Survey of Event Correlation Techniques for Attack Detection in Early Warning Systems

Tobias Limmer and Falko Dressler

Computer Networks and Communication Systems
University of Erlangen, Germany
{limmer,dressler}@informatik.uni-erlangen.de

Technical Report 01/08
University of Erlangen, Department of Computer Science 7

Abstract. In the context of early warning systems for detecting Internet worms and other attacks, event correlation techniques are needed for two reasons. First, network attack detection is usually based on distributed sensors, e.g. intrusion detection systems. During attacks but even in normal operation, the generated amount of events is hard to handle in order to evaluate the current attack situation for a larger network. Thus, the concept of event or alert correlation has been introduced. This survey was motivated by recent work on early warning systems. We summarize and clarify the typical terminology used in this context and present a requirement analysis from an early warning system's point of view. In the main part of this survey, we summarize and classify event correlation techniques as described in the literature.

1 Introduction

In order to cope with the increasing quantity and quality of recent attacks in the Internet, automated attack detection and mitigation techniques are strongly required [1,2]. For example, the German government is currently investing into a national IT early warning system, which is intented to help with early detection of new Internet attacks. This system should not only detect singular attacks but also distributed worm spreadings and others.

Figure 1 depicts the main components of such an IT early warning system [3]. On the lower layer, network sensors such as monitors, which are providing packet and flow data, or quality of service measurements collect information about the current network behavior. This data can be analyzed on the next layer in which Intrusion Detection Systems (IDSs) interpret the received information. The information exchange between the sensors and the analyzers can be performed using standardized data formats such as Internet Protocol Flow Information Export (IPFIX) and Packet Sampling (PSAMP). A key challenge for the efficient use of the monitoring infrastructure is the ability to dynamically reconfigure the sensors according to the current requirements on the attack detection level. This can be done using standard network management techniques but also based on

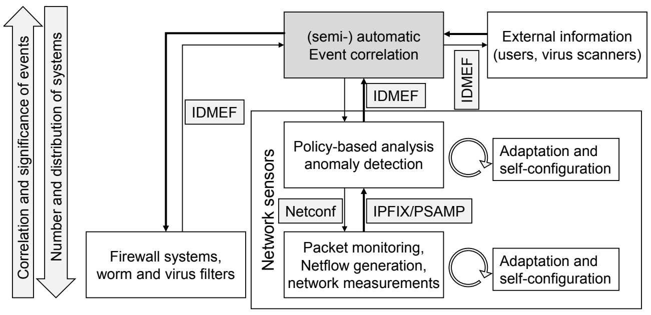
Correlation and significance of events
Number and distribution of systems
(semi-) automatic Event correlation
External information (users, virus scanners)
IDMEF
IDMEF
IDMEF
Network sensors
Policy-based analysis anomaly detection
Adaptation and self-configuration
Netconf
IPFIX/PSAMP
Firewall systems, worm and virus filters
Packet monitoring, Netflow generation, network measurements
Adaptation and self-configuration

Fig. 1. General architecture of early warning systems

self-organization methods that allow an autonomous self-controlled monitoring and attack detection environment.

As required by the volume of attack traffic in the current Internet, monitoring and attack detection need to be performed in a highly distributed way. This becomes possible using decoupled packet monitoring, attack or intrusion detection, and event analysis.

The use of distributed network sensors requires the correlation of detected alarms or events. Therefore, on the highest level, all the collected attack information need to be collected on centralized systems for further analysis. We refer to this level as *event correlation* because connections between multiple events are detected based on logical relations such as temporal or spatial correlations – in the literature, sometimes the term *alarm correlation* is used as a synonym. Metainformation can be used for improved event correlation. Again, the information exchange between the attack detection level and the correlation entities can be based on standardized data formats such as Intrusion Detection Message Exchange Format (IDMEF).

The contributions of this study are threefold:

– We start with some clarification of the used terminology in Section 2. While there is a common agreement on the meaning of attack or intrusion detection, the terms *event* and *event correlation* are not clearly outlined in the literature.
– Furthermore, we provide a summarized requirements analysis related to the entire IT early warning system based on distributed network sensors up to the event correlation engine (Section 3).
– The main part of this study is a comprehensive survey of event correlation techniques (Sections 4 and 5). This includes methods and procedures that have been described in the literature not only as event detection but also in related fields, e.g. attack detection.

2 Terminology

In the scope of this section we describe terms used in this report and define their meaning. Especially it is important to to distinguish between *intrusion detection* and *event correlation*, as well as between *event* and *meta data*.

2.1 Intrusion Detection System

Intrusion Detection Systems (IDSs) have been developed to detect attacks or intrusions against / into systems connected to the Internet. According to Bace et al. [4], an IDS is a software or hardware based system which automatizes this monitoring and analysis process. The resulting information ("events") are stored or transmitted to be evaluated by a network operator.

The term IDS has been firmly established in the literature by now. In its literal meaning, the term "intrusion" characterizes only a small part of attacks in a network. This report uses the more general term "attack detection", which is also increasingly used in current literature. We also employ the term attack detection system in this report as comprehensive description for systems which detect and analyze security related incidents.

Attack detection systems may be classified according to their information source. *Network based* attack detection systems monitor networks and collect transferred data. By analyzing the data, it is possible to detect attacks being performed over the network. *Host based* attack detection systems process information, which is collected within a single host, e.g. application log files.

Figure 2 shows an example for a comprehensive attack detection systems that also provides means of countermeasures. Sensors (host-based IDS or network-based IDS) report detected attacks or anomalies to the analyzer. The analyzer itself (re-)configure the attack detection system itself, end-systems or firewalls, so that incidents can trigger automatic reactions. Honeypots and sandboxes offer the possibility to proactively learn new attack signatures, which may be employed on the attack detection systems. This process is also controlled by the analyzer.

2.2 Event

Events are individual or aggregated messages or alarms describing or relating to activities in a network. This rather vague definition covers almost all events, which are exchanged in all kinds of monitoring and evaluation systems. On the one hand, it may be information about the content of a network data packet, or, on the other hand, it may be a message about a worm propagating through the Internet with additional information about the exploited security hole. In this study, we will discuss the structure of certain event types and their contextual meaning.

RFC 4766 [5] defines an event as follows: *The occurrence in the data source that is detected by the sensor and that may result in an IDMEF alert being transmitted, for example, attack.* Similarly, the term *alert* is used: *A message*

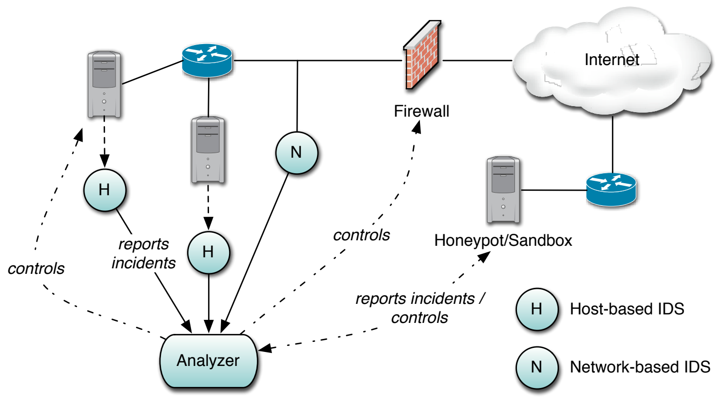

Fig. 2. Topology of an attack detection system

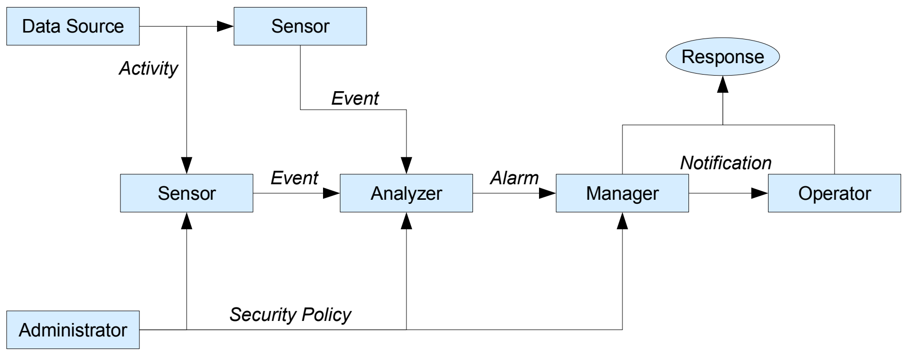

Fig. 3. Usage of different event types according to RFC 4766 [5]

from an analyzer to a manager that an event of interest has been detected. An alert typically contains information about the unusual activity that was detected, as well as the specifics of the occurrence. Relationships between different event types and their source and destination are depicted in Figure 3.

Attack messages may be categorized as a subclass of general events. In the scope of this report, the term *attack* covers all sorts of incidents in a network which may be assigned to the area of security. Preparatory actions for an actual attack, e.g. a portscan, are also described as an attack.

2.3 Meta data

Besides normal events, which describe a single event or alarm as observed by an attack detection tool, meta data information plays an important role in event correlation. *Meta data* specify all additional information, which is helpful for the assessment of an event, but which is not part of the event itself or of other events. One example is the information about the location of an entity in a network. Meta data greatly help to understand the nature of an event. Often several disjointed events may be related to each other based on this type of information. Meta data also encompasses administrative knowledge or active user inputs. Examples are topological information of a network, knowledge about the administration of computer systems, or error messages from users which may be used as input for correlation.

2.4 Event Correlation

In the area of mathematics, two statistical random variables will correlate to each other, if they are dependent from each other. In the same basic sense, two events will correlate in the area of computer security, if they have a causal connection – in this case, only logical relations are important. The process of *event correlation* tries to find these connections between different events. Relating events can be joined to a combined event – we call this “meta event” – or be classified in certain categories.

The main goal of the correlation process is to identify more significant events in a potentially huge set of recorded events. To indicate which events are considered significant, each event often is assigned a priority value that can then be checked during the process of deciding whether a response is needed.

In summary, event correlation can be defined as follows: *Event correlation is a process for consolidating events to increase their information quality, while reducing the quantity of events. Meta data (such as time, location, network topology or administrative information) can be used to improve the quality of single or combined events.*

Methods for event evaluation can be roughly separated in two classes:

- *Signature-based analysis*: Pre-defined signatures and rules are applied to events. If rules match, the events will be classified or changed accordingly. The term “rule-based” may be used synonymously.
- *Anomaly detection*: The normal behavior of a system is defined by statistical methods or algorithms in the area of machine-learning. If a deviation of the norm is detected that exceeds a certain threshold, events are created to report the anomaly.

In the example of Figure 2, event correlation may be performed on the host and network based attack detection systems, as well as on the analyzer. It is also possible to respond to priority events by adapting the configuration of firewalls or other systems.

3 Requirements Analysis

The requirements analysis of event correlation in general and in the scope of a national IT early warning system is as important as the discussion and description of correlation methods themselves. Thus, we list and discuss aspects, which will have a major effect on the efficiency and behavior of event correlation systems. We roughly follow the approach described in RFC 4766 (Intrusion Detection Message Exchange Requirements) [5].

