postulate has not been validated theoretically. Experimental systems show that it is possible to detect some malicious activity by anomaly-based techniques, but do not qualify the coverage of this detection process.

The first models were based on learning techniques. A set of variables is defined that represents the interesting factors of the information system. Acceptable ranges for these variables are defined through observation of past data. A range here can be an association of average and standard deviation, or more complex statistical measures [16]. The model is trained during an observation period, and should converge towards stable values at the end of the observation period.

This area is still a research subject. New models and detection methods are regularly proposed, that improve constantly on existing technologies.

Advantages of Anomaly Detection Anomaly detection has, at least in principle, several advantages over misuse detection. First of all, it should be able to detect usage of unknown vulnerabilities. This is particularly important, as it does not rely on explicit security knowledge.

It also does not rely (or only in a limited way) on operating system specific knowledge, or application-specific knowledge. This is a great advantage when monitoring heterogeneous systems. After measures have been collected for the model, the intrusion detection system performs the modelling and detection process autonomously.

Finally, it can also detect abuse of privileges and insider attacks. Insiders usually have access to the monitored information system and do not need to use well known vulnerabilities to compromise the system and get access to the information they need. Misuse detection seldom detects insider attacks, whereas anomaly detection could show deviations from normal usage patterns.

Anomaly Detection and False Positives. Anomaly detection techniques often have a high false positive rate. This phenomenon arises from the fact that deviations from the model are often observed for any incident occurring on the monitored information system. Deviations also occur with configuration changes. Hence, the workload for processing alerts is large, and the operators have a frequent feeling that alerts are irrelevant to security.

This feeling is aggravated by the lack of explanation coming with the alerts. The root cause of the phenomenon is unknown, and the information provided seldom helps in resolving the issue.

Anomaly Detection and False Negatives. False negatives in anomaly detection have two main causes, corruption of the behaviour model and absence of measurement.

Corruption of the behaviour model occurs when the model learns an intrusive behaviour and incorporates it in its coverage. The intrusion detection system becomes incapable of detecting occurrences of the attack that has been accepted as part of the normal behaviour of the information system. Learning intrusive behaviour as normal occurs in particular in intrusion-detection systems where the model is constructed using past samples. Such systems need to be retrained periodically, and unfiltered training data could include malicious behaviour. Current research is therefore going away from learning technologies, and developing specification-based techniques to construct the model of normal behaviour.

Also, attacks sometimes do not impact the measures used by the model of normal behaviour. Let's take the very simple example of an intrusion-detection system that would monitor CPU usage and not disk usage. An attack that would fill the disk would not be detected by such a system. Of course, intrusion-detection systems make use of much more complex measures, including dynamic ones. As such, the exact coverage of the monitoring is difficult to establish.

Anomaly Detection and Counter-Measures. Alerts coming from an anomaly-based intrusion-detection system are often difficult to analyze. Counter-measures are difficult to deploy because neither target nor attack source are clearly identified, as well as the attack principle. Without an explicitly identified attack principle, counter-measures become extremely hazardous.

A new approach based on honeypot-like technology has recently been developed to improve identification of attack sources. When suspicious requests are identified, the response provided by the intrusion-detection system contains uniquely-identifiable information. When these specific tags come back, the intrusion-detection system can clearly identify the anomaly and its source.

5 Security Information Management

Once alerts are generated, they need to be handled by operators. Due to the volume and diversity of alert sources, security information management (SIM) platforms have emerged in recent years as the solution for concentrating heterogeneous logs and providing the security officer with a homogenous view of the security state of its information system.

The requirement for a central event and alert processing platform comes from the fact that many devices only provide a partial view of the security state of the information system, coherent with their role. Offering the desired global view can only be done by concentrating and consolidating as many information sources as possible. Note, however, that this does not mean that there will be only one SIM platform per organization, as SIM platforms should be able to communicate with one another.

5.1 SIM Functions

A SIM platform should cover the following four functions today.

Event Acquisition. Event acquisition deals with gathering and transporting events to a central point for further processing. This function covers the reliability of event transport, allowing both push and pull collection models, over a variety of protocols, to ensure that firewalls and other access control devices are properly traversed. This function has to deal with fairly heavy data flows, and it should be able to send hundreds to thousands of events per second to the central platform.

Another task carried out during event acquisition is related to filtering, aggregation and normalization. Given the enormous volume of event information that needs to be inserted in the database, the acquisition process must be able to select which events get inserted into the database. Also, for regular event streams, it is sometimes preferable to aggregate several identical events as one, adding the count of such aggregated events to the one tat is finally inserted in the database.

Finally, the normalization part of the acquisition process deals with ensuring a uniform representation of events in the database. There are differences in the naming conventions adopted by security tools such as intrusion detection systems or anti-virus systems. Two products tend to name the same attack with different signatures. The normalization process aims at ensuring that two events representing the same attack get the same name. Also, this process includes the capability to add reference information to the signatures, to ensure that internal references and processes are properly taken into account.

Contextual Information Management. Alert information usually includes some identification of the victim or source of the attack, identifying users and machines affected. This identification is often partial, including only network addresses or host names. However, most organisations maintain inventory information or vulnerability information assessment. This information should be attached to the host or user independentely of the host or user representation provided in the alert.

Therefore, the role of the contextual information management function is to ensure that all the contextual data is properly attached to hosts and users, and managing changes in this data so that it is kept up to date and accurate. This can be a tricky task in dynamic environments, for example with DHCP [17] or when remote users connect via VPN connections.

Alert Correlation. Alert correlation has as main objective to decide which alerts should be presented first to the security officer. It is in essence a triage and priority management system, which must ensure that the most critical alerts will be seen first. This triage system is supported by the association of a priority or security level with each event. Priority levels and schemes vary, but this is the role of the acquisition process to ensure that they are normalized to the IDMEF [1] set of values. To ensure that this triage process is successful, correlation must:

– fuse alerts that represent identical threat information together so that this threat is handled only once. This process is made difficult by the fact that clocks are often not exactly synchronized, and that some hypotheses must be made as to whether the fused events have the same root cause, e.g. have been raised by the observation of the same packet.
– relate alerts that participate in the same threat. Real attacks translate in multiple attacker actions, translating into multiple observations and multiple alerts being generated by the various intrusion detection and monitoring systems.
– aggregate high-volume alerts that cannot be interpreted individually, to ensure that patterns of aggregate alerts conform to the usual behaviour of the information system.
– incorporate contextual information into the evaluation of the severity of the event, to ensure that it has the proper awareness level. The most common representation of this process is to compare alert information and vulnerability assessment information, to inform the security officer of attacks associated with a security risk.

