Proceedings of the 8th Colloquium for
Information Systems Security Education
West Point, NY, June 2004

Event Data Exchange and Intrusion Alert Correlation in Heterogeneous Networks

Antti Hätälä, Camillo Särs, Ronja Addams-Moring and Teemupekka Virtanen, *Helsinki University of Technology*

Abstract: If we want to correlate alerts from various intrusion detection system (IDS) sources, its is necessary that the sources of alerts agree on what they actually are seeing, on how to report what they are seeing and on the amount of information they should report. In this paper, we review the Intrusion Detection Message Exchange Format (IDMEF) data model as an event data exchange mechanism and analyze how different correlation algorithms are being utilized in real-life systems. Based on these analyses, we propose a simple taxonomy of intrusion alert correlation algorithms, to complement the IDMEF data model.

Intrusion Detection, Alert Correlation, Network Security

I. Introduction

Increasingly often, when attackers attempt to gain access into private networks, they need to take a step-by-step approach. As all the target network's defenses will unlikely be down or vulnerable at the same time, the attacker involves several hosts and services, and uses each compromised host as a stepping stone to get deeper into the target network.

To recognize such step-by-step attacks, a need for an automated mechanism for detecting traces of large-scale intrusion plans (instead of distinct intrusion events) has evolved. This mechanism, a correlation function, attempts to recognize the intrusion plan by examining alerts from heterogeneous sources. So correlation is, in essence, a form of abstracting the alert data.

However, no intrusion detection system (IDS) can, alone, supply all the data needed for recognizing high-level attack scenarios. To achieve good recognition, the data needs to be collected from various sources: firewalls, web server logs, IDS systems of multiple manufacturers, etc.

One recognition answer is i*ntrusion alert correlation,* the process of identifying alerts that belong to the same intrusion scenario. The goal of intrusion alert correlation is to present a clear high-level picture of attacks that are going on in the target network, to the people administering the network.

The function of intrusion alert correlation can be divided in three parts [1]:

- *Correlation as aggregation* Grouping together correlated alerts reduces the rate of the alert stream, making it easier for analyzing software or personnel to see what really is going on.
- *Correlating information from heterogeneous sources* Only by correlating alerts from a plethora of different sources can we truly understand the nature of the attack.
- *Context correlation* Intrusions can be better understood as attack scenarios than individual low-level events. The gain is improved sensitivity and a decreased false-alarm rate.

In this article we analyze and classify different alert correlation methods and algorithms from a theoretical point of view. We examine examples of current IDS alert correlation implementations and discuss the algorithms used in them, the languages they provide for describing correlation and their organization and structure. We discuss IDS interoperability issues on a general level, reviewing a proposed standard, Intrusion Detection Message Exchange Format (IDMEF), for alert data exchange. Finally, we propose a simple taxonomy of alert correlation algorithms which could be added to the IDMEF data model.

With the number and variety of network incidents steadily rising, sophisticated alert correlation schemes will be of paramount importance in networked intrusion detection systems. Moreover, a classification scheme such as the one proposed here will be needed to be able to address issues of interoperability in future networked IDS systems. The challenge of automated response to IDS alerts, however, remains outside the scope of this paper.

II. Correlation methods

The boundaries between the terms alert aggregation, alert clustering and alert correlation are vague, though all of

Helsinki University of Technology, Telecommunicactions software and multimedia laboratory. P.O.Box 5400, 02015 HUT..

ISBN 0-87145-192-1/$15.00

them aim for the same goals. We define alert correlation to be the top of the hierarchy in the ``alerts to alarm'' -- process. Alert aggregation and clustering happen also on lower levels -- in the sensors themselves -- and they usually only deal with alerts corresponding to a single attack, while correlation takes as input alerts already clustered from heterogeneous sources and mimics the function of a human expert in trying to reconstruct the attack scenario. Aggregation and clustering can take place in the correlator as well.

Note that *alert* and *event* are distinct concepts [2]. An event is a low level entity that is analyzed by the IDS, whereas an alert is generated by the IDS to notify interested parties of interesting events. A single event can cause many alerts, especially in a networked IDS environment, and a single alert can describe a set or sequence of events. Furthermore, from the point of view of an IDS that performs correlation of alerts generated by other IDS systems, alerts are in fact also events themselves.