3.1 Architecture types

The basic architecture of an IT early warning system should not depend on specific methods of event correlation. Therefore, we do not assume any further requirements that exceed the needs of the reasonable operation of a correlation system. The following exemplary aspects will ensure grades of freedom for the structure and operation of the entire IT early warning system.

Event correlation should be able to work regardless of which systems, sensors, and functions for analysis are installed. We regard sensors as well as analyzing entities as functional components, which may be installed on arbitrary systems. This means that sensors and analyzers may be installed and operated separately, i.e. distributed in a network, or in a combined way without compromising the functionality of event correlation. This way of operation should not affect functionality. We will show in the study of correlation methods, that integrated systems provide technical advantages: a wider data pool may be used and feedback loops are easier to implement. The way of coupling those systems should not have any effect on the operation, e.g. a hierarchical architecture or a GRID structure comprising several distributed autonomous systems may be used.

In order to enable hierarchical event correlation, certain components need to be equipped with different roles in a network. Therefore, a system may look like a correlator *from the bottom*, whereas *from the top* it may look like an analyzer or general data supplier. Furthermore, it should be possible to use event correlation in different scenarios. This covers the degree of automation as well as the degree of reactivity of the network, e.g. its ability to automatically reconfigure firewalls without human interaction.

Finally, the fact that attacks often affect large parts of a network and not only single institutions needs to be considered. Therefore, activities in the scope of attack detection (and corresponding countermeasures) need to be combined and coordinated.

3.2 Internet threats

Before analyzing the requirements of attack detection systems, we briefly introduce currently available threats in the Internet. Basically, we try to answer the following questions: What are goals of attacks? How are attacks being performed? At first, we will present two papers that tried to estimate the threat that may emanate from malware in the Internet.

Staniford et al. [6] simulated the spreading of worms in the Internet and estimated infection times based on data gained from the worms "Code Red" I, II and "Nimda". They determined, that in their initial phases these worms spread with exponential growth – a large part of the Internet was infected within hours by these malicious applications. The authors developed concepts based on

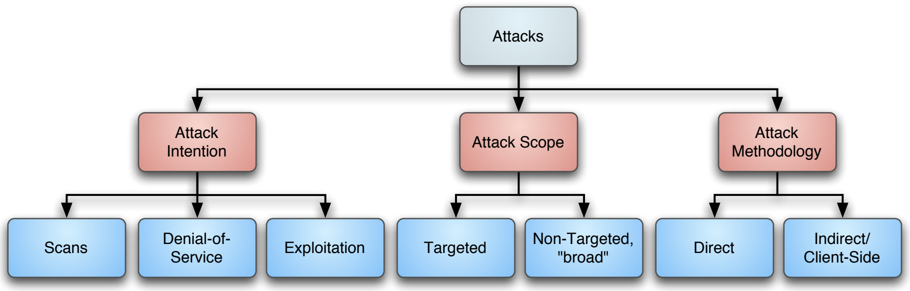

Fig. 4. Taxonomy of network-based attacks

these studies, that enable worms to improve their infection speed. In a simulated scenario, almost all vulnerable systems in the Internet were infected within 15 minutes. We need to pay attention to the fact, that all parameters of this simulation were set to optimal settings, that may never be achieved in the real Internet, e.g. hosts may have slow connectivity or networks may be protected by firewalls.

Moore et al. [7] analyzed the requirements to slow down or stop the spreading of worms in the Internet. They assumed, that either each end system in the network or central nodes like routers from providers implement a firewall. Then it is possible to block connections from other systems or block access for malicious code that is detected by a network traffic signature. The authors evaluated the parameters time for reaction, containment strategy (block single system or malicious code) and application scenario (firewall install locally or at ISP) by simulation. They concluded that, if firewalls block infected systems within 20 minutes or directly block all malicious traffic using a specific signature within 2 hours, at most 1% of all systems in the network will be infected after 24 hours incubation time.

There are several types of threats and types of attack in networks that need to be detected by attack detection systems and feature completely different properties. An overview is depicted in Figure 4.

Attack intention Attacks may be performed with different intentions and results for the attacker and the targeted host, ranging from gathering data to compromising hosts. Considering this property, attacks may be distinguished in the following ways:

– *Scans:* Attackers perform scans to get information about systems and topology of a target network. Most of the time this task is performed before the actual attack. Therefore, detected scans may indicate upcoming attacks and it may be possible to react appropriately before the attack.
– *Denial of Service (DoS):* DoS describes attacks, which impair or even interrupt services of the targeted systems. Usually, this is achieved by overloading the systems. Often this method includes a large quantity of packets / connections, and these anomalies are easily recognizable by flow-based anomaly

detection methods. Nevertheless, sometimes systems feature vulnerabilities that enable attackers to break the system more efficiently with less transferred data, which makes these types of attacks as hard to detect as the exploitation of security vulnerabilities.
A passive method to count the amount of DoS attacks in the Internet was developed by Moore et al. [8]. They relied the fact, that DoS attacks are often performed with spoofed source addresses and so reply packets from the targeted systems are sent into non-involved networks. By monitoring and analyzing of those reply packets, it is possible to roughly estimate the number of DoS attacks. In the analyzed network, they estimated 13.000 DoS attacks in one week.
- *Exploitation of security vulnerabilities:* In this case, attackers exploit the availability of a vulnerability of a system. These attacks are often only detectable by analyzing the payload of network packets or evaluation of log files, as they often do not have any easily identifiable attributes which may be detected by anomaly-based methods.

Scope of attack Attacks may also differentiated according the number of source and destination hosts. DoS attacks sometimes also originate from multiple compromised hosts, but most of the time, only a single host is used as origin of the attack. Depending on the number of destination hosts for an attack, the following types may be distinguished:

- *Targeted attacks:* Only a single host is used as an attack's target. These attacks are generally hard to detect by anomaly-based techniques, as only little traffic is generated by them.
- *Non-targeted, "broad" attacks:* The attacks target multiple hosts or even complete networks at once. Even if only a few attack packets are generated per host, the overall attack traffic in the network will become visible if entire networks are monitored. Thus, this kind of attack can be more easily detected compared to targeted attacks.

Attack methodology Recently, methodologies of attacks have changed slightly in terms of how attackers try to access their targets:

- *Direct attacks:* Connections are directly established from the attacking host to the target. Thus, firewalls that are placed between target and attacker may prevent these attacks.
- *Indirect / client-side attacks:* In the last few years, direct attacks have declined in popularity. Targets are no longer directly attacked. Instead, attacks rely on users to access some malicious content in the Internet. Due to vulnerabilities in client software, accessed content may exploit the local client computer. At the moment, mostly two types of applications are exploited: mail clients by sending spam mails to unsuspecting users who tend to open attached malware, or web browsers by directing users to specially crafted web sites that contain malicious code. Both techniques often rely on social

engineering techniques to motivate the users to access certain infected sites in the Internet.

3.3 Technical requirements

The goal of this section is to summarize technical and organizational requirements for the data exchange between sensors, attack detection systems, and event correlation.

Formats and interfaces Requirements for standardized data exchange apply to used protocols and formats in the communication between separate systems. In the scope of the Internet Engineering Task Force (IETF) standardization efforts, adequate candidate protocols have been introduced in the last years.

– Standardized formats – The following formats belong to data and configuration exchange: IDMEF for created events and alarms, Netconf [9] for configuration data, as well as IPFIX [10] and PSAMP [11] for exchange of statistical information about flows or raw packet data.
– Interface definitions – Adequate interfaces for interaction of different institutions are needed. This aspect particularly includes administrative and security-related qualities. In the scope of authentication, authorization and accounting (AAA), solutions for secure data and configuration exchange are suggested in the context of Diameter [12].

Data quantity and quality Besides the requirement of standardized data transfer, all subsystems from the network monitoring to the event correlation layer of an IT early warning system make varying assumptions and demands of the amount of processed data and its significance and quality.

– Monitoring – Extreme data quantities are produced and processed in the monitoring layer. This information in its raw form has only limited expressiveness. An intermediate analysis of the data is reasonable before it is fed into an comprehensive and distributed event correlation system. Configuration of the network monitors is also critical for the overall system, as well as the option to adaptively configure the operational parameters is desirable.
– Pre-analysis – Attack detection systems like IDSs are typically being used in the first analysis step. These systems use adequate detection algorithms for a qualitative improvement of transferred event data and thus for a quantitative reduction of the amount of data to be processed. However, local attack detection cannot detect some attacks (or events). These attacks are lost in the mass of event data. Thus, more comprehensive correlation methods are inevitable.
– Event correlation – Event correlation typically encompasses multiple instances. So, especially in this area, there is a need for standardized data exchange. Furthermore, feedback to IDSs and monitoring systems is needed to increase the overall detection quality. Event correlation should be able to use meta data to increase the informational value of events.

Minimum prerequisites on event data During the processing of events, a number of minimum requirements need to be fulfilled by the data itself. According to RFC 4766 [5], the following six requirements or prerequisites for data contents apply:

– Monitored event (data) – The actual event has to be described adequately and particularly unambiguously.
– Event identification (name space) – The assignment of an event to a global valid category must be possible. To achieve that, (not yet standardized) naming conventions are needed.
– Background information – Meta data about the created of an event are essential for adequate assessment of an event. Included are e.g. current system configuration, sampling rates, filter, position (topology), administrative limitation, information about the sensor, informations about the analyzer (attack detection system) and so on.
– Event information (source, destination) – In addition to the characterization of an event, information about the described attack's destination is needed. This can include the IP address or the destination domain.
– Impact – The impact describes the importance of an event. Knowledge of the event source, e.g. whether it was evaluated manually, half or fully automatically, may impact further event correlation. Often, the term *degree of confidence* is used in this context.
– Proposal of countermeasures – Finally, event-creating systems may offer proposals for countermeasures, which may either be adopted or provide a basis for further actions.

3.4 Operational boundary conditions

Finally, operational aspects need to be regarded besides the technical requirements. Such aspects may essentially impact the operation of event correlation methods. The following boundary conditions have to be detected and integrated with appropriate configuration parameters by the operators of sub systems as well as the deployed correlation systems:

– Quality and quantity of expected sensor data – What precise sensors are in use, how are those linked to the analysis systems?
– Required performance for data analysis – Depending on the information about network topology and knowledge about sensors, the required computational power for analysis and correlation methods need to be estimated.
– Storage requirement – How much and how long does data need to be cached? Possibly legal aspects need to be regarded here.
– Time synchronization – A common time basis is important for the deployment of distributed architectures.
– Relevance of original data – Does the event correlation probably also need original data (packet or flow traces)? Usually this type of data is not included in abstract events any more.

– Ability for coalition and multi-client capability – What parties may work simultaneously on what parts of the original data? Thereof partitioning and role management are essential.