Alert correlation is an important research topic, particularly related to the processing of large volumes of alerts and to the intelligence of the correlation process.

Reporting and Exchanging. Finally, one needs to realize that a SIM platform does not live in isolation, but must offer several interfaces for accessing and pushing information. Typically, the following interfaces need to be provided :

– Operator real-time interface. This interface provides real-time alert information to the operator, typically through a scrolling window. This is the most common interface available in SIM consoles today, but not the most useful one, as operators need to be constantly on watch.
– Forensics analysis console. This interface provides navigation capabilities over the database of alerts, so that the security analyst can understand the incident, gather all related alerts, and propose solutions for better detection and resolution of the threat in later instances.
– Real-time incident reporting. In many cases, the threat requires countermeasures that have an impact on the normal function and the configuration of the monitored information system. However, if the configuration of the information system is not handled by the SIM console, it needs to send threat information to the system management console for proper handling.

The reporting and exchanging modules can also be used to create the peer relationships between SIM consoles or hierarchical relationships according to the needs of the organization.

5.2 Alert Representation

To support these functions, we have organized our data model as a set of concentric circles. Our data model is inspired from the Snort relational database schema, the IDMEF message format [1], and the M2D2 model [18]. We participated in deploying these tools and developing these models, so they naturally were used as a starting point for our development. However, we believe that event and contextual information are not equivalent and this is not obvious in the three models cited before. Hence, we choose to provide a different representation shown in figure 2.

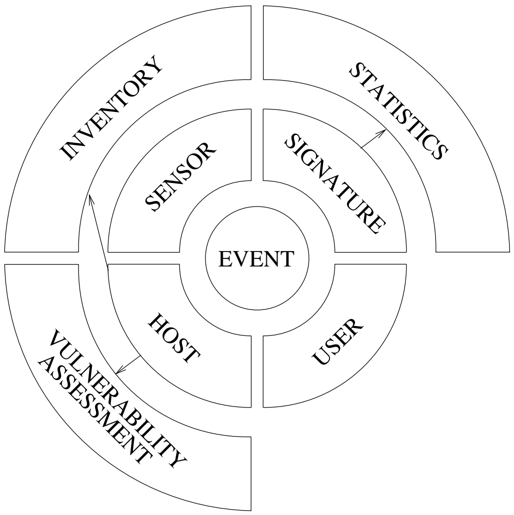

Fig. 2. Representation of alerts in the database

Event Information. The inner circle represents core event information, sensor, signature and timestamp. Note that the two first bits (sensor and signature) are in fact quite complex, comprising several tables and attributes in our database schema. These more complex bits are stored in the second circle, and each event links to the second circle for sensor and signature reference. This mechanism naturally takes into account differences in volume, as there are only a few hundred different sensors and a few thousand signatures for several million events. It also naturally renders the fact that sensor and signature information evolves on a much longer timeframe than event information.

Contextual Information. The second circle represents contextual information. This core event information links to host and user information in the second circle. Signature information links to sensor configuration, to indicate whether a sensor is able to detect an attack or not, and under which condition. Contextual information is mostly generated by the knowledge management processes (see 5.3) and by statically entered configuration information. While this contextual information evolves slowly over time, there is a need to track changes, as they have an impact on the signification of the events. Signatures are tracked using revision numbers that reflect improvements in their design. As such, events attached to the same signature message but with a higher revision number are considered more reliable than earlier events. The same process is used to track the evolution of sensor properties, each property change being tagged with the appropriate timestamp.

Transient Information. The third circle represents transient information that is generated by correlation processes (see Section 6). For example, statistical processes need to store numerical values associated with the statistical model; we use this area for the EWMA control values that monitor signature activity [19]. This circle also links to the first two. For example, the signature trending tool associates signature information in the second circle and event flows stored in the first circle.

The arrows of Figure 2 represent examples of links between contextual information and event information. A network event links to host information with source and destination of the network connection. A system event links to local host information indicating on which host the event occurred.

5.3 Contextual Information

Contextual information is related to the description of hosts and users. The objective of the contextual information knowledge management module is to ensure that the information linking alerts on one hand and hosts and users on the other hand are kept synchronized.

Structure of Host Representation. Logs represent host by three different keys, a host *name*, a host *IP address* and a host *MAC address*. The name is either fully qualified or a simple machine name, depending on the information source. This type of information is often provided by host-based information sources, or by devices configured to do on-the-fly reverse DNS mapping. An IP address is often provided by network-based IDS sensors and other network equipments. Finally, MAC addresses are provided by low-level networking devices such as wireless access points and switches, when specific network or wireless attacks are detected. All three keys are frequently found in event logs.

Different information sources will describe the host using different keys. To ensure that the same device is recognized by different sources, these three keys are associated in the same structure. Each key is associated with a Boolean value indicating whether this key was used by an information source or was derived from a data enrichment process. Upon insertion of an event, the process will first retrieve the appropriate host key with the Boolean set to $TRUE$, checking whether the same device has already been accessed. It will then re-query the same host key with the Boolean set to $FALSE$, checking whether enrichment of the host has taken place for an already existing host. If this is the case, it means that the host was previously inserted using another key coming from another source. For example, an IP address could be used for the first insertion, then a reverse DNS resolution could provide the host name that could then be encountered in host events. If such a host entry is found, the Boolean associated with the key will be set to $TRUE$ to facilitate future insertions, and the host entry found will be used to insert the event. If both searches fail, it will create a new host entry.

The knowledge management process attempt among other things to complete the key information associated with a host in the database. If a host is identified by an IP address, then a reverse DNS lookup is attempted to obtain the host name. If a host is identified by a host name, a DNS lookup is also attempted. Both operations are costly and would result in undue delays upon insertion of a new host, hence the choice of off-loading the acquisition process and pushing such task to a background process.

While defining the keys vas fairly straightforward, it happens that there are a number of issues with the keys that we have designed.