Most correlation efforts follow the reasoning behind ``misuse detection'' intrusion detection systems, correlating alerts with prior knowledge of misuse patterns. Simple patterns only match similarities between the attributes of alerts, while more elaborate ones aim to encompass complex attack scenarios.

Most of the misuse detection based correlation methods present their own event correlation language for defining the misuse patterns, and the power and level of abstraction of this language is of great importance. In addition to simple alert attribute correlation (ch. 2 A) and correlation using knowledge about known attack scenarios (ch. 2 B) we will also have a look at a more recently introduced approach where misuse patterns are defined as pre- and postconditions for individual alerts (ch. 2 C). This threefold classification of misuse based correlation methods is due to [3].

The most obvious shortcoming in all of the misuse pattern detection algorithms is that they fail to correlate alerts of previously unknown attacks. To address this shortcoming, a totally different approach has been developed. The idea is to correlate alerts using statistical attack scenario analysis much in the fashion of anomaly detection IDS systems. This approach is discussed in chapter 2 D.

A. Correlating events between similar attack attributes

Alerts belonging to the same attack often have similar attributes. Inspecting alert attributes and finding similarities between them is a straightforward way of recognizing correlation. The important questions here are which attributes to compare, how to tell if they are similar and what weight the different attributes should be given in the comparison. Also notable is that it often is unfeasible to try to correlate all alerts pair wise with one another and therefore a decision has to be made which alerts or groups of alerts should be considered for correlation.

Valdes and Skinner use in their EMERALD correlation component [4] a well-founded list of alert attributes -- or alert *features* as they put it. They consider 1) sensor identification which includes the identifier, location and name of the sensor, 2) the alert thread, which is a way of the sensor deciding about correlation itself and overrides all other criteria, 3) the source and target IP lists, 4) the target TCP/UDP port lists, 5) the source user id, 6) the target user id and 7) time. Specific incident signatures as reported by sensors are not of great relevance, since they will vary quite a lot between heterogeneous sensors.

Deciding the level of similarity of attributes is by no means a trivial task. Only considering perfect matches is called *heuristic alert correlation*, which is in many cases too simplified an approach. *Probabilistic alert correlation* [4], expresses similarity on a scale from 0 to 1. Depending on the attribute, the similarity value can be for example taken from a predefined similarity matrix (e.g. similarity of different attack classes), calculated as the numerical distance (e.g. time) or the degree of overlap of lists (e.g. lists of IP-addresses). Naturally, not all sensors produce alerts with all of the similarity criteria -- only attributes available are considered in a similarity calculation.

In probabilistic alert correlation, the overall similarity of two alerts is calculated as a weighted average over the attribute similarities, using predefined similarity expectations as weights. Attributes can also be assigned situation-specific minimum similarity values. By using different similarity expectation and minimum similarity values we can get several different views of the incidents. For example, if we relax the expectations for sensor identification and demand a high probability for attack class, we hope to get a better view of individual security incidents reported by heterogeneous sensors. On the other hand, by relaxing expectations for similarity of attack class we can expect to see various steps in a multistage attack.

Algorithms of this type are ultimately not able to capture the true relationships between events, as stated by Ning et al. [8]. Carefully handcrafted similarity criteria (i.e. similarity matrices, similarity expectation values) can lead to well-behaving systems that capture the essence of many previously known attack types but the system in itself does not understand the attacks -- it is the expert maintaining the system that is responsible for the knowledge.

ISBN 0-87145-192-1/$15.00

B. Correlating events based on predefined attack scenarios

The next logical step in correlation method improvement is to program the system with known attack scenarios. Programming the correlation system can either be done by machine learning mechanisms or by human experts feeding blueprints -- or templates -- of known attack scenarios into the system. Run-time correlation then is a process of filling in attack scenario templates and the main issue is how to formulate the attack scenarios in an abstract and general way. Attack scenarios are usually broken down to explicit correlation rules, i.e. conditions that need to be fulfilled for two alerts to correlate.