3.5 Objectives for efficient event correlation

Correlation systems for attack detection and security related information processing generally operate on high data volumes at the base, the sensor level, and try to filter out important security-related information to reduce data rates and increase quality of generated events. More specifically, the following list describes goals and possible features of these systems:

– *Reduction of event quantity:* Many events are generated in multiple locations especially on lower layers of attack detection systems, like in the case of using several sensor nodes that monitor partly identical traffic. These events need to be merged as soon as possible to reduce the degree of redundancy, i.e. the overhead. Load on analyzers located downstream is reduced by this method, as well as the evaluation by security administrators. If many irrelevant events are reported by an attack detection system, especially the review by human operators will suffer.
– *Identification of the problem's cause:* Many activities in a network cause several events in multiple attack detection systems to be reported simultaneously. One of the main responsibilities of a correlation system is to find the cause of events by detecting dependencies between different events, also including the quality of single events into the calculations. Then the result is the original event that contains a summary and analysis of received events.
– *Short response time:* Available methods for correlation support offline and online analysis. These terms describe the execution time of algorithms: *Offline analysis* processes are executed at pre-defined regular times, like once a day. Usually a data store like a SQL-database is used for this task. Events that are received between the execution times, are cached in a database until they are processed by the next process run. On the contrary, *online analysis*, also called *real-time analysis*, continuously receives events and processes input data immediately. Corresponding algorithms are also described as *streaming algorithms*. Results of those methods may also be generated continuously, or at pre-defined intervals. To achieve minimal response times, online analysis methods are a better choice, as no waiting times caused by periodic execution are involved in the process.
– *Observation of data privacy:* From an academic or even from an operators perspective, the basic principle in network security is to collect as much information as possible, as more information means more detailed analysis that produces better results. The quantity of information can be increased, if more institutions join an attack detection system and provide sensors in their own networks or hosts. But this procedure introduces the problem of data privacy: often sensitive data is monitored by sensors that is not allowed to leave the institution's boundary. So sensors for attack detection also need to

observe policies regarding data privacy issues, especially if multiple organizations collaborate with each other that are not willing to exchange sensitive data.

- *Automated reaction:* Most of the time, attack detection systems report detected security incidents to human operators that may take action on this information. But sometimes it is necessary to automatically act on detected security incidents, like the automated configuration of a firewall to shut out attackers. Rules for activating automated reactions need to be considered carefully, as this process may create new vulnerabilities that can be exploited by attackers. An example of this topology is described in Section 5.3.

Human involvement in the system should be reduced as far as possible, especially in the layers where high data quantities are processed. Event correlation is one of the most important parts of the system which try to improve quality of reported data (or events in higher layers). The following aspects specifically deal with features good event correlation systems should provide:

- *Few false-positives:* Events that are generated and reported to an operator without being relevant for further analysis, or even erroneous, are called false-positives. Event correlation algorithms need to detect events, which are of no relevance and deal with them accordingly. To efficiently detect false-positives, detailed information needs to be available about the creation of events, especially the reliability of used techniques. Events from systems that have high false-positive rates should not be rated as high as events from systems with low false-positive rates. One possible method to confirm received events is the correlation of events from multiple sources - if multiple events point to the same cause, the probability of them being false-positives is lower.
- *High speed:* Computationally efficient correlation algorithms reduce hardware requirements and enable new possibilities for further correlation. Considering the algorithmic complexity, the following categories for data processing algorithms can be distinguished:
  - *1-pass:* Only one pass over incoming events needs to be performed. These algorithms are easily integrated into a system which performs online analysis. An overview of the basics of these algorithms can be found in [13].
  - *n-pass:* The algorithm needs several runs over incoming events. This type of operation results in substantial additional expenses compared to the 1-pass analysis.
  - *state-based:* Often correlation techniques manage internal states. Depending on the criteria that differentiates groups of input data, the amount of needed states varies greatly – exemplary states could be saved on a flow, IP address or even subnetwork basis. These algorithms need to pay attention to memory and computational constraints of hardware at incoming events that feature high variances in information content.

– *Reliability:* Concerning especially attacks on systems which involve high data rates for their analysis, it is very important that correlation algorithms are prepared for possible overloads and are able to react adequately without loosing important information. Examples for attacks producing high data volumes are DoS attacks or mass spreadings of worms.

4 Event Correlation – A Taxonomy

We studied various kinds of event correlation techniques. Many of the used algorithms have been developed in the context of attack and intrusion detection. In this section, we outline the relations and the dependencies between the components of attack detection and event correlation system. This overview represents a comprehensive taxonomy of event correlation techniques. In order to provide a better understanding and as a reference guide, the next section associates algorithms presented in the literature to the categories discussed in the following taxonomy.

In general, event correlation methods can be distinguished according a number of orthogonal criteria:

– *Correlation techniques* as discussed in Section 4.1 classify the used algorithms for signature matching or anomaly detection.
– *Layers of event correlation* are presented in Section 4.2. As network sensors, attack detection, and event correlation systems will usually be employed in a hierarchy, the properties of the different layers need to be identified.
– *Data formats* of received packets, flows, and events must be clearly identified. We outline typical formats in Section 4.3.

Figure 5 show the taxonomy of techniques used in components and the data flow of a comprehensive attack detection system.

4.1 Correlation techniques

Correlation techniques can rely on completely different underlying methodologies. Specifically, signature-based and anomaly-based techniques need to be differentiated.

Signature-based techniques Signature-based correlation methods use events and pre-defined signatures as input data. The signature represents a kind of filter which is applied to all incoming events. If the signature matches (parts of) the received data, some pre-determined action is performed, like the generation of an event which notifies higher layers in the attack detection system about the detection. Sometimes the combination of the signature/filter and the commands that are executed when a match occurred, is called a rule.

Generally, signature-based systems tend to create fewer false positives compared to anomaly-based methods: most of the time, signatures try to identify

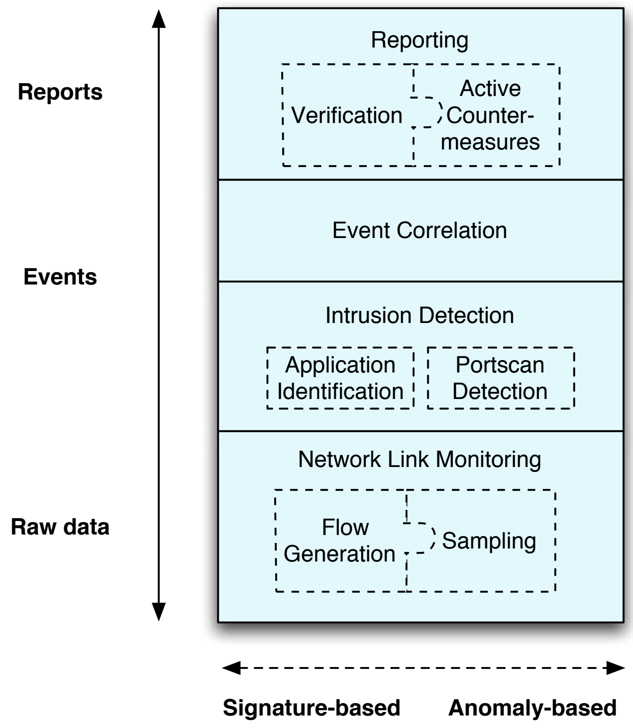

Fig. 5. Taxonomy of techniques used in an attack detection system

a specific type of data inside the input stream, like a certain type of exploit in payload-based IDSs. Of course, the detection quality still depends on the process how these signatures were created: often, IDSs like Snort are equipped with manually created signatures. These signatures offer a high detection quality with high true positives and few false negatives.

Specific attacks can be easily detected by adding a specialized rule. When outdated rules are no longer needed for correlation analysis, these rules can be removed from the system. Thus, correlation algorithms may be flexibly adapted to new requirements. This fact especially helps during further analysis by security administrators who want to investigate the details of an incident. For this task, new rules can be included in the correlation system without loosing functionality.

The reliability of event correlation strongly depends on the pre-defined rules. Badly specified rules may degrade the effectiveness of the algorithm substantially. Additionally, only attacks can be detected that are covered by signatures. When a new attack is discovered, a new signature needs to be defined and included in the system. Only after this process, the attack may be detected and reported by a signature-based correlation algorithm. So the used set of signature needs to be updated on a regular basis. The network-based IDS Snort for example offers functions to update its rule-set regularly by fetching new rules from Internet websites.

The computational efficiency of signature-based detection methods often strongly relates to the amount of signatures that are present in the system, so it may be necessary that this amount needs to be limited. In the case of Snort, old signatures that do not occur in traffic any more are replaced by newer signatures.

Anomaly-based techniques Anomaly-based techniques are used to identify unusual system behavior. Such unusual behavior can be detected based on two different methods. First, *specification-based* solutions continuously analyze the system's behavior and compare it to a given parameter range. If this scope is violated, an event will be generated that points to a possible attack. The parameter scope of normal system operation is either generated manually or semi-automatically. Secondly, *data-mining-based* methods can be employed, which have been developed for model extraction from big databases [14]. Usually, in a first step the correlation algorithm is trained for a system in which only "normal" activities occur. In general, the definition of "normal" activities in a system is also a hard problem, not to mention generating or monitoring traffic without any malicious content for optimal training. A model is built during this training process that describes the normal state of the system. In a second step, the actual system is monitored and constantly compared to the generated model. If the difference between the trained model and the actual behavior of the system is bigger than a certain threshold, an event will be created that reports the anomalous behavior. A multitude of algorithms are available for their analysis, examples are algorithms from the area of statistics, pattern recognition, learning-based systems and databases [15].

Anomaly-based event correlation features one important advantage compared to rule-based systems: the anomalous behavior of a monitored system can be detected. So, also unknown attacks can be identified. When one of these events is confirmed to indicate a relevant incident, it can be used to trigger other systems that generate signatures for rule-based correlation systems and so increase the detection rate of these systems.

Often, the behavior of systems is analyzed statistically. Therefore, anomaly-based attack detection can be circumvented by "slow" attacks, that are not noticeable in relation to the system's normal behavior and, thus, do not trigger an anomaly-based detection system. One example for this kind of attack is a slow network scan which is executed over several days. This attack changes the behavior of a system only marginally, the threshold defined in the attack detection system is not reached and no event is generated.

As it is hard to define the system's behavior unambiguously as well as to configure the sensitivity of anomaly detection systems in a proper way, the amount of false-positives may be quite high depending on the scenario. This needs to be taken into account for developing an event correlation architecture.

Differentiation of normal / anomalous behavior Furthermore, correlation algorithms may be differentiated according the detected traffic. Either, they detect normal behavior, i.e. the algorithm is trained with legitimate traffic. This enables the algorithm to identify normal input data, and if observed input data pattern differ from the normal trained set, the anomaly will be reported as an event. As the algorithm does only know that it detected an anomaly, it is not able to identify the anomaly or provide a specific description, e.g. the name of an exploit.

On the other hand, the training phase may be geared towards the detection of anomalous patterns of a certain type. If then normal, non-anomalous input data is observed, the trained pattern will not match to the input data. When input data converges to a pattern of anomalous behavior, like an attack that was trained beforehand, the algorithm reports an event including the description of the already trained and specified anomalous behavior.