Host Information Collection Point. The first remark is that we collect host information in a different location that event information. Most if not all of our sensors are passive devices, to limit the risk of attack against them. Therefore, they do not have the capability of adding host information to events. As a result, the gathered host information is from the vantage point of the application server and not the sensor. The advantage is that all events are tagged from the same viewpoint, thus normalizing the events. The disadvantage is that the application server needs to be able to access all host information, and that static local information (e.g. host names stored in `/etc/hosts`) will not be accessible. Even though one could fear that the visibility from the application server and from the sensor is different, we have not observed wrong host information as a result of this process.

Network Address Translation. Network address translation [20] is frequently used by private companies and internet service providers to mask the internal structure of their environment and to lessen address space requirements. As such, different machines may be seen as a single one by our application. When NAT is in place, the name resolved through DNS is the name of the NAT device and does not reflect the exact name of the target or source of the event. As a result, our application in this configuration identifies a domain but not the exact target or source.

This problem rarely occurs for hosts under our control. This means that we usually can still precisely identify hosts that are within our realm.

Dynamic Host Configuration. The dynamic host configuration protocol (DHCP [17]) allows the same machine to have multiple IP addresses over time.

Moreover, host name information is sometimes generic as well, reusing for example the two last bytes of the IP address. When this is the case, our application is not able to uniquely identify a machine.

DHCP is a frequent occurrence in the environments monitored by our application. Our tool attempts to determine whether the host is within a DHCP environment. When this is recognized, the host is identified by its host name, which is stored in our DNS servers during DHCP handshake and is uniquely generated in the corporation. When such a host is identified by an IP address, the knowledge management tool reconfigures the event-host associations a posteriori, using DNS queries. This is a domain-specific solution and may not be applicable to other environments. In particular, ISPs tend to use generic names, as mentioned earlier. Another solution would be to use the MAC address, but this key is very rarely available.

Mobility. Mobility is a frequent occurrence in our corporate environment. Laptops and the use of DHCP facilitate remote connections, as well as the generalization of VPN connectivity. However, this poses a problem when such a host is the subject of a security problem, particularly a viral or worm infection. We need to distinguish the case of laptops connected internally into a site that is not their home site, and laptops using VPN connections to access the information system.

In the case of locally-connected laptops, it is often the case that the resolution between host name and host IP address is wrong. Let's take the case of a virus infection. This virus infection is logged into the NT Event Log of the laptop, which in turns connects to a central server to deliver the infection alert. This infection alert is based on the host name, which is a unique key in our corporate network. As such, the event will be correctly assigned to the host. Unfortunately, geographic information is based on the IP address. If an IP address already exists for the host name, this information is not systematically refreshed as this is a costly process. Therefore, getting the current geographic coordinates of the infected laptop requires an additional DNS lookup to retrieve the current IP address and its associated geographical location. This process is quite time consuming and correct information may not be immediately available; if this process takes too much time the connection is terminated.

In the case of VPN-connected laptops (which is also used for wireless connections), the IP address of the laptop resolves to the IP address of the VPN concentrator. Therefore, it is impossible to retrieve the physical location of the infected machine and the connection is terminated.

5.4 Vulnerability Assessment

Collecting vulnerability assessment information is generally done through the usage of a vulnerability scanner, either remotely by querying the audited host, or locally by installing an inventory module. Local auditing us usually more accurate, but requires software installation on the tested host, which is not always feasible. Therefore, vulnerability assessment information is often connected through the network.

While vulnerability assessment reports are extremely useful for the security officer, they suffer from the following issues:

Server side only Remote vulnerability assessment report query active services. Therefore, they do not provide information about client side vulnerabilities or firewalled ports. Only local software installation can provide this information.

Audit cost and timeliness More than actual bandwidth consumption these days, the audit cost is the time it takes to test and evaluate large numbers of hosts. Therefore, audits may only be carried out at spaced intervals.

Audit risk and accuracy Certain tests can have undesirable side effects on the tested host, such as leaving it vulnerable to certain attacks, or bringing it down. The more accurate an audit report is, the riskier it generally is as well.

To take into account some of these issues, passive network observation has been introduced to collect information related to vulnerabilities on both clients and servers. By collecting product names and versions from the network, either with a dedicated tool such as ettercap or with a network intrusion detection system equipped with a specific rule set, the passive network observation sensor can obtain a fairly complete inventory of the information system. Using external vulnerability databases such as OSVDB, it is then possible to deduce vulnerability assessment information for the information system.

6 Alert Correlation

The need to automate alert processing and to reduce the amount of alerts displayed to the operator by the system is a widely recognized issue and the research community has proposed as one solution to correlate related alerts to facilitate the diagnostics by the operator [21].

Alert correlation has three principal objectives with regard to information displayed to the operator:

Volume reduction: Group or suppress alerts, according to common properties. E.g. several individual alerts from a scan should be grouped as one meta alert.

Content improvement: Add to the information carried by individual alert. E.g. the use of topology and vulnerability information of monitored system to verify or evaluate the severity of the attack.

Activity tracking: Follow multi-alert intrusions evolving as time passes. E.g. if attacker first scans a host, then gains remote-to-local access, and finally obtains root access, individual alerts from these steps should be grouped together.

We perform volume reduction eliminating redundant information by aggregating alerts that are not strictly symptoms of compromise and appear in high volumes. Only changes in the behaviour of the aggregate flow are reported to the user. Correlation techniques capable of detecting unknown, novel relationships in data are said to be *implicit* and techniques involving some sort of definition of searched relationships are called *explicit.* As the aggregation criteria are manually selected, this is an explicit correlation method. Overall, we aim to save operator resources by freeing the majority of time units that manually processing the background noise would require and thus to enable him to focus on more relevant alerts. Even though this manual processing is likely to be periodic skimming through the accumulated noise, if there are several sources with omnipresent activity, the total time used can be significant. Next we discuss why despite the large amounts of alerts background noise monitoring can be useful.