In an experiment led by Dain and Cunningham [5], the probabilistic approach to correlation was compared to a data mining approach where the attack scenarios used were in advance fed to a data mining system. The authors actually call the probabilistic correlation heuristic correlation, but it is in essence the same algorithm as depicted by Valdes and Skinner using only three attributes: attack class, time and source IP. The results are clear: the probabilistic approach produced the correct correlation result with a certainty of 88.81% while the data mining approach scored an impressive 99.90%. The attack scenarios defining the correct correlation were hand-picked by a human expert. The experiment shows the potential benefits of using data mining techniques, but the environment of the experiment was not very realistic as the authors used alerts captured in a hacking contest (DEF CON 2000 Capture the Flag) and their data mining system was fed with information of all identified scenarios in the contest in advance.

Correlation languages for describing attack scenarios have been developed in plenitude. A recent work by Morin and Debar utilizes the chronicles formalism [2]. Chronicles provide a very high level language for describing the scenarios based on time information, and they are used in many areas to monitor dynamic systems. Krügel et al. propose an attack specification language ASL in their distributed pattern matching scheme [12]. The LAMBDA-language used in the Mirador-project is also very similar [9]. These languages utilize predicate logic to describe states of the system.

The NetSTAT framework uses pre-written attack scenarios in another way: it divides the scenarios in advance to smaller tasks that can be carried out by single sensors and then correlates these tagged alerts [6]. Although an interesting approach, it is not very general and therefore does not contribute much to the correlation problem.

Predefined attack scenarios specified by human users or learned through training datasets are indeed a good source for correlation information, but this type of correlation is still bound to be restricted to known attacks.

C. Correlating events by information of pre- and postconditions of individual attacks

One recently introduced powerful mechanism of expressing correlation criteria is defining pre- and postconditions for individual attacks. This can be seen as just another way of describing attack scenarios, but because we are only stating pre- and postconditions for distinct attacks, we do not have to know complete attack scenarios in advance and we do not have to insert huge amounts of correlation rules manually. It is sufficient to state the required conditions for a known attack and the possible outcomes of that attack. This way, also new attack combinations and scenario types can be recognized and reported.

This approach has been studied in parallel by Templeton et al. in the JIGSAW attack description language [7], by Ning et al. [8] and by Cuppens et al. [9]. Implementations of the idea differ in the language used to describe pre- and postconditions and in the additional features of the algorithms.

D. Correlating events by statistical causality analysis

The last approach moves away from traditional misuse detection toward anomaly detection. Qin and Lee use an algorithm called the Granger Causality Test to correlate events [10]. Pure statistical causality analysis does not need predefined knowledge about attack scenarios, thus completely new attack scenarios can also be recognized.

The motivation for statistical analysis is that every multi-step attack generates alerts that have statistical similarities in their attributes, and that attack steps have a causal relationship: if step X is the cause of step Y, then X must precede Y, and most likely the steps of an attack will have a high probability of occurring together (in a small time window). Qin and Lee also present the idea of running the statistical correlation engine offline with training datasets to compute and store correlations to be used for pattern matching at run-time. [10]

As it is today, the statistical causality approach is not a feasible solution for the complete correlation process, but it can be utilized as a part of a larger system to pre-process alerts or to provide meta-alert signatures.

ISBN 0-87145-192-1/$15.00

III. Correlation in current networked IDS frameworks

In this chapter, real-life examples of implementing alert correlation techniques are studied. In addition to utilizing an efficient correlation algorithm and presenting a high-level declarative language for the system administrator to feed in correlation rules, the system needs to have an internal structure and organization of accomplishing correlation. Currently at least three classes of structures can be seen:

1. completely centralized event data processing
2. a hierarchical structure
3. a completely distributed structure

In describing the different alert correlation framework examples, we will concentrate on 1) the algorithm, 2) the correlation specification language and 3) the structure. The systems are being reviewed purely from the alert correlation point of view, other merits are not discussed.