Both detection variants may be applied to either signature-based or anomaly-based techniques. If normal data was used as training data, events will be generated when the trained pattern does not match any more. If anomalous data was used for training, events will be created when the trained pattern matches.

4.2 Layers of event correlation

Several layers for event generation and analysis can be defined in a distributed system for early warning. Figure 6 differentiates between three layers: *raw data layer*, *event layer*, and *report layer*. Each layer tries to filter out as many non-relevant events as possible to correlate relevant ones and to aggregate them appropriately. Data from different network sensor sources is gathered at the *raw data layer*. Here, this data is processed and the results are then forwarded to the *event layer*. At this layer, IDSs and event correlators are used to categorize and prioritize input data, so that relevant data is separated from non-relevant data. The output is then forwarded to the *report layer*. In this final stage, summaries of the monitored data can be displayed and post-processed. All correlation techniques described before can be used in each layer.

Raw data layer This layer consists mainly of network sensors, which receive raw data and conduct a first analysis. Network-based monitoring systems directly collect data packets from directly connected networks. They may specialize on full packet payload or IP header data only. Other sources for monitoring systems in this layer may be log data, as many network-based applications produce protocols and write them in log files. Sources for log files are e.g. central virus scanners at mail servers, web proxies or firewalls. The gained log data may be used in central attack detection systems or directly in event correlation systems to improve the quality of the results.

Methods for correlation in this layer focus on efficiency and high aggregation ratios because of high data rates. More elaborated correlation techniques usually should be moved to upper layers in the attack detection system, where lower data rates need to be processed.

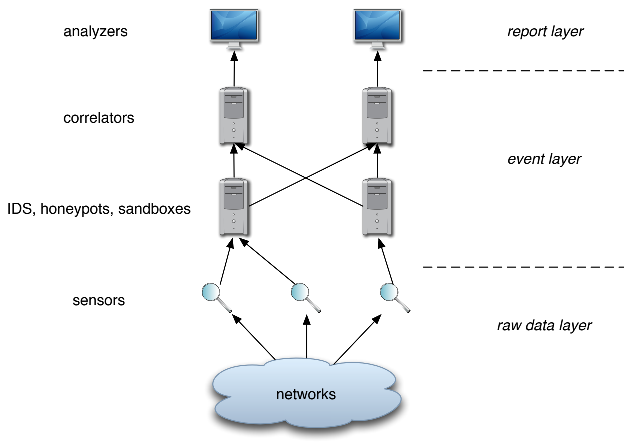

Fig. 6. Hierarchical correlation and aggregation of events

Event layer Network sensors and IDSs may produce a large quantity of events. Interestingly, most of these events may be of no interest for further evaluation. According to Kruegel et al. [16], events reported from these systems may be classified as follows:

– *True-positives* – The event sent from the sensor has really happened and is relevant for further processing.
– *Non-relevant positives* – The sensor has correctly identified and reported an incident, but which is not relevant for further evaluation. For example, this class of events contains attacks on systems, which are not vulnerable to the observed attack and thus the attack would not be successful.
– *False-positives* – An incident was incorrectly detected and identified by a sensor. The event is irrelevant for further analysis.

Now, the goal of event correlation is to filter out all events of the second and third class. Depending on the security policy and goal of the analyzing system, it must be decided, which events have to be classified as non-relevant positives and, thus, not to be further processed. Event correlators are used to enrich events of the first class with relevant metadata and remove redundant information. Some events can be processed fully automatically.

Furthermore, honeynets [17] and sandboxes [18] may also be integrated in the event layer. These systems offer the possibility to reconfigure attack detection systems in real-time and improve the quality of results of the overall system, as they are able to detect new attacks, to analyze and evaluate them automatically.

Report layer Specialized applications adequately visualize the results of event correlation at the report layer. Based on the final analysis – which may also include automated response strategies and validation techniques – the attacks are identified and countermeasures can be selected. Functions of this application should include the possibility to access lower layers of the analyzing system like attack detection systems to request details about certain events.

Another area in the report layer is manual evaluation of statistical data like data transfer rates, average packet sizes or top-N lists of sub networks having the highest transferred data rates. This way, incidents can be covered which may not be detected using fully automated methods.

4.3 Data formats

Input data is essential for effective event correlation. In this section, we will concentrate on the different data formats and types that are used in each of the previously described event correlation layers. In all the discussed layers, privacy plays an important role. We will cover this separately in the last part of this section.

Packet payload All monitored information is provided by network sensors with direct network access. This can be a dedicated PC system using the Packet Capturing (PCAP) library or an IP router or switch. Frequently, also IDSs provide capabilities to directly access the network. Due to the high level of detail, only low data rates may be processed without loss. For transmission of packet data including full payload, an extension to the IPFIX protocol called PSAMP [11, 19] has been developed.

Flow data Flow or "netflow" data only provides header and statistics information of a set of packets sharing some common properties. This form of aggregation is widely used in the Internet and is directly supported by many commercial routers. Aggregated flows are transferred to so called collectors for further analysis. Typicall, the IPFIX protocol [10, 20] or its predecessor Netflow.v9 [21] is used. IPFIX was specified by the IETF and enables efficient transfer of aggregated flow data – this may, it is possible to even monitor gigabit networks.

Flow data may provide all the information that is necessary for flow-based application identification or attack detection. For example, counting and evaluating the number of IP packets with the `SYN` flag helps to determine if a DoS attack or a scan operation is performed [22–24]. Of course, events generated by such anomaly-based techniques may also be false-positives.

Event reports IDSs generate events that contain information about a possible attack, often including an attack category or the detailed description of the attack. Events generated by these detection systems have varying levels of relevance, depending on the evaluation method, origin and type of input data. Anomaly-based systems can only decide with a certain possibility, if the detected incident is an attack. Results of signature-based systems primarily depend on the quality of the used signatures. Due to this variance in confidence, events are

```
                     IDMEF-Message
                           /_\
                            |
    +-------------------+-------------+
    |                                 |
+-------+   +--------------+    +-----------+   +----------------+
| Alert |<>-|   Analyzer   |    | Heartbeat |<>-|    Analyzer    |
+-------+   +--------------+    +-----------+   +----------------+
|       |   +--------------+    |           |   +----------------+
|       |<>-|  CreateTime  |    |           |<>-|   CreateTime   |
|       |   +--------------+    |           |   +----------------+
|       |   +--------------+    |           |   +----------------+
|       |<>-|  DetectTime  |    |           |<>-| AdditionalData |
|       |   +--------------+    +-----------+   +----------------+
|       |   +--------------+
|       |<>-| AnalyzerTime |
|       |   +--------------+
|       |   +--------+   +----------+
|       |<>-| Source |<>-|   Node   |
|       |   +--------+   +----------+
|       |   |        |   +----------+
|       |   |        |<>-|   User   |
|       |   |        |   +----------+
|       |   |        |   +----------+
|       |   |        |<>-| Process  |
|       |   |        |   +----------+
|       |   |        |   +----------+
|       |   |        |<>-| Service  |
|       |   +--------+   +----------+
|       |   +--------+   +----------+
|       |<>-| Target |<>-|   Node   |
|       |   +--------+   +----------+
|       |   |        |   +----------+
|       |   |        |<>-|   User   |
|       |   |        |   +----------+
|       |   |        |   +----------+
|       |   |        |<>-| Process  |
|       |   |        |   +----------+
|       |   |        |   +----------+       +----------------+
|       |   |        |<>-| Service  |       +----------------+
|       |   |        |   +----------+  +----| Classification |
|       |   |        |   +----------+  |    +----------------+
|       |   |        |<>-|   File   |  |    +----------------+
|       |   +--------+   +----------+  | +--|   Assessment   |
|       |<>----------------------------+ |  +----------------+
|       |<>------------------------------+  +----------------+
|       |<>----------------------------------| AdditionalData |
+-------+                                   +----------------+
```

Fig. 7. Overview of the IDMEF data model

usually marked with an alarm and confidence level. The alarm level describes the severity of the detected event – for example portscans are not treated as important as detected intrusions into hosts. Depending on the labels, events may be evaluated and processed in different ways.

The XML-based data format IDMEF [25] has been specified to represent single events. The protocol is designed to unify data formats for communication between different IDSs. Main focus during development of IDMEF was the support for heterogeneous information that is generated by different sensors. The content of events varies depending on the type of sensor. Furthermore the kind of information reported by those systems strongly relates to the monitored media. The data model of IDMEF is prepared for this application with its object-oriented approach and may be adapted flexibly. Extensions are defined by derivation or association of new classes. The flexibility of IDMEF causes one of the main problems when using this format: frequently, IDSs use (proprietary) extensions of IDMEF. This contradicts the main goal of IDMEF to unify data formats.

```
<IDMEF-Message version="1.0">
  <Alert ident=bc123456789">
    <Analyzer analyzerid="hq-dmz-analyzer62">
      <Node category="dns">
        <location>Headquarters Web Server</location>
        <name>analyzer62.example.com</name>
      </Node>
    </Analyzer>
    <CreateTime ntpstamp="0xbc72b2b4.0x00000000">
      2000-03-09T15:31:00-08:00
    </CreateTime>
    <Source ident=bc01">
      <Node ident=bc01-01">
        <Address ident=bc01-02" category="ipv4-addr">
          <address>192.0.2.200</address>
        </Address>
      </Node>
    </Source>
    <Target ident="def01">
      <Node ident="def01-01" category="dns">
        <name>www.example.com</name>
        <Address ident="def01-02" category="ipv4-addr">
          <address>192.0.2.50</address>
        </Address>
      </Node>
      <Service ident="def01-03">
        <portlist>5-25,37,42,43,53,69-119,123-514</portlist>
      </Service>
    </Target>
    <Classification origin="vendor-specific">
      <name>portscan</name>
      <url>http://www.vendor.com/portscan</url>
    </Classification>
  </Alert>
</IDMEF-Message>
```

Fig. 8. Example of an IDMEF message describing a portscan

Figure 7 depicts the IDMEF data model. It lists the components an event may be composed of and how those are further divided. Figure 8 shows a small example for an IDMEF message describing a portscan.

Transfer of IDMEF formatted events may be performed using different protocols. Intrusion Detection EXchange Protocol (IDXP) [26] is a connection-oriented protocol on the application layer. It is designed for data exchange between analyzer and manager of an attack detection system and realizes the transfer of IDMEF messages. Transfer of text or binary data is also possible. IDXP is based on Blocks Extensible Exchange Protocol (BEEP) [27], which represents a framework for defining protocols at the application layer. Often, IDMEF messages are transferred using Simple Object Access Protocol (SOAP) and HTTP as usage of IDXP and BEEP requires high implementation efforts in practice.

Metadata Data which is not part of reported events but originate from other sources is called metadata. An example is the model of a network topology. Information about the location of systems and routers are linked to the respective events. Additionally, it is possible to create a database about the vulnerability of hosts to be able to automatically determine which attacks were successful. Non-technical metadata includes for example contractual information, data privacy policies, or information about the administrators and end-users.