6.1 Statistical Correlation

According to our experience (see Sect. 6.1) a relatively large portion of alerts generated by a sensor can be considered as background noise of the operational system. However, the division to true and false positives is not always so black and white. The origins of problem can be coarsely divided to three. 1) Regardless of audit source, the audit data usually does not contain all required technical information, such as the topology and the vulnerability status for the monitored system for correct diagnosis. 2) The non-technical contextual factors, such as operator's task and the mission of the monitored system, have an effect on which types of alerts are of high priority and relevant. 3) Depending on the context of the event, it can be malicious or not, and part of this information can not be acquired by automated tools or inferred from the isolated events. For the first case, think of a Snort sensor that does not know if the attack destination is running a vulnerable version of certain OS or server and consequently can not diagnose whether it should issue an alert with very precise prerequisites for success. An example of the second is a comparison of on-line business and military base. At the former the operator is likely to assign high priority on the availability of the company web server, and he might easily discard port scans as minor nuisance. At the latter the operator may have only minor interest towards the availability of the base web server hosting some PR material, but reconnaissance done by scanning can be considered as activity warranting user notification. Instead of high priority attacks, the third case involves action considered only as potentially harmful activity, such as ICMP and SNMP messages that indicate information gathering or network problems, malicious as well as innocuous as part of normal operation of the network. Here the context of the event makes the difference, one event alone is not interesting, but having a thousand or ten events instead of the normal average of a hundred in a time interval can be an interesting change and this difference can not be encoded into signature used by pattern matching sensor.

This kind of activity is in the grey area, and the resulting alerts somewhere between false and true positive. Typically the operator can not afford to monitor it as such because of the sheer amount of events. The current work on correlation is largely focusing on how to pick out the attacks having an impact on monitored system and show all related events in one attack report to the operator. Depending on the approach, the rest of the alerts are assigned such a low priority that they do not reach the alert console [22], or they can be filtered out before entering the correlation process [23,24]. However, if the signature reacting to grey area events is turned on, the operator has some interest towards them. Therefore it is not always the best solution to only dismiss these less important alerts albeit their large number. Monitoring aggregated flows can provide information about the monitored system's state not available in individual alerts, and with a smaller load on operator. Our work focuses on providing this type of contextual information to the user.

EWMA control charts were originally developed for statistical process control (SPC) by Roberts [25], who used the term geometric moving averages instead of EWMA, and since then the chart and especially the exponentially weighted moving average have been used in various contexts, such as economic applications and intrusion detection [26,27,28]. Our needs differ from those of Roberts' quite much, and also to a smaller degree from those of the related work in intrusion detection. Below our variation of the technique is described, building largely on [28], and the rationale for changes and choices is provided.

The monitored measure is the alert intensity of a flow x, the number of alerts per time interval. One alert flow consists typically of alerts generated by one signature, but also other flows, such as alerts generated by a whole class of signatures, were used. Intensity x is used to form the EWMA statistic. This statistic is called the *trend* at time i. It is quite impossible to define a nominal average as the test baseline x_0, since these flows evolve significantly with time. Like Mahadik et al. [28], to accommodate the dynamic, non-stationary nature of the flows, the test baseline is allowed to adapt to changes in alert flow.

Learning Data. The tool was developed for an IDS consisting of Snort sensors logging alerts into a relational database. The sensors are deployed in a production network, one closer to Internet and two others in more protected zones. This adds to the difficulty of measuring and testing, since we do not know the true nature of traffic that was monitored. On the other hand, we expect the real world data to contain such background noise and operational issues that would not be easily incorporated to simulated traffic.

The data set available to us in this phase contained over 500 K alerts accumulated during 42 days. Of the 315 activated signatures, only five were responsible for 68 % of alerts as indicated in Table 1 and we chose them for further scrutiny. To give an idea of the alert flow behaviour, examples of alert generation intensities for four of these signatures are depicted in Fig. 3. The relatively benign nature of these alerts and their high volume was one of the original inspirations for this work. These alerts are good examples of the problem three and demonstrate the reason why we opt not just filter even the more deterministic components out. For example, the alert flow in Fig. 3(c) triggered by SNMP traffic over UDP had only few (source, destination) address pairs, and the constant component could be easily filtered out. However, this would deprive the operator being notified of behaviour such as the large peak and shift in constant

Table 1. Five most prolific signatures in the first data set

signature name	number of alerts	proportion
SNMP Request UDP	176 009	30 %
ICMP PING WhatsupGold Windows	72 427	13 %
ICMP Destination Unreachable (Communication Administratively Prohibited)	57 420	10 %
LOCAL-POLICY External connexion from HTTP server	51 674	9 %
ICMP PING Speedera	32 961	6 %
sum	**390 491**	**68 %**

component around February 15th as well or the notches in the end of February and during March 15th. Not necessarily intrusions, but at least artefacts worth further investigation. On the other hand, we do not want to distract the operator with the alerts created during the hours that represent the stable situation with constant intensity. For the others, Fig. 3(a) shows alerts from a user defined signature reacting to external connections from an HTTP server. The alerts occur in bursts as large as several thousands during one hour and the intensity profile resembles impulse train. As custom made, the operator has likely some interest in this activity. In Figs. 3(b) and 3(d) we have alerts triggered by two different ICMP Echo messages, former being remarkably more regular than latter. In the system in question, deactivation of ICMP related signatures was not seen as a solution by the operator as they are useful for troubleshooting problems. Consequently, we had several high volume alert flows for which the suppression was not the first option.

Effect of Flow Volume. Judging from the busy interval reduction, the method is useful for alert flows that had created more than 10 K alerts, the effectiveness increasing with the flow volume. The busy interval reduction for flows below 10 K alerts is already more modest, and below 1 K alerts the reduction is relatively negligible. Tables 2 and 3 depict respectively the reduction as percentage from non-zero intervals and alerts flagged anomalous, due to space constraints only for flows of over 10 K alerts. Reduction is shown with smoothing factors 0.80 and 0.92 for each of the three models, continuous, hourly, and weekday. In Table 2 also the total number of active intervals, and in Table 3 the total number of alerts are shown for each flow.