A. The ACC component of the Tivoli Enterprise Console

Debar and Wespi present the Aggregation and Correlation Component (ACC) they built on top of the Tivoli Enterprise Console (TEC), an event-handling product of IBM/Tivoli Systems in their RAID paper from year 2001 [11]. They focus on addressing four of the major problems in current IDS systems, namely: alert flooding, context insensitivity, false alerts and scalability. The system architecture consists of ACC's and probes. The probes are either directly IDS alert generators that can output alerts in the common TEC-event format (support for the emerging IDMEF standard is planned for the future) or gateway probes that are attached to proprietary systems and transform the alerts to the common format. The authors claim a hierarchical structure of ACC's, but no notion of how this is accomplished is presented.

The correlation algorithm employed in the ACC is a hybrid algorithm using both predefined attack scenarios and alert attribute comparison. Whenever correlation succeeds, alerts are merged according to the correlation rule and treated as one thereafter and alarms are created if necessary. The system has no correlation language, the correlation rules have to be entered into the system manually or generated from the configuration files.

ACC uses two different kinds of preprogrammed correlation rules: duplicates and consequences. Duplicate rules tie together alerts from different sources that represent the same attack. The rule definition of duplicates specifies which attack classes are considered as duplicates and which attributes of the alerts must match. The attack classes are specific names of known attacks. Consequence rules work in a similar way, binding together alerts that are known to happen pair wise in a multi-step attack. In consequence rules, two probe ids and two attack classes are defined and a wait period is given to represent the maximum time between the two alerts considered for correlation. Predefined attack scenarios can be programmed into the system by defining appropriate duplicate and consequence rules but this is very cumbersome and lacks generality.

Another aspect in the ACC algorithm is the attribute similarity calculation. The algorithm utilizes an elementary version of the minimum similarity and expected similarity -idea described in chapter 2 A by only comparing selected attributes to form different views of the incident. Only perfect matches of attributes are considered and the algorithm uses only three attributes: the source, the target and the attack class.

B. The CRIM module of the MIRADOR project

Cuppens and Miége have developed a cooperative alert correlation module called CRIM [9] in the MIRADOR project of the French Defense Agency. The CRIM module actually also takes care of more rudimentary alert clustering and merging, but our interest is strictly in the correlation function. The CRIM model is completely centralized: all analyzing and processing takes place in a central module, although the CRIM module creates as output a correlated alert (an alert of a full attack scenario) that can be sent to another module. Events are received by CRIM in the IDMEF format (see chapter 4 A).

The correlation algorithm in CRIM uses predefined attack scenarios in the form of *explicit correlation rules* that state the conditions when two alerts correlate. These rules can be fed into the module in the LAMBDA language by the system administrator. LAMBDA is a predicate logic language for describing attacks. Attacks have five attributes: preconditions, postconditions, attack scenarios, detection scenarios and verification scenarios. The alerts to be generated are described in the detection scenario and the actual recognition of the attack is defined in the attack scenario. Correlation in LAMBDA is carried out with the alert_correlation predicate, which simply takes two alerts as arguments. The definition of alert correlations has three parts: two descriptions of alerts to be correlated and the conditions to be satisfied to correlate the alerts.

Cuppens and Miége state, however, that writing explicit correlation rules is too cumbersome, too error-prone and the resulting knowledge base will rarely be exhaustive enough; therefore they introduce semi-explicit correlation rules, which implement the idea of pre- and postconditions of individual attacks as described in

ISBN 0-87145-192-1/$15.00

chapter 2 C. Their proposal is to create the explicit rules from the pre- and postcondition definitions of attacks in advance and to continue using explicit runtime rules.

The process of transforming semi-explicit definitions into explicit rules and using the explicit rules in correlation includes the following ideas:

- **Direct correlation** means finding all the correlating pairs where there exists a most general unifier that unifies an attacks postconditions with another's preconditions.
- With **indirect correlation** the system includes a mechanism of representing ``known facts'' or *ontological rules*, that are rules that bind together predicates via external rules. These are useful for representing common knowledge such as if the NetBios-port is open the OS is Windows. By using these ontological rules as bridges, alerts can be correlated indirectly.
- With **abductive correlation** the system is also equipped with a mechanism of correlating scenarios in situations where there are missing links in the correlation chain by creating virtual alerts when needed.