The generation of metadata may be performed in a passive or active way. Passive generation is performed manually or in specified time intervals, which are independent from the reception time of events. Yemini et al. [28] show an example for this method. On the contrary, active generation is performed as reaction to a received event to gather additional information.

Data privacy Data privacy needs to be regarded in any system handling private information. The exchange of IP packets, which is observed by attack detection systems belongs to this class of systems. Sensors for security-related systems almost always collect information that is sensitive for either a single people (because of privacy regulations) or for the institution itself (data that can compromise confidential information).

This problem is a separate area of research because of its opposing goal to security analysis: optimized detection of attacks requires as much information as possible, whereas data privacy issues limit the available information. The issue is further complicated, as practical experience and research turned out that removing sensitive information is a complex problem that is not easily achieved. The use of anonymization or pseudonymization techniques often results in obfuscated data that still contains more information than intended [29]. The theory of inference control helps restricting the possibilities to extract additional information from datasets: using logical conclusions, all data is removed that may lead to unwanted conclusions. A good overview is presented in [30], which copes with Query Set Size Control, Cell Suppression and the Lattice Model.

5 Correlation algorithms

In this section, we outline selected algorithms that relate to the context of early warning systems in general and event correlation in particular. According to the taxonomy presented in the previous section, we group the algorithms to the different layers of event correlation. Table 1 summarizes all important correlation algorithms including references.

5.1 Raw data layer

Event correlation not only happens at the “abstract” event level, but also at the lower layer based on inspections of the network traffic. Raw packet data from the network layer is usually not forwarded to event correlation systems because the data rate would be too high. Specialized correlation systems on the lower layer need to deal with this problem. In the following, we present correlation and evaluation algorithms that are based on headers and / or payload information of monitored network packets.

Raw data layer		
Packet sampling	Methods for effective aggregation of packets in flows	[31–35]
Probabilistic analysis	Generation of meaningful statistics from flow data	[36–38]
Attack detection	Detection of anomalous data streams	[2, 39–42]
Detection of portscans	Detection of slow and fast portscans	[23, 24, 43]
Application identification	Flow-based identification of application	[44–46]
Payload analysis	Analysis of packet payload for attacks or anomalies	[47–55]

Event layer		
Local correlation	Correlation techniques, that are executed locally	[28, 56–59]
Distributed correlation	Correlation techniques that operate on multiple systems and cooperate	[60–67]
Data privacy compliance	Techniques that combine data obfuscation with correlation	[68–73]

Report layer		
Active countermeasures	Possible automatic countermeasures to events / attacks	[7, 74–77]
Event verification	Techniques for improvement of event confidence	[16]

Table 1. Overview of essential correlation algorithms and corresponding references

Packet sampling In order to cope with increasing traffic rates, research effort has been invested on how to reduce the amount of data to be processed at the raw data layer and packet sampling techniques were introduced to solve this problem. Depending on various statistical methods, these techniques include only a fraction of transferred network packets in the monitoring data. Several suggestions are available that try to optimize the selection of packets so that as few information as possible is lost by the sampling process. Other methods rely on data reconstruction algorithms that infer original data statistics from the available sampled data.

A simple example for statistical packet sampling is the "n out of m" algorithm that includes n packets in the monitored data out of m packets transferred over the network. Depending on available hardware, sometimes even 1 out of 100 packets are sampled from network traffic of 10 Gbit/s data rates or higher.

Estan et al. [31] published one of the essential papers in this topic: They improved hitherto existing methods to process sampled flow data: formerly, a sampling rate and algorithm was configured by an operator once and was never changed again afterwards. Due to this static configuration, changes in incoming data rates might have caused a system to become overloaded and loose data.

This is why the authors developed a dynamic technique that adapts the traffic sampling rate according to current resource consumption in the monitoring system. Aggregation of flows was performed using hash tables, so when the sampling rate was changed, the amount of already aggregated, i.e. cached, flows also needed to be changed to maintain statistical properties of the traffic. The authors suggested an efficient algorithm to solve that problem which changed the size of the hash table in use and its contents.

Another work by Hu et al. [32] tries to solve the problem of flow aggregator overload by detailed analysis of already collected data. Especially DoS attacks often involve high bandwidth operations in networks. An example would be SYN-attacks where source addresses in IP packets are spoofed and thus are unimportant for further analysis. In this case, it is sensible to aggregate all packets having the same destination IP to one data record, i.e. flow by omitting the source addresses. This approach uses less memory and unburdens the monitoring system by excluding irrelevant information from the beginning. The important information, in this case the target system of the attack and its extent, is still available in the more aggregated version.

A closer look at the interpretation of sampled data is necessary: for many evaluations, statistical properties of the original traffic are needed, so methods for inferring these properties from sampled data were introduced. Trivial approaches like simple multiplication of flow data rates with the sampling rate $\frac{m}{n}$ are inferior, as flows with high packet rates are captured with high possibility after sampling, whereas short flows are often missed and never recorded.

More successful techniques were covered by Duffield et al. [33, 34]. They preferred packets with large sizes during sampling for data rate estimations. Another interesting approach was proposed by Hohn et al. [35] which switched from packet sampling to flow sampling. Only each n-th flow, instead of each n-th packet, was sampled by their monitor. Resulting properties of flow-based statistics were improved by their method.

Flow-based analysis Flow aggregation provides means for clearly reducing the amount of monitoring data in high speed networks. Nevertheless, subsequent analysis becomes complicated because much information, e.g. the packet payload, is completely removed. In the following, we first focus on efficient probabilistic data analysis, and then we outline methods that detect attacks based on anomaly-based algorithms.

Probabilistic data analysis There are several techniques available that efficiently process high data volumes and perform statistical operations on this data. To provide a high level of efficiency, these techniques rely on probabilistic methods to calculate their results.

Estan et al. [36] and Keys et al. [37] propose efficient ways how to count active flows in high speed networks. Their results can be used for example to detect DoS attacks. Statistics produced by those techniques contain the most used ports and IP addresses. Kumar et al. [38] introduce an algorithm to efficiently measure networks and produce statistics. Their proposal is based on the use of Bloom filters, that were introduced by Bloom in 1970 [78] and detect duplicate input data in constant time using probabilistic methods.

Attack detection So-called superspreaders can be detected by an algorithm proposed by Venkataraman et al. [39]. Superspreaders are worms that rapidly spread in the Internet by using remote exploits of security vulnerabilities in operating systems or applications. Their method tries to find hosts, that contact a high number of other hosts in a certain time. Jung [2] introduces a similar technique. It tries to detect hosts that contact other hosts with an unusual high rate. Main goal of both methods is to detect malicious applications that spread in a network with a trial-and-error technique to find other vulnerable hosts.

Some attacks in the Internet also rely on diverging from protocol specifications. Analyzing correct specification-adhering behavior usually requires high computational performance, as conformance checking often is only possible by performing state-based inspection. States need to be preserved for each connection, so these approaches have extreme computational and memory-related costs. An example of these proposals is described by Sekar et al. [40], who detect non-conformance to the TCP/IP protocol by state-based analysis.

Toth et al. [41] concentrate in their proposal to attributes of an ongoing attack that are almost never seen in normal traffic: If a host was compromised by malware or a hacker, often they try to compromise more hosts originating from the first compromised host. This process can be seen in network traffic by detecting connection chains: host A initiates a connection to host B, host B connects to host C and so on.

Pure anomaly-based analysis that primarily targets good visualization of monitored data is covered in [42]. The technique called *Traffic Clustering* processed statistical data. Basis for the algorithm was recorded flow data from a network monitor. Using a graph-based method, groups of hosts like subnets can be found which are conspicuous regarding a pre-defined attribute, e.g. the transferred amount of data. The results are summarized in daily reports which then contain clusters of hosts whose attributes have changed in a conspicuous way in relation to the previous day and the overall volume of the attribute. The authors show the advantages of this method to standard top-N lists. Their implementation is to be executed off-line and becomes inefficient if multiple attributes are included in the analysis.

Portscan detection Detection of portscans in networks is a large scientific research topic in the area of attack detection systems. An overview is given in [24]. Often the rate of initiated connections including information whether connections were established successfully is included into the evaluation. The slower portscans are performed, the higher is the computational cost to detect those portscans. Staniford et al. [24] suggest a method that differently weight probed ports depending on how commonly those ports are used. Port 80 for webservers would have a low priority, as it is one of the most used ports in the Internet, whereas higher ports above 1024 are much less commonly used and would receive a higher priority.

Horizontal portscans can be efficiently detected by the technique suggested by Jung [23]. Their method even allows to detect slow portscans that persist for hours. For each initiated connection, the source IP address is stored in a table and depending on the number of failed connections to different hosts, these IP addresses are rated as either malign or benign. It is currently one of the most promising approaches to detect horizontal portscans. Weaver et al. [43] increase the efficiency of Jung's algorithm, implement it on fast hardware and complement it with a counter-measure that blocks hosts detected to perform portscans.

All presented algorithms only detect portscans originating from single hosts. If a distributed scan is started, e.g. by a botnet including thousands of hosts and each system only performs a small part of the overall portscan, all the presented methods may fail to detect the attack.

Application identification Basing on monitored flow data that contains only IP header information, it is very difficult to exactly determine used protocols. Many approaches use methods of machine-learning to train specific protocols, like accesses to web servers, to recognize those protocols in a monitored network [44–46]. Unfortunately, most pure machine-learning methods feature a high false-positive rate, so results only provide a rough overview of the monitored traffic. This effect is caused by the fact that many protocols share common attributes like average connection length or similar traffic patterns. Machine-learning methods do not compensate this effect: often it is not clearly understood which properties of training data are used for detection. The methods often specialize on some application-unrelated features of the trained data and trained signatures produce high false-positive event rates for other networks.

Payload-based analysis Payload-based detection methods analyze the payload per packet for conspicuous features. Due to the high volume of data that needs to be processed, these techniques are computationally more expensive compared to flow-based techniques. On the other hand, of course a much finer evaluation may be performed, as more data is available to the algorithms.

One of the basic examples in the field of payload-based analysis is the application Snort, which was developed by Roesch et al. [47]. The application uses a ruleset containing signatures for many types of attacks compares all packets monitored in a network with those signatures. If a match is found, an event will be generated and reported by various means. It is currently one of the most wide-spread IDS that is freely available in the Internet. Today, community support contributes to its success, as many signatures are already available and still constantly created. Performance improvements for implementation on Field Programmable Gate Array (FPGA) hardware were suggested by Yusuf et al. [48].

Paxson et al. [49] introduced a very flexible version of a signature-based monitoring system called Bro. It contains an "event engine", which reduces monitored network data to abstract events and automatically follows connection states.

Compared to Snort, Bro is able to process a much more sophisticated filter language which, among others, supports variables and functions.