Table 4 summarizes alert reduction results with continuous model and smoothing factor 0.92. All 85 flows are grouped to four classes according to both their output volume (over 100, 1 K, 10 K or 100 K alerts) and the achieved reduction in busy intervals and alerts (below 5 %, 10 %, 50 % or 100 % of original), respectively. These results show also the poorer performance for flows below the 10 K limit. The busy intervals show more consistent relation between the volume and reduction. On the right hand side of Table 4 in the class over 100 K alerts, `ICMP Dest Unreachable (Comm Admin Proh)` stands out with reduction signifi-

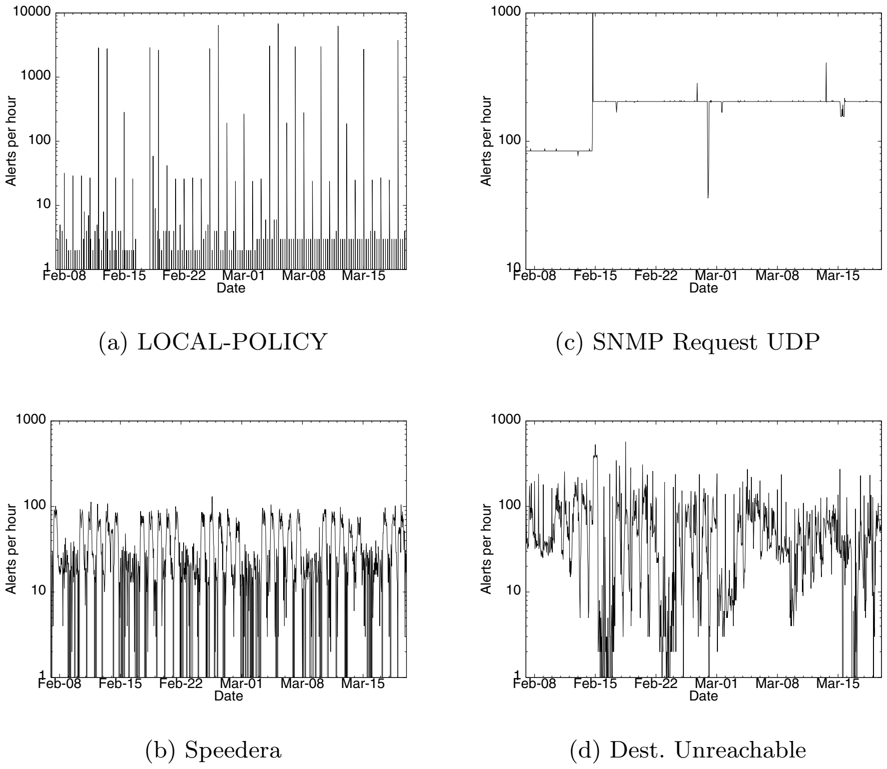

Fig. 3. Hourly alert intensity for some of the most prolific signatures in learning data. Horizontal axis is the time, and the vertical shows the number of alerts per hour

cantly smaller than others in the same class. We found two explanations for this behaviour. First, there was one large alert impulse of approximately 17 K alerts flagged in the test data. This makes up roughly 10 % of flagged alerts. Second, the flow nature is more random compared to others, this is visible in Fig. 3(d) for learning data and applies also for the larger data set. This randomness causes more alert flagging, but still the reduction in busy intervals is comparable to other flows in this volume class.

Reasons for Poor Summarization. There seems to be two main reasons for poorer performance. 1) Many flows had few huge alert peaks that increase the alert flagging significantly. 2) The intensity profile has the form of impulse train that has negative impact both on reduction of alerts and busy intervals. As the first cause does not increase remarkably the number of reported anomalous intervals i.e. the number of times the user is disturbed, this is smaller problem. However, the second cause renders our approach rather impractical for monitoring such a flow, as the operator is notified on most intervals showing activity.

Table 2. The proportion of flagged intervals from intervals showing activity for the flow with different models and smoothing factors

		cont.		daily		weekd.	
flow	**int.**	.80	.92	.80	.92	.80	.92
Known DDOS Stacheldraht infection	563	1.6	1.8	8.9	8.5	2.0	2.5
SNMP request UDP	2311	4.3	2.9	5.8	4.6	4.2	3.0
ICMP PING WhatsupGold Windows	2069	5.1	3.3	5.8	2.6	5.1	3.2
DDOS Stacheldraht agent→handler (skillz)	512	1.2	1.6	12	16	1.8	2.1
ICMP Dst Unreachable (Comm Adm Proh)	2578	5.4	3.5	6.7	5.8	5.4	3.4
ICMP PING speedera	2456	3.3	1.7	4.2	2.9	3.3	0.9
WEB-IIS view source via translate header	2548	5.2	3.8	6.4	5.7	5.1	4.0
WEB-PHP content-disposition	2287	6.8	4.3	7.7	5.2	6.7	4.0
SQL Sapphire Worm (incoming)	1721	2.2	1.2	4.9	3.5	2.4	1.6
(spp_rpc_decode) Frag RPC Records	421	13	7.8	20	20	12	9.0
(spp_rpc_decode) Incompl RPC segment	276	21	13	27	27	22	13
BAD TRAFFIC bad frag bits	432	34	23	37	33	35	22
LOCAL-WEB-IIS Nimda.A attempt	537	24	16	30	25	24	16
LOCAL-WEB-IIS CodeRed II attempt	1229	6.3	4.6	14	14	6.9	5.3
DNS zone transfer	855	9.7	6.7	13	10	9.8	6.5
ICMP L3retriever Ping	107	29	26	71	70	28	23
WEB-MISC http directory traversal	708	12	9.3	15	13	12	9.5
(spp_stream4)STLTH ACT(SYN FIN scan)	29	65	58	82	79	62	62

Table 3. The percentage of flagged alerts with different models and smoothing factors

		cont.		daily		weekd.	
flow	**alerts**	.80	.92	.80	.92	.80	.92
Known DDOS Stacheldraht infection	308548	1.2	1.2	4.4	8.4	1.4	1.5
SNMP request UDP	303201	4.4	3.0	4.9	4.4	4.2	3.2
ICMP PING WhatsupGold Windows	297437	5.4	4.0	4.5	2.9	5.2	3.1
DDOS Stacheldraht agent→handler (skillz)	280685	0.8	1.0	7.3	7.0	1.2	1.2
ICMP Dst Unreachable (Comm Adm Proh)	183020	32	28	39	37	32	28
ICMP PING speedera	95850	5.5	3.1	2.5	2.3	5.3	1.4
WEB-IIS view source via translate header	58600	25	21	12	11	24	22
WEB-PHP content-disposition	48423	18	14	15	13	18	14
SQL Sapphire Worm (incoming)	38905	3.0	1.9	11	9.1	3.1	2.5
(spp_rpc_decode) Frag RPC Records	38804	63	62	94	93	63	62
(spp_rpc_decode) Incompl RPC segment	28715	64	62	93	93	64	62
BAD TRAFFIC bad frag bits	27203	51	42	57	54	53	42
LOCAL-WEB-IIS Nimda.A attempt	25038	65	61	69	64	64	62
LOCAL-WEB-IIS CodeRed II attempt	20418	11	7.5	17	22	11	7.1
DNS zone transfer	15575	32	35	55	55	32	36
ICMP L3retriever Ping	12908	11	12	90	90	11	12
WEB-MISC http directory traversal	10620	41	38	46	45	41	38
(spp_stream4)STLTH ACT(SYN FIN scan)	10182	96	90	93	93	96	96