C. Quicksand

In contrast to the two centralized correlation schemes presented in previous chapters, *Quicksand* by Krügel et al. has a completely distributed structure [12]. Quicksand divides the process of correlation to Event Correlation Units (ECU) residing on numerous machines on the network. These units communicate directly to the probes (IDS's) using a standard message format (IDMEF, see ch. 4 A) to receive alerts and implement the distributed pattern matching algorithm. In the typical case, an ECU resides on the same machine that the probe it is communicating with and there is one ECU per probe.

Quicksand is a pure ``predefined attack scenarios'' algorithm as described in chapter 2 B, the novel idea being the ability to run the algorithm in a distributed manner. The attack scenarios are fed in using the Attack Scenario Language (ASL) developed by the authors. Patterns are defined as correlating sets of events with events having unique identifiers by event type and situation-specific constraints. Patterns can have a `send'--event that triggers the sending of the events collected in the pattern to another ECU for further processing. The complete attack scenario is thus divided to local patterns in several ECUs. The ECU's have all the information needed to make decisions locally about which events to forward. A central Control Unit is still needed to initialize the patterns on each of the ECUs according to the ASL definitions.

Distributing alert correlation in a network serves many purposes: the amount of required network traffic decreases since ECU's know which alerts are relevant to other nodes, fault-tolerance increases and the overall performance of the correlation function is better.

IV. Interoperability

In large networks, IDS and other activity reporting systems from different vendors exist and will continue to exist. An open standard for the intrusion alert data model is needed for correlating the events produced by heterogeneous sources. Ideally, a standard for the intrusion event data would be capable of conveying sufficient intrusion event information for any correlation scheme utilized. However, a standard intrusion alert data model does not solve all problems related to cross-product alert correlation.

A. The IDMEF data model and intrusion event exchange format

The Internet Engineering Task Force (IETF) has been working on an intrusion alert data model and accompanying message format standard called the Intrusion Detection Message Exchange Format (IDMEF). The XML-based message format specification currently exists as an Internet Draft, which is more a working memo of the working group than a publication [13]. It has been submitted to the Internet Engineering Steering Group for consideration as an RFC. The Internet Draft expired on July 31 2003: up-to-date information about the current situation does not seem to be available. The working group responsible for the draft (IDWG) has not attended the last two IETF meetings and the IDWG meeting report of the IETF held in Atlanta in November 2002 states that the working group is ``idle awaiting action by IESG'' [14]. Even though the standardization process is still going on and seemingly having difficulties, the IDS research and development community has welcomed the draft specification with great excitement. All the IDS frameworks discussed in chapter III either currently employ the draft message format or state clear plans of supporting the format whenever the IDMEF gets approved as a RFC. Also other proprietary intrusion alert formats are readily transformable into the IDMEF standard since it represents a good superset of best practices used in distributed IDS alert models. It has few strictly defined mandatory attributes and it has been designed to be easily extensible without loosing the common base. The IDMEF draft is a profound effort toward interoperability of IDS systems and it is the only proposition backed by any standardization at the time being, since the Common Intrusion Detection Framework development has lost momentum [15].

ISBN 0-87145-192-1/$15.00

One of the main design goals of the IDMEF data model is to be able to express relationships between alerts. This goal is directly derived from the need of alert correlation. In describing the motivation of IDMEF the authors of the draft state: ``an event correlation system that could accept alerts from a variety of intrusion detection products would be capable of performing more sophisticated cross-correlation..." [13].

The data model of the IDMEF is object-oriented. Currently the top level class message is inherited by two subclasses, the Heartbeat and the Alert messages. Furthermore, the Alert class is currently inherited by three more specific Alert types: the ToolAlert, the OverflowAlert and the CorrelationAlert. Alert aggregation is expressed with the ToolAlert and CorrelationAlert subclasses. Both of them are alerts that are meant to group together a set of previously sent alerts and they carry the alert id, analyzer id information of the related alerts in them. The analyzer id -part is needed to correlate alerts from different sources since the alert id is only unique within one source. The ToolAlert differs from the CorrelationAlert only by carrying also information about recognized attack tools or other malicious programs that have been used in the event that the alerts refer to. The CorrelationAlert and ToolAlert make it possible to correlate alerts in a hierarchic way: the lower-level correlator can send information about alerts it has successfully correlated upward with the two alert aggregation message types.