Both Snort and Bro only support detection of manually generated signatures, but in scientific research many other approaches based on machine-learning and automatic signature generation are suggested. Those approaches usually do not produce as good results as detection systems using manually crafted signatures, but nevertheless they can be combined in a detection system with other methods and results can be correlated to achieve better detection rates. One of the main advantages of anomaly-based algorithms is the detection of previously unknown attacks and threats.

One of the approaches called PAYL has been presented by Wang et al. [50]. In this publication, a good overview of requirements for anomaly detection methods is given. Their algorithm detects anomalies in network traffic by comparing payload 1-grams with trained vectors. Bolzoni et al. [51] improve the approach by prepending a Self-Organizing Map (SOM) [52] to the anomaly detection process. Detection of regularly repeated strings is performed by Singh et al. [53] and Kim et al. [54]. Repeated strings can be monitored in network traffic, e.g. for worms that try to exploit security vulnerabilities on multiple hosts. The tools automatically generate signatures from repeated strings to be able to detect them later on effortlessly.

Another interesting approach also cares for data privacy issues resulting from payload-based analysis techniques. Parekh et al. [55] compare different algorithms to detect similarities between packet payloads that are obfuscated to retain data privacy. Unfortunately data privacy is very difficult to ensure, so methods that reduce data as much as possible are best. In their paper, the authors suggest to use Bloom filters in combination with other techniques.

5.2 Event layer

In the following, we will describe correlation systems whose events are located in a more abstract layer than analysis of traffic data: messages about incidents from lower layers in the attack detection system events are processed here. Examples for these events are suspected attacks, like a notification about an ongoing portscan. One of the main goals in this layer is aggregation, also by correlation, of multiple events to one event and collecting as much information as possible. The ideal result would be the detection of the event's cause.

Local correlation One class of algorithms are correlation techniques optimized for local execution on a system and do not arrange any calculations on distributed systems. They assume, that a global view on the database is available. Nevertheless, input data for the algorithms depends on the execution type, i.e. online or offline operation.

One of the essential publications was written by Yemini et al. [28]. They describe a correlation system that tries to find the correlation of symptoms to problems by analyzing possibly occurring problems and additional information like the model of the network topology. As an example, the authors examine problems that appear in a network like the breakdown of a router. The proposed correlation method enables the extraction of original problems that cause the monitored symptoms. The authors developed a modeling framework to represent a networking architecture and assume that each problem may cause arbitrary many symptoms. The set of events that are caused by a symptom is called "code", which identifies a problem. So, correlation may be regarded as a process of "decoding" the set of monitored symptoms by finding the corresponding problem. Two steps accomplish this task: in the first step, which is called *codebook selection*, a subset of events is selected for monitoring. This subset is called *codebook*. In the succeeding decoding step the events are monitored in real-time. Incoming symptoms are assigned to a problem there, also making use probabilistic methods like the Hamming distance.

A simple tool called SEC that correlates events coming from log files has been introduced by Vaarandi et al. [56]. This host-based system that is realized by a simple perl script supports local rule-based correlation of events using a lightweight language for rules.

Vigna et al. [57] present an attack detection system that implements the STAT framework. This system evaluates incoming events using a rule-based language called STATL. It describes attacks by states and transitions between states. Emphasis was put on flexibility: the rule language can be extended and adapted by modules, and the language METASTAT allows dynamic reconfiguration and management of the framework. Various modules for network-based, host-based, log and IDMEF analysis are also provided.

A host-based attack detection platform called CRIM has been introduced by Cuppens et al. [58]. It performs grouping, merging and correlation of alarms. Incoming IDMEF events are saved as tuples in a database. Attacks are specified using the language LAMBDA that includes pre- and postconditions, attack, detection and verification scenarios each assigned to a combination of events. Then several attack specifications can be merged automatically by comparing pre- and postconditions. This enables tracing the progress of an attack within a host. Using an attack history, the system can calculate estimations, what actions may be performed by the attacker after that.

Valeur et al. [59] introduced an implementation of a real-time correlation engine on an abstract event level. It is a comprehensive approach to specifically reduce the number of false alarms, also using metadata. The engine is composed of several modules: first incoming events are normalized to use a common structure (as multiple sources for events are assumed that export events with different formats), then they are enhanced with metadata (e.g. using verification). Basic principle of the correlation algorithm is to find identical and related events (same destination, same source, etc.) and to build so-called meta-events that are hierarchically composed of single events. This structure enables later evaluation to analyze the correlation process, if events were correlated in several steps. As a last step, the engine assigns priorities while taking into account metadata and correlation dependencies.

Distributed correlation After covering correlation systems that are executed locally on a single system, we concentrate now on correlation techniques that are based on a distributed architecture to analyze and correlate events. Some techniques will cover other areas of event correlation, like the topic of fault determination for devices within a network. These techniques may probably be adapted for the security field.

According to Yegneswaran et al. [60], global interaction between different attack detection systems leads to better analysis results. The authors performed an empirical analysis of attacks in the Internet using a high number of network-based IDS logs over a time of four months. They detected a high amount of scans from worms. Furthermore, $60 - 70\%$ of all scans were horizontal. Scaled to the overall Internet, they estimated a total of 25 billion attacks per day, tendency increasing. Data from attack detection systems in small networks were not sufficient to detect most active attackers or most popular ports that are attacked. It can be concluded that for a global view on attacks, cooperation between several networks is needed.

Correlation methods can be used within networks where all systems are trusted, so no data privacy issues need to be regarded. We distinguish algorithms for this scenario from such that take privacy issues into account, so those systems are fit for deployment in untrusted networks where some systems are mistrusted.

Trusted networks By making use of a peer-to-peer based algorithm called Quicksand, Kruegel et al. [61] tried to perform a decentralized evaluation of events. A set of events per attack was specified in an ASL (attack specification language). Their process allowed to correlate events using graph-based algorithms. Parts of the dependency graphs created with ASL could be analyzed by different systems. Their architecture was defined as follows: The lowest level collects data on local host-based sensors. This data is sent to “Event Correlation Units” which correlated them using the graphs specified by ASL. “Control Units” configured the system, compiled ASL specifications to C code and then to object files and distributed them over SSL connections.

Prelude is an open source IDS, which has been developed and improved by several groups [62,79]. The goal is to realize global attack detection within a heterogeneous network containing a multitude of information sources. The project supports network and host based sensors, event aggregation, storage, visualization, correlation, and reactive measures. The sensors are similar to rule-based systems that analyze payload of packets like Snort. Additionally, bindings for popular attack detection systems like Bro or Snort are provided. These systems are configured to send IDMEF messages to managers or correlation systems providing authentication and data integrity with SSL. Systems in a higher correlation layer can be placed in a tree-like topology: Events are forwarded to the corresponding parent system to enable decentralized analysis. The correlation techniques of Prelude are limited, as only one correlator was developed that cooperates with the vulnerability scanner Nessus [80].

Gruschke et al. [63] introduced a graph-based system that used dependency graphs to process correlation information. The approach is used for failure management of devices within a network and was especially suited for extending existing management systems with event correlation. Functional dependencies have been modeled for the monitored system using dependency graphs. These graphs are parsed by a correlator to find elements that would cause numerous events when breaking down.

An application for reducing high numbers of error messages in a mobile phone network has been suggested by Froehlich et al. [64]. It is a rule-based system that uses extended local programming. Its primary use is to limit the probability of overburdened network administrators during severe failures in the infrastructure. The algorithm monitors the breach of integrity policies to diagnose networks.

Probabilistic techniques are used by Steinder et al. [65]. They developed a layered system model which related services and functions between neighboring protocol layers. A bipartite probabilistic dependency graph, which is based on the network topology, specifies the relation between different systems. Errors in the network can be found by two localizing methods that employ iterative algorithms for Bayesian networks [66].

Untrusted networks Correlation systems in untrusted networks, like the Internet, where different institutions cooperate which do not trust each other, require a different set of techniques for the differing requirements. Especially data privacy issues and varying security policies by the institutions must be observed.

Anagnostakis et al. [67] presented a system, that envisions sensors provided and operated by ISPs or similar institutions. New correlation and filter rules may be loaded onto the sensors by an authenticated external institution. During that process, all security policies set by the operator of the sensor need to be adhered. So distributed sensors can be controlled and reconfigured by a central entity and simultaneously sensor operators can specify security policies. They also analyzed the system behavior: if false-alarms are intentionally inserted by malicious entities that contribute to the global shared information pool about current events.

Data privacy guaranties for data sources are issued by the system presented by Lincoln et al. [68]. The data sources obfuscate the data themselves, so that sources do not need to trust a central entity and use hash-based algorithms like HMAC on IP headers or payload data. Based on this system, the authors investigate what analysis techniques are appropriate for producing a detailed and error-free evaluation of the events.

Xu et al. [72] introduced a method to combine data privacy with requirements of attack detection. In a first step, sensitive data in an event is generalized to an abstract concept. This results in data with partially semantic information that included uncertainties for data privacy. In a second step, event correlation is performed on the obfuscated data and similarity functions are defined between attributes that were used to create attack scenarios.

A theoretical work about mathematical procedures for data exchange between mistrusting parties was published by Kissner et al. [69]. The authors use mathematical attributes of polynomials to build a framework of efficient and secure multiset operators, that can be combined with each other.

Legal aspects are very hard to identify in this area, as regulations as well as leading decisions by courts are missing. In a comprehensive study the possibilities and regulations for monitoring and storage of packet data in the Internet were investigated primarily based on German law [73]. One of the results were, that there is an astonishing degree of freedom for monitoring and attack detection, but raw IP packets are quickly worth of protection in regard to data privacy issues.

5.3 Report layer

If attacks and threats have been detected, adequate countermeasures need to be initiated. Additionally, automated verification of the detected events can help to increase the quality of the entire security system. In the following, we briefly outline relevant aspects in the report layer.

Active counter-measures Considering the possible attacks in the Internet, it is desirable to automatically react on certain incidents and protect systems from threats. Reactions on worm propagation can be summarized in the following categories [7]:

- *Prevention* – This topic includes techniques that limit or stop worm propagation in the first stages of the infection. Best possible approaches in the Internet is the removal of vulnerabilities in software before those can be exploited. But this is often not possible, so the time until reaction should be as short as possible. A set of heterogeneous systems is an advantage, as those systems usually share only little code and so different vulnerabilities are present. The possibility of detecting vulnerabilities for each single system in use is very low compared to the chance of finding a single vulnerability for one system, so less systems are usually infiltrated.
- *Treatment* – When machines are already infected by some malware, this task covers their disinfection. Unfortunately, this approach does not help during malware propagation, as generation of detection and cleaning algorithms by virus software vendors takes at least several hours.
- *Confinement* – Confinement techniques like those used in firewalls, content filters or black lists in routers can be used against propagating malware and gains time for treatment measures, as it slows down the spread. To be effective, reaction time should be within minutes.