Table 4. All 85 flows grouped by the number of alerts created and the percentage level below which busy intervals or alerts were flagged

busy interval reduction

alerts	**5 %**	**10 %**	**50 %**	**100 %**
> 100 K	5	0	0	0
> 10 K	5	3	4	1
> 1 K	0	4	19	7
> 100	0	1	12	24
sum	10	8	35	32

alert reduction

alerts	**5 %**	**10 %**	**50 %**	**100 %**
> 100 K	4	0	1	0
> 10 K	2	1	6	4
> 1 K	0	1	15	14
> 100	0	0	8	29
sum	6	2	30	47

The flow `(spp_stream4)` on the last row of Tables 2 and 3 is a typical example, as its alert profile consisted only from impulses. In such situation a large majority of active intervals are flagged as anomalous. A closer look on alert impulses revealed that they were usually generated in such a short time interval that increasing the sampling frequency would not help much. Instead, other means should be considered to process them.

Represented Alert Types. Amongst the most prolific signatures, we can identify three main types of activity, *hostile*, *information gathering* and alerts that can be seen to reflect the *dynamics* of networks.

Hostile activity is represented by DDoS tool traffic and worms with five signatures. The two DDoS signatures are actually the same, different names were used by the operator for alert management reasons. If busy interval reduction below 5 % with continuous model and $(1 - \lambda) = 0.92$ is used to define EWMA monitoring applicable for a flow, then we have three fourths in feasible range for the hostile activity.

In the system in question, possible information gathering is the most common culprit for numerous alerts. This category can be further divided to information gathering on applications (web related signatures) and network architecture (ICMP, SNMP and DNS traffic). In both categories, there are both suitable and unsuitable flows for this type monitoring.

The ICMP Destination Unreachable (Communication Administratively Prohibited) message is an example of the activity that describes the dynamics of the network. It reflects the network state in terms of connectivity, and the origins and causes of these events are generally out of operators control.

Signatures firing on *protocol anomalies* can be considered as an orthogonal classification, since they can present any of the three types above. (`(spp rpc decode)`, `(spp stream4)` and `BAD TRAFFIC`) were all poorly handled by the method. Another common factor is the smaller degree of presence in the data set in terms of non-zero intervals. As the `(spp_stream4)` means possible reconnaissance, and being present only on 29 intervals, it is less likely to be just background noise.

The nature of these alerts and their volumes in general support the claim that large proportion of generated alerts can be considered as noise. Even in the

Table 5. The omnipresence of signatures and their types. Presence measured in active intervals

signature	type	< 5 %	active	present
ICMP Dst Unreachable (Comm Adm Proh)	network	ok	2578	95 %
WEB-IIS view source via translate header	info_web	ok	2548	93 %
ICMP PING speedera	info_net	ok	2456	90 %
SNMP request UDP	info_net	ok	2311	85 %
WEB-PHP content-disposition	info_web	ok	2287	84 %
ICMP PING WhatsupGold Windows	info_net	ok	2069	76 %
SQL Sapphire Worm (incoming)	hostile	ok	1721	63 %
LOCAL-WEB-IIS CodeRed II attempt	hostile	ok	1229	45 %
DNS zone transfer	info_net	no	855	31 %
WEB-MISC http directory traversal	info_web	no	708	26 %
Known DDOS Stacheldraht infection	hostile	ok	563	20 %
LOCAL-WEB-IIS Nimda.A attempt	hostile	no	537	19 %
DDOS Stacheldraht agent-¿handler (skillz)	hostile	ok	512	18 %
BAD TRAFFIC bad frag bits	proto	no	432	15 %
(spp_rpc_decode) Frag RPC Records	proto	no	421	15 %
(spp_rpc_decode) Incompl RPC segment	proto	no	276	10 %
ICMP L3retriever Ping	info_net	no	107	3 %
(spp_stream4)STLTH ACT(SYN FIN scan)	proto	no	29	1 %

case of hostile activity the originating events warrant aggregation. This applies in our case, but the situation may vary with different operating environments.

Table 5 shows the signature flows ordered by their omnipresence giving the number of active intervals and the percentage this makes out of the whole testing interval. A rough division according to the 5 % watershed is made and type of signature according to above discussion is assigned. We can see that for all signatures showing activity on more than 45 % of the intervals the number of alerts issued to operator can be significantly reduced in this system.

It would seem that the omnipresence of alerts would be better criteria than the alert type for determining whether EWMA monitoring would be useful or not.

Impact of Time Slot Choice. According to these metrics the usefulness of daily and weekday models was limited to a few exceptions, generally the continuous model was performing as well as the others. We just happened to have one of the exceptions that really profited from hourly approach in our early experimentations, and made the erroneous hypothesis of their commonness. The metrics are however limited for this kind of comparisons. It is especially difficult to say if the hourly approach just marks more intervals as anomalous or is it actually capturing interesting artefacts differently. On many occasions the smaller reduction was at least partly due to abrupt intensity shifts. As several different statistics making up the hourly model signal an anomaly whereas the continuously updated statistic does this only once. The two DDoS flows had intensity profiles resembling a step function, which caused the hourly model to flag significantly more alerts than the continuous.

Another factor encumbering the comparisons is the difference in efficient lengths of model memories. As the time slot statistics of hourly and weekday models are updated with only the corresponding intensity measures the values averaged have longer span in real time. For example the hourly model's statistics are affected by 8 or 24 *days* old measurements.