All of the components of the Alert class are relevant in alert correlation, and the list of components seems to be comprehensive in the light of example implementations of correlation mechanisms discussed in chapter 2. However, the source, target and classification components need further investigation. Their most interesting attributes are shown in tables 1-3. Note that here attributes and aggregate classes are treated similarly, the purpose of the table is not to present the precise relation of objects but to show which things are considered in different aggregate classes of the Alert class.

Attribute	Multiplicity	Description
Node	Zero or one	Information about the host or device that the event is being targeted at
User	Zero or one	Information about the user that the event is being targeted at
Process	Zero or one	Information about the process that the event is being targeted at
Service	Zero or one	Information about the network service involved in the events
Filelist	Zero or one	Information about the file(s) involved in the events

Table 1: IDMEF Alert Class aggregate class "Target" relevant attributes

Attribute	Multiplicity	Description
Node	Zero or one	Information about the host or device that appears to be causing the event
User	Zero or one	Information about the user that appears to be causing the event
Process	Zero or one	Information about the process that appears to be causing the event
Service	Zero or one	Information about the network service involved in the events

Table 2: IDMEF Alert Class aggregate class "Source" relevant attributes

ISBN 0-87145-192-1/$15.00

Attribute	Multiplicity	Description
Origin	Exactly one	The source from which the name of the alert originates:"unknown", "bugtraqid", "cve" or "vendor-specific".
Name	Exactly one	The name of the alert, as stated in Origin
URL	Exactly one	A URL at which in-depth information about the attack can be found

Table 3: IDMEF Alert Class aggregate class "Classification" relevant attributes

As we can see from the table, the target and source components carry very similar information. The Filelist component exists only in target information since files, passive by their nature, can not be seen as sources of events, only targets. The list of sources and targets of an event takes into consideration both host--based and network--based misuse detection, and further divides host--based sources and targets into ``user" and ``process" and network--based sources and targets into ``node" and ``service". The approach has clearly been bottom--up: to encompass all known attack types. The resulting specification allows all sorts of meaningless combinations, lacks generality and enforces the event sources and targets into artificial categories. However, for most modern attacks the target and source definitions are sufficient and provide correlators with well-defined and useful data. All identifiers in the attributes follow global standards and schemes, for example a ``Node" can carry an address of type ``e-mail", ``ipv4-addr", ``mac" or ``atm", just to name a few.

Unfortunately, no global scheme for event classification exists and therefore the Classification component of the Alert is not globally useful for correlation. The correlator has to have knowledge about specific attack strings that the messages carry and the participating IDS systems must all use the same vendor-specific or otherwise agreed upon scheme, a scenario that is not very likely in heterogeneous networks. This will be further discussed in the next chapter.

The IDMEF alert model succeeds quite well in covering all the elemental attributes needed for alert correlation. For example, all attributes considered for comparison in the EMERALD correlation component listed in chapter 2 A are in place, noting that the alert thread can be expressed with the CorrelationAlert subclass. Also correlation using predefined attack scenarios can be done using the IDMEF messages, provided that the participating sensors agree on the attack class classification scheme. The IDMEF format does not directly support defining attack scenarios or basing correlation on them, other mechanisms have to be utilized for this effect, and CorrelationAlert and ToolAlert messages can be used for exchanging correlation information. The CorrelationAlert message lacks means of expressing what the correlation is based on, but since the XML definition is designed to be extensible, correlation schemes can introduce their own CorrelationAlert subclasses to provide additional data. Also statistical analysis can be done with the information exchanged in the IDMEF messages as reported by Qin and Lee [10], but more generic anomaly detection based schemes that are likely to emerge in the future will require a totally different approach, since they do not operate on alerts at all.