An approach for confinement of attacks like DoS or flash crowds has been proposed by Mahajan et al. [74]. Their scenarios involves high overload of routers or network links that disabled services like web servers. The authors describe a technique that monitors and analyzes packets that were dropped by routers. The source addresses of those packets are sorted and subnetworks, where packets are coming from, are determined. In a second step, a so-called push-back mechanism is employed, that tells upstream routers to apply Random Early Detection (RED) on certain aggregates and so prevents overload of network links that are closer to the target.

Often the so-called *white list approach* is suggested as a solution for this problem. All systems are inserted into that list that may never be blocked. Unfortunately, this is not a flexible solution and is no option for Internet-wide service providers who have customers all over the world and are not able to limit IP address spaces. Another possibility would be the insertion of blocking rules only for a certain time (also see [75,76]), which would avoid having large firewall rulesets during smaller attacks.

Several possibilities for defense against DoS attacks have been presented by Chang et al. [77]. The authors suggest to prevent IP spoofing by filtering IP packets containing invalid addresses already at source networks, but this solution is often not easy to realize. Possibilities on the side of the destination network are among others: IP hopping, where the service providing server changes IP addresses. But this counter-measure is ineffective, if attackers regularly perform DNS requests. Another possibility is buffering of half-open TCP requests by an upstream proxy server. An interesting hybrid approach is the collaboration of ISP and local user systems: a DoS attack is detected by a local system and reported to the provider, who in turn checks the message and may take counter-measures.

All the approaches that we discussed in this section acts automatically and without human intervention. This enables attackers to actively exploit these systems. Exemplary, an attacker could fake an attack and spoof source IP addresses. The attack detection system recognizes the attack and uses the packets' source IP addresses to block access to the network for these seemingly attacking hosts. Attackers could use this mechanism to block out legitimate users interactively and perform a simple DoS attack on the whole network. So all attack detection systems with integrated automated counter-measures must ensure that reported incidents have no (or very low) false-positive rates and this system is not easily to be exploited. Conventional attack detection systems only report these types of incidents to human operators which are not as easily tricked as automatic mechanisms.

Event verification Events that are reported by sensors or other monitoring systems, sometimes need to be checked for validity before they are further processed. An example for these types of events are those coming from anomaly-based systems that have high false-positive rates. Several verification techniques can be classified according to their execution time.

Passive verification methods are executed in regular time intervals, independent of the arrival of events. An example is the generation of metadata, which is used to verify incoming events. Used metadata is usually stored in databases. This method offers the advantage, that it does not interfere the monitored network and so remains invisible. Furthermore, the verification process of single events is usually very quick, as it is executed locally on monitoring systems.

On the other hand, passive verification techniques rely on metadata that has a certain age, and may provide outdated information.

On the other hand, *active verification* algorithms are executed every time an event is reported. These methods may include access to the affected system, which checks if the host is vulnerable to the detected exploit. Further possibilities include reconfiguration of sensors which monitor more details about a specific attack, so that better analysis is possible. Active verification produces more up-to-date data than passive verification, and used verification techniques may be adapted for the type of the incoming event. This may result in better analysis results, as metadata only provides event-independent information and so may be less detailed for current analysis. Active generation of verification data may also be counted as a disadvantage, as it may produce significant load in a network during an attack. Often implemented verification algorithms are executed after some attack was detected. Gathered information may be outdated, as the attacker may already have manipulated the target system to react differently to verification procedures.

Kruegel et al. [16] further differentiate active verification in *remote access*, where a connection to the target system is established for verification, and the target system is not prepared in any way for this verification. *Authenticated access* requires the target system to support the verification procedure like providing a special user account. A *dedicated sensor* is an application, that is already executed on the target system and returns detailed information to the attack detection system on request. The authors combined the rule-based IDS Snort with the vulnerability scanner Nessus. Events generated by Snort were verified by a vulnerability analysis by Nesses. As expected, the high rate of false-positives was strongly reduced.

Manual reactions Severe attacks usually are reported to security operators after event correlation. The attack detection system must forward the events to the people in charge and visualize them appropriately. A management user interface should be provided which allows querying detailed information about incidents. Event archiving may also be valuable for later analysis like the generation of statistics.

6 Conclusion

The domain of event correlation covers a wide spectrum of algorithmic solutions. We have shown in this report that higher layer event correlation needs detailed knowledge about lower layer monitoring and event generation architectures. Basically, correlation of data is an essential method for use in every layer of attack detection and early warning systems, ranging from the raw input data layer over the event layer to the report layer. The main goal is always to improve quality of detected events, to determine if these are related to real incidents, and to prioritize the events according to pre-defined rules. Finally, only highly aggregated relevant information should be presented to the user, so that there is neither information overload nor information loss.

In this study, we also discussed terms thar are used in the scope of attack detection systems – and tried to define the term *event correlation* in this scope. Event correlation groups disjoint events coming from one or multiple systems according to specific attributes, and creates a basis for further analysis. Additionally, metadata about location, time information, or administrative knowledge may be included with the goal to generate results with higher relevance.

We also discussed basic requirements and described different layers in the overall attack detection system. Based on the resulting taxonomy, we performed a literature review to summarize available techniques and systems in a legible way. The whole process of monitoring, attack detection, and correlation of abstract events has been analyzed and available techniques have been associated to this process. Furthermore, we covered relevant technical challenges and administrative barriers.

The primary goal of this study was to establish a starting point for further scientific activities as well as practical work in development of early attack detection in the Internet. Based on the results of this study, the architecture of a planned IT early warning system may be designed and extended by features identified as fundamental in this review.

References

1. Kemmerer, R., Vigna, G.: Intrusion Detection: A Brief History and Overview. IEEE Computer, Special Issue on Security and Privacy (2002) 27–30
2. Jung, J., Milito, R.A., Paxson, V.: On the Adaptive Real-Time Detection of Fast-Propagating Network Worms. In Hämmerli, B.M., Sommer, R., eds.: 4th International Conference of Detection of Intrusions & Malware, and Vulnerability Assessment (DIMVA 2007). Volume LNCS 4579., Lucerne, Switzerland, Springer (2007) 175–192
3. Dressler, F., German, R., Holleczek, P.: Selbstorganisierende Netzwerksensoren und automatisierte Ereigniskorrelation. In: BSI-Workshop IT-Frühwarnsysteme, Bonn, Germany (2006) 117–128
4. Bace, R., Mell, P.: Intrusion Detection Systems. NIST Computer Security Special Publication SP 800-31, National Institute of Standards and Technology (2001)
5. Wood, M., Erlinger, M.: Intrusion Detection Message Exchange Requirements. Technical Report RFC 4766, IETF (2007)
6. Staniford, S., Paxson, V., Weaver, N.: How to Own the Internet in Your Spare Time. In: 11th USENIX Security Symposium, San Francisco, CA, USA (2002) 149–167
7. Moore, D., Shannon, C., Voelker, G.M., Savage, S.: Internet Quarantine: Requirements for Containing Self-Propagating Code. In: 22nd IEEE Conference on Computer Communications (IEEE INFOCOM 2003), San Franciso, CA, USA, IEEE (2003)
8. Moore, D., Voelker, G.M., Savage, S.: Inferring Internet Denial-of-Service Activity. In: 10th USENIX Security Symposium, Washington, DC (2001)
9. Enns, R.: NETCONF Configuration Protocol. Technical Report RFC 4741, IETF (2006)

10. Claise, B.: Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of IP Traffic Flow Information. RFC 5101, IETF (2008)
11. Claise, B.: Packet Sampling (PSAMP) Protocol Specifications. Internet-Draft (work in progress) draft-ietf-psamp-protocol-09.txt, IETF (2007)
12. Calhoun, P., Loughney, J., Guttman, E., Zorn, G., Arkko, J.: Diameter Base Protocol. Technical Report RFC 3588, IETF (2003)
13. Henzinger, M.R., Raghavan, P., Rajagopalan, S.: Computing on data streams. External memory algorithms **50** (1999) 108–118
14. Fayyad, U.M., Piatetsky-Shapiro, G., Smyth, P.: From Data Mining to Knowledge Discovery: An Overview. In: 2nd ACM International Conference on Knowledge Discovery and Data Mining (KDD 2008), Las Vegas, USA (1996) 1–34
15. Lee, W., Stolfo, S.J.: A framework for constructing features and models for intrusion detection systems. ACM Transactions on Information Systems Security **3**(4) (2000) 227–261
16. Kruegel, C., Robertson, W., Vigna, G.: Using Alert Verification to Identify Successful Intrusion Attempts. Praxis der Informationsverarbeitung und Kommunikation (PIK) **27**(4) (2004) 219–227
17. Balas, E., Viecco, C.: Towards a Third Generation Data Capture Architecture for Honeynets. In: 6th IEEE Information Assurance Workshop, West Point, New York, IEEE (2005)
18. Willems, C., Holz, T., Freiling, F.: Toward Automated Dynamic Malware Analysis Using CWSandbox. IEEE Security and Privacy **5** (2007) 32–39
19. Dietz, T., Claise, B., Aitken, P., Dressler, F., Carle, G.: Information Model for Packet Sampling Exports. Technical Report draft-ietf-psamp-info-09.txt, IETF (2008)
20. Quittek, J., Bryant, S., Claise, B., Aitken, P., Meyer, J.: Information Model for IP Flow Information Export. RFC 5102, IETF (2008)
21. Claise, B.: Cisco Systems NetFlow Services Export Version 9. Technical Report RFC 3954, IETF (2004)
22. Siris, V.A., Papagalou, F.: Application of Anomaly Detection Algorithms for Detecting SYN Flooding Attacks. In: IEEE Global Telecommunications Conference (IEEE GLOBECOM 2004), Dallas, TX, USA (2004)
23. Jung, J., Paxson, V., Berger, A.W., lakrishnan, H.B.: Fast Portscan Detection Using Sequential Hypothesis Testing. In: IEEE Symposium on Security and Privacy, Berkeley/Oakland, California, USA (2004)
24. Staniford, S., Hoagland, J.A., McAlerney, J.M.: Practical Automated Detection of Stealthy Portscans. Journal of Computer Security **10**(1/2) (2002) 105–136
25. Debar, H., Curry, D., Feinstein, B.: The Intrusion Detection Message Exchange Format (IDMEF). Technical Report RFC 4765, IETF (2007)
26. Feinstein, B., Matthews, G.: The Intrusion Detection Exchange Protocol (IDXP). Technical Report RFC 4767, IETF (2007)
27. Rose, M.: The Blocks Extensible Exchange Protocol Core. Technical Report RFC 3080, IETF (2001)
28. Yemini, S., Kliger, S., Mozes, E., Yemini, Y., Ohsie, D.: High speed and robust event correlation. IEEE Communications Magazine **34**(5) (1996) 82–90
29. Hager, N.: Secret Power - New Zealand's Role in the International Spy Network. Volume ISBN 0-908802-35-8. Craig Potton Publishing (1996)
30. Anderson, R.J.: Security Engineering: A Guide to Building Dependable Distributed Systems. Wiley (2001)
31. Estan, C., Keys, K., Moore, D., Varghese, G.: Building a better NetFlow. In: ACM SIGCOMM 2004, Portland, OR, USA, ACM (2004) 245–256