Class Flows. Grouping signature classes together increased the flagging percentage. Table 6 shows obtained reductions with continuous model and $(1-\lambda) = 0.92$ for class aggregates with more than 1000 alerts. In fact, almost every class contains one or more voluminous signatures that were problematic statistically already by themselves, and this affects the behaviour of class aggregate. The increased flagging could also indicate that anomalies in signature based flows with smaller volume are detected to some degree. The levels of busy intervals are reduced relatively well and again generally the flagging increases as alert volume decreases. The aggregation by class might be used to gain even more abstraction and higher level summaries in alert saturated situations. However, there are likely to be better criteria for aggregation than the alert classes.

Table 6. The reduction in alerts and busy intervals when aggregating according to signature classes. Results for continuous model with $1 - \lambda = 0.92$

	raw		**anomalous**	
flow	**int.**	**alerts**	**int.**	**alerts**
misc-activity	2678	618273	1.9 %	8.9 %
class_none	1429	380568	4.8 %	18.3 %
attempted-recon	2635	360613	3.7 %	7.0 %
known-issue	563	308548	1.7 %	1.1 %
web-application-activity	2569	88554	3.3 %	16.3 %
bad-unknown	2559	65883	3.7 %	20.9 %
known-trojan	1511	46014	5.4 %	34.9 %
misc-attack	1727	39070	1.3 %	2.1 %
web-application-attack	1017	9587	9.1 %	40.5 %
attempted-user	272	3694	19.4 %	40.6 %
attempted-dos	361	2782	24.3 %	67.8 %
attempted-admin	444	1760	20.2 %	33.1 %

Flow Stability. To give an idea of the stability of flow profiles, Table 7 compares the alert and busy interval reduction obtained for four signatures used in the learning phase against the reduction in testing data. In general the flagging is slightly higher in the training data set. The most notable exception is signature `ICMP Destination Unreachable (Communication Adm Prohibited)`, where a significant number of alerts is marked anomalous in the test set. The large alert impulse in this flow, mentioned earlier, accounts for approximately

Table 7. A comparison of results obtained during learning and testing phases. $(1-\lambda) = 0.92$

	alerts		**intervals**	
flow	**learn.**	**test**	**learn.**	**test**
SNMP request UDP	2.7	3.5	2.2	3.5
ICMP PING WhatsupGold Windows	4.6	3.6	2.9	3.6
ICMP Dst Unreachable (Comm Adm Proh)	12	36	3.2	3.7
ICMP PING speedera	2.8	3.2	1.3	2.0

14 % units of this increase in test data. Even if those alerts were removed, the increase would be large. Still, the reduction in busy intervals is quite similar, suggesting higher peaks in the test set. The fifth signature enforcing a local policy, also viewed in the learning phase, did not exist anymore in the testing data set. This signature created alert impulses (see LOCAL-POLICY in Fig. 3(a)) and the alert reduction was marginal in learning data.

It seems like with the used parameters the reduction performance stays almost constant. This would suggest that after setting parameters meeting the operators needs, our approach is able to adapt to lesser changes in alert flow behaviour without further adjustment. At least during this test period, none of the originally nicely-enough-behaving flows changed to more problematic impulse-like nor vice versa. Also signatures having a constant alert flow or more random process type behaviour, both feasible for the method, kept to their original profile.

To wrap up the results, it seems possible to use this approach to summarize and monitor the levels of high volume background noise seen by an IDS. Up to 95 % of the one hour time slots showing activity from such an alert flow can be unburdened from the distraction. For the remaining intervals, instead of a barrage of alerts, only one alert would be outputted in the end of the interval. As both data sets came from the same system, the generality of these observations is rather limited, and more comprehensive testing would be required for further validation.

If the user is worried that aggregation at signature level loses too much data, it is possible to use additional criteria, such as source and destination addresses and/or ports to have more focused alert streams. The reduction in aggregation is likely to create more flagged intervals, and this is a tradeoff that the user needs to consider according to his needs and the operating environment. Determining if the summarization masked important events in the test set was not possible, as we do not possess records of actual detected intrusions and problems in the monitored system against which we could compare our results.

6.2 Correlation with Vulnerability Information

Correlation between alerts and vulnerability information aims at assessing the risk that the monitored information system incurs from the attacker's actions. The actions of the attacker are deemed potentially successful and high risk if the vulnerability exists on the information system.

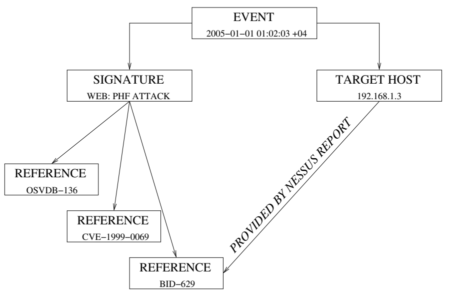

Fig. 4. Explicit correlation between event and vulnerability information, using vulnerability assessment reports.

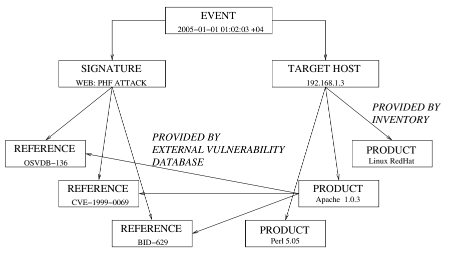

Fig. 5. Explicit correlation between event and vulnerability information, using inventory information.

The easiest way to realize this correlation mechanism is shown in Figure 4. An event contains information about the attack mechanism and the target host of the attack. The attack mechanism is associated with references, often to public external databases, that constitute an open dictionary for all intrusion detection and SIM vendors. The vulnerability assessment information provides the link between hosts and vulnerability references. When a loop can be found in the graph, the alert represents an attack for which the information system was vulnerable, hence a more serious risk.

The difficulty with this kind of correlation is that we may not have a vulnerability assessment report for the host. When this is the case, we may use inventory information, either created externally or through passive network observation as mentioned in Section 5.4. As shown in Figure 5, the association between hosts and vulnerability references goes through product information.

Note that this procedure also offers another advantage, which is the possibility to correlate alerts with non-existent vulnerabilities. Vulnerability assessment tools rarely indicate explicitly when a server is not vulnerable to a given attack. This lack of information can be attributed to the absence of the vulnerability, but there could be other factors that make the test fail or not complete, while the vulnerability would still exist. However, vulnerability databases often include non-vulnerable information related to the product versions. If the product versions are comparable, then it is also possible to lower the severity of an alert when the risk does not exist in the information system.

This correlation mechanism is largely in use in SIM consoles. We are currently studying the efficiency of this mechanism, to precisely evaluate what it can and cannot provide.

7 Conclusion

In this paper, we have presented intrusion detection and security information management as two important and active research domains for information systems security. While intrusion detection offers mature technologies for deployment, security information management remains an interesting research subject from which we can expect new advances.

Related to intrusion detection, we expect that research will focus on application-level attacks and on much more accurate sensors and detection algorithms than are currently available. The sheer number of uninteresting intrusion detection alerts generated by these tools will require continuous tuning and development, until systems become more secure.

Security information management will continue to foster research in alert correlation, leading to more complex scenarios that actually provide reliable threat information to the security officer. Once this stage is reached, we will see a large body of research taking place on automated countermeasures, i.e. ensuring that attacks are dealt with efficiently and accurately, with as little human intervention as possible.

References

1. Debar, H., Curry, D., Fenstein, B.: Intrusion Detection Message Exchange Format Data Model and Extensible Markup Language (XML) Document Type Definition. Internet Draft (work in progress) (2005) http://search.ietf.org/internet-drafts/draft-ietf-idwg-idmef-xml-14.txt.
2. Feinstein, B., Matthews, G., White, J.: The intrusion detection exchange protocol (idxp). Internet Draft (work in progress), expires April 22nd, 2003 (2002)
3. Roesch, M.: Snort - Lightweight Intrusion Detection for Networks. In: Proceedings of LISA'99, Seattle, Washington, USA (1999)
4. Northcutt, S., Novak, J.: Network Intrusion Detection. 3 edn. QUE (2003) ISBN 0735712654.
5. Ptacek, T.H., Newsham, T.N.: Insertion, Evasion, and Denial of Service : Eluding Network Intrusion Detection. Secure Networks, Inc (1998)
6. Zhang, Y., Paxson, V.: Detecting stepping stones. In: Proceedings of the 9th USENIX Security Symposium, Denver, CO (2000)
7. Paxson, V.: An analysis of using reflectors for distributed denial-of-service attacks. Computer Communication Review **31** (2001)
8. Handley, M., Kreibich, C., Paxson, V.: Network Intrusion Detection: Evasion, Traffic Normalization, and End-to-End Protocol Semantics. In: Proceedings of the 10th USENIX Security Symposium, Washington, DC (2001)
9. Fieldings, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., Berners-Lee, T.: Hypertext Transfer Protocol – HTTP/1.1. RFC 2616 (1999)
10. Denning, D.E., Edwards, D.L., Jagannathan, R., Lunt, T.F., Neumann, P.G.: A prototype IDES — A Real-Time Intrusion Detection Expert System. Technical report, Computer Science Laboratory, SRI International (1987)
11. Snapp, S.R., Smaha, S.E.: Signature Analysis Model Definition and Formalism. In: Proc. Fourth Workshop on Computer Security Incident Handling, Denver, CO (1992)
12. Boyer, R.S., Moore, J.S.: A fast string searching algorithm. Communications of the ACM **20** (1977) 762–772
13. Thomson, K.: Regular expression search algorithm. Communications of the ACM **11** (1968) 419 – 422

14. Denning, D.E., Neumann, P.G.: Requirements and model for IDES - a real-time intrusion detection expert system. Technical report, Computer Science Laboratory, SRI International, Menlo Park, CA (1985)
15. Denning, D.: An Intrusion-Detection Model. IEEE Transactions on Software Engineering **13** (1987) 222–232
16. Javitz, H.S., Valdez, A., Lunt, T.F., Tamaru, A., Tyson, M., Lowrance, J.: Next generation intrusion detection expert system (NIDES) - 1. statistical algorithms rationale - 2. rationale for proposed resolver. Technical Report A016–Rationales, SRI International, 333 Ravenswood Avenue, Menlo Park, CA (1993)
17. Droms, R.: Dynamic host configuration protocol. RFC 2131 (1997)
18. Morin, B., Mé, L., Debar, H., Ducassé, M.: M2D2 : A Formal Data Model for IDS Alert Correlation. In: Proceedings of the Fifth International Symposium on Recent Advances in Intrusion Detection (RAID). (2002)
19. Viinikka, J., Debar, H.: Monitoring ids background noise using ewma control charts and alert information. In: Proceedings of the 7th International Symposium on Recent Advances in Intrusion Detection (RAID 2004). Lecture Notes in Computer Science, Springer-Verlag (2004)
20. Egevang, K., Francis, P.: The ip network address translator (nat). RFC 1631 (1994)
21. Debar, H., Morin, B.: Evaluation of the Diagnostic Capabilities of Commercial Intrusion Detection Systems. In: Proceedings of RAID 2002. (2002)
22. Porras, P.A., Fong, M.W., Valdes, A.: A Mission-Impact-Based Approach to INFOSEC Alarm Correlation. In: Proceedings of RAID 2002. (2002) 95–114
23. Julisch, K.: Mining Alarm Clusters to Improve Alarm Handling Efficiency. In: Proceedings of the 17th Annual Computer Security Applications Conference (ACSAC). (2001)
24. Morin, B., Debar, H.: Correlation of Intrusion Symptoms: an Application of Chronicles. In: Proceedings of RAID 2003. (2003) 94–112
25. Roberts, S.W.: Control Chart Tests Based On Geometric Moving Averages. Technometrics **1** (1959) 230–250
26. Ye, N., Vilbert, S., Chen, Q.: Computer Intrusion Detection Through EWMA for Autocorrelated and Uncorrelated Data. IEEE Transactions on Reliability **52** (2003) 75–82
27. Ye, N., Borror, C., Chang, Y.: EWMA Techniques for Computer Intrusion Detection Through Anomalous Changes In Event Intensity. Quality and Reliability Engineering International **18** (2002) 443–451
28. Mahadik, V.A., Wu, X., Reeves, D.S.: Detection of Denial of QoS Attacks Based on χ^2 Statistic and EWMA Control Chart. Online document, http://arqos.csc.ncsu.edu/papers.htm (2002) Submitted for Usenix 2002.