32. Hu, Y., Chiu, D.M., Lui, J.C.: Adaptive Flow Aggregation - A New Solution for Robust Flow Monitoring under Security Attacks. In: 12th IEEE/IFIP Network Operations & Management Symposium (IEEE NOMS 2006), Vancouver, Canada (2006) 424–435
33. Duffield, N.G., Lund, C.: Predicting resource usage and estimation accuracy in an IP flow measurement collection infrastructure. In: 3rd ACM SIGCOMM Conference on Internet Measurement (IMC 2003), Miami Beach, FL, USA, ACM (2003) 179–191
34. Duffield, N.G., Lund, C., Thorup, M.: Estimating flow distributions from sampled flow statistics. In: ACM SIGCOMM 2003, Karlsruhe, Germany, ACM (2003) 325–336
35. Hohn, N., Veitch, D.: Inverting sampled traffic. In: 3rd ACM SIGCOMM Conference on Internet Measurement (IMC 2003), Miami Beach, FL, USA (2003) 222–233
36. Estan, C., Varghese, G., Fisk, M.: Bitmap algorithms for counting active flows on high speed links. In: 3rd ACM SIGCOMM Conference on Internet Measurement (IMC 2003), Miami Beach, FL, USA (2003) 153–166
37. Keys, K., Moore, D., Estan, C.: A robust system for accurate real-time summaries of internet traffic. In: ACM Conference on Measurement and Modeling of Computer Systems (SIGMETRICS 2005), Banff, Alberta, Canada, ACM (2005) 85–96
38. Kumar, A., Xu, J., Wang, J., Spatschek, O., Li, L.: Space-Code Bloom Filter for Efficient Per-Flow Traffic Measurement. In: 23rd IEEE Conference on Computer Communications (IEEE INFOCOM 2004), Hong Kong, China, IEEE (2004)
39. Venkataraman, S., Song, D.X., Blum, P.B., Gibbons, A.: New Streaming Algorithms for Fast Detection of Superspreaders. In: 12th Annual Network and Distributed System Security Symposium (NDSS 2005), San Diego, California, USA (2005)
40. Sekar, R., Gupta, A., Frullo, J., Shanbhag, T., Tiwari, A., Yang, H., Zhou, S.: Specification-based anomaly detection: a new approach for detecting network intrusions. In: 9th ACM Conference on Computer and Communications Security (ACM CCS 2002), Washington, DC, USA, ACM (2002) 265–274
41. Toth, T., Kruegel, C.: Connection-history based anomaly detection. In: 3rd IEEE Information Assurance Workshop, West Point, New York, USA (2002)
42. Estan, C., Savage, S., Varghese, G.: Automatically Inferring Patterns of Resource Consumption in Network Traffic. In: ACM SIGCOMM 2003, Karlsruhe, Germany, ACM (2003) 137–148
43. Weaver, N., Staniford, S., Paxson, V.: Very Fast Containment of Scanning Worms. In: 13th USENIX Security Symposium, San Diego, CA, USA (2004) 29–44
44. Karagiannis, T., Papagiannaki, K., Faloutsos, M.: BLINC: multilevel traffic classification in the dark. In: ACM SIGCOMM 2006, Pisa, Italy, ACM (2005) 229–240
45. Wright, C.V., Monrose, F., Masson, G.M.: On Inferring Application Protocol Behaviors in Encrypted Network Traffic. Journal of Machine Learning Research **6** (2006) 2745–2769
46. Bernaille, L., Teixeira, R.: Early Application Identification. In: 2nd Conference on Future Networking Technologies (CoNext 2006), Lisboa, Portugal (2006)
47. Roesch, M.: Snort: Lightweight Intrusion Detection for Networks. In: 13th USENIX Conference on System Administration (LISA 1999), Seattle, WA, USA (1999) 229–238
48. Yusuf, S., Luk, W.: Bitwise Optimised CAM for Network Intrusion Detection Systems. In: 15th IEEE International Conference on Field Programmable Logic and Applications (FPL 2005), Tampere, Finnland (2005) 444–449

49. Paxson, V.: Bro: A System for Detecting Network Intruders in Real-Time. In: 7th USENIX Security Symposium, San Antonio, TX (1998)
50. Wang, K., Cretu, G., Stolfo, S.J.: Anomalous Payload-Based Worm Detection and Signature Generation. In: 8th International Symposium on Recent Advances in Intrusion Detection (RAID 2005), Seattle, Washington, USA (2005) 227–246
51. Bolzoni, D., Etalle, S., Hartel, P.H., Zambon, E.: POSEIDON: a 2-tier Anomaly-based Network Intrusion Detection System. In: 4th IEEE International Workshop on Information Assurance (IEEE IWIA 2006), Royal Holloway, UK, IEEE (2006) 144–156
52. Kohonen, T.: Self-Organizing Maps. Springer (2000)
53. Singh, S., Estan, C., Varghese, G., Savage, S.: Automated Worm Fingerprinting. In: 6th USENIX Symposium on Operating Systems Design and Implementation (OSDI 2004), San Franciso, CA, USA, USENIX Association (2004) 45–60
54. Kim, H.A., Karp, B.: Autograph: Toward Automated, Distributed Worm Signature Detection. In: 13th USENIX Security Symposium, San Diego, CA, USA (2004) 271–286
55. Parekh, J.J., Wang, K., Stolfo, S.J.: Privacy-preserving payload-based correlation for accurate malicious traffic detection. In: ACM SIGCOMM Workshop on Large-Scale Attack Defense (LSAD 2006), Pisa, Italy, ACM (2006) 99–106
56. Vaarandi, R.: Simple Event Correlator for real-time security log monitoring. Hakin9 Magazine **1**(6) (2006) 28–39
57. Vigna, G., Valeur, F., Kemmerer, R.A.: Designing and implementing a family of intrusion detection systems. In: European Conference on Software Engineering (ESEC), Helsinki, Finland (2003) 88–97
58. Cuppens, F., Mige, A.: Alert Correlation in a Cooperative Intrusion Detection Framework. In: IEEE Symposium on Security and Privacy, Oakland, California, USA, IEEE (2002)
59. Valeur, F., Vigna, G., Kemmerer, C., Krügel, R.: A Comprehensive Approach to Intrusion Detection Alert Correlation. IEEE Transactions on Dependable and Secure Computing **1**(3) (2004) 146–169
60. Yegneswaran, V., Barford, P., Ullrich, J.: Internet Intrusions: Global Characteristics and Prevalence. In: ACM International Conference on Measurement and Modeling of Computer Systems (ACM SIGMETRICS 2003). (2003) 138–147
61. Krügel, C., Toth, T., Kerer, C.: Decentralized Event Correlation for Intrusion Detection. In: 4th International Conference on Information Security and Cryptology (ICISC 2001). Volume LNCS 2288., Seoul, Korea (2001) 114–131
62. Blanc, M., Oudot, L., Glaume, V.: Global Intrusion Detection: Prelude Hybrid IDS. Technical report, AIRstack (2003)
63. Gruschke, B.: Integrated Event Management: Event Correlation Using Dependency Graphs. In: 9th IFIP/IEEE International Workshop on Distributed Systems: Operations and Management (DSOM 1998), Newark, Delaware, USA (1998)
64. Fröhlich, P., Nejdl, W., Schroeder, M., Damásio, C., Pereira, L.M.: Using extended logic programming for alarm-correlation in cellular phone networks. In: 12th International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems (IEA/AIE 1999). Volume LNCS 1611., Cairo, Egypt (1999) 343–352
65. Steinder, M., Sethi, A.: End-to-end service failure diagnosis using belief networks. In: 8th IEEE/IFIP Network Operations & Management Symposium (IEEE NOMS 2002), Florence, Italy (2002) 375–390
66. Pearl, J., Russell, S.: Bayesian Networks. In Arbib, M.A., ed.: Handbook of Brain Theory and Neural Networks. MIT Press, Cambridge, MA (2003) 157–160

67. Anagnostakis, K., Greenwald, M., Ioannidis, S., Li, A.K.D.: A Cooperative Immunization System for an Untrusting Internet. In: ACM International Conference on Measurement and Modeling of Computer Systems (ACM SIGMETRICS 2003), San Diego, California, USA, ACM (2003)
68. Lincoln, P., Porras, P.A., Shmatikov, V.: Privacy-Preserving Sharing and Correlation of Security Alerts. In: 13th USENIX Security Symposium, San Diego, CA, USA (2004) 239–254
69. Kissner, L., Song, D.X.: Privacy-Preserving Set Operations. In: 25th Annual International Cryptology Conference (CRYPTO 2005), Santa Barbara, California, USA (2005) 241–257
70. Huang, Q., Wang, H.J., Borisov, N.: Privacy-Preserving Friends Troubleshooting Network. In: Network and Distributed System Security Symposium (NDSS 2005), San Diego, California, USA (2005)
71. Wang, H.J., Platt, J.C., Chen, Y., Zhang, R., Wang, Y.M.: PeerPressure for automatic troubleshooting. In: ACM International Conference on Measurement and Modeling of Computer Systems (ACM SIGMETRICS 2004), New York, NY, USA, ACM (2004) 398–399
72. Xu, D., Ning, P.: Privacy-Preserving Alert Correlation: A Concept Hierarchy Based Approach. In: 21st Annual Computer Security Applications Conference (ACSAC 2005), Tucson, AZ, USA (2005) 537–546
73. Haibl, F., Dressler, F.: Anonymization of Measurement and Monitoring Data: Requirements and Solutions. Praxis der Informationsverarbeitung und Kommunikation (PIK) **29**(4) (2006) 208–213
74. Mahajan, R., Bellovin, S.M., Floyd, S., Paxson, J.I.V., Shenker, S.: Controlling high bandwidth aggregates in the network. ACM SIGCOMM Computer Communication Review (CCR) **32**(3) (2002) 62–73
75. Dressler, F.: Adaptive Re-Configuration of Network Monitoring Applications. In: Dagstuhl Seminar 06011: Perspectives Workshop: Autonomic Networking, Schloss Dagstuhl, Wadern, Germany (2006)
76. Dressler, F.: Bio-inspired Promoters and Inhibitors for Self-Organized Network Security Facilities. In: 1st IEEE/ACM International Conference on Bio-Inspired Models of Network, Information and Computing Systems (IEEE/ACM BIONETICS 2006), Cavalese, Italy, IEEE (2006)
77. Chang, R.K.C.: Defending against Flooding-Based Distributed Denial-of-Service Attacks: A Tutorial. IEEE Communications Magazine **10** (2002) 42–51
78. Bloom, B.H.: Space/time trade-offs in hash coding with allowable errors. Communications of the ACM **13**(7) (1970) 422–426
79. Zaraska, K.: Prelude IDS: current state and development perspectives. Technical report (2003)
80. Carey, M., Criscuolo, P., Petruzzi, M., Rogers, R.: Nessus Network Auditing. 2nd edn. Syngress Media (2007)