B. Intrusion event taxonomy and ontology

A common message format is not sufficient for IDS systems to really understand the reported events if they have not explicitly been programmed into each IDS. From the correlation point of view, whenever a novel attack type emerges it must be programmed into all participating IDS systems so that correlation can take place. Clearly this is not satisfactory. In addition to the message format, we need at least some sort of a common taxonomy (or classification) of attacks, or preferably a way of defining higher-level knowledge about the domain -- an ontology of intrusion events -- built in the data model itself. Kemmerer and Vigna express this concern in a brief overview of the state of intrusion detection systems by stating ``...Additional effort is needed to provide a common ontology that lets sensors agree on what they see...", while commenting on the IDMEF data model [16]. This issue is even more prevalent with the advent of next generation anomaly based IDS systems.

Classifying intrusion events is difficult, because the variety of events is overwhelming and because there are no obvious characteristics of intrusions that could be used as classification criteria. Lindqvist and Jonsson propose using *intrusion result* and *intrusion technique* as dimensions for classification and they formulate two taxonomies based on the different criteria [17]. Their taxonomies represent the results of a small intrusion experiment and would need to be extended to be generally useful. Other proposals include taxonomies based on attack signatures and on computer security flaws [18, 19].

A novel effort of defining an intrusion event ontology is done by Undercoffer et al. They present a way of distributed IDS systems sharing domain knowledge about

ISBN 0-87145-192-1/$15.00

intrusions and making correlation decisions based on knowledge built in the data model. The ontology is defined using the DARPA Agent Markup Language+Ontology Inference Layer (DAML+OIL) and is made publicly available. The authors argue that using their ontology, IDS systems would not have to be programmed with knowledge of specific attacks to recognize, report and correlate events. The only prerequisite for a distributed IDS system with heterogeneous sensors and analyzers is that they share the same ontology. [20]

C. Additional information about the environment and the state

Another major issue with a common intrusion data model is that plain intrusion event data is sometimes not enough for correlation, also information about the system and the environment is needed. As a simplified example, consider the situation where there has been an alert from a host about a port scan involving the port 139 (Windows NetBIOS port) succeeded by another alert from the same about a Microsoft IIS buffer overflow vulnerability exploit. The two alerts correlate only if we know that the host under attack runs the Microsoft Windows OS, because we can now deduce that an attacker tried to find out which operating system the host runs and launched an operating system specific exploit after that.

Information about the system environment and about the state of the system is especially crucial to the method of correlating alerts based on pre- and postconditions of individual attacks. As observed from the CRIM module in chapter 3 B, additional ``known facts" are needed to be able to indirectly justify the correlation of alerts using the known fact as a link from a postcondition of an alert to the precondition of another.

M2D2 is a formal data model developed by Morin et al. that takes into account in addition to event information also ``information related to the characteristics of the monitored information system, information about the vulnerabilities and information about security tools for the monitoring". The M2D2 data model can potentially uncover correlation criteria that are not recognizable by only looking at intrusion events: for example the authors show a way of aggregating alerts that refer to the same vulnerability. The M2D2 model can use alert messages in the IDMEF format, which further increases its attractiveness. [2]

V. CONCLUSIONS

We discussed the IDMEF IDS interoperability standard draft from the correlation point of view and saw that, for it to be useful in a more general manner, the common message format must be complemented with a commonly accepted ontology and other data models to represent information about the system environment and the state of the system.

We have seen that the intrusion alert correlation methods used in real networked IDS systems indeed can be classified into several families and that there is a pressing need for such a taxonomy to aid in the research of intrusion alert correlation algorithms and to achieve interoperability on a more abstract level. We feel that at least four distinct alert correlation classes are justified:

1. Algorithms based on similarities in attack attributes
2. Algorithms based on knowledge about attack scenarios
3. Algorithms based on information about pre- and postconditions of individual attacks
4. Algorithms based on statistical analysis

The IDMEF standard caters well for all of our correlation algorithm classes and the data model can be used even for event data exchange across the algorithm class boundaries. More research is needed, though, to show how algorithms of different types can co-operate.

The taxonomy we have proposed is a very simple one -- the classification criteria consists only of the existing ideas of how to correlate intrusion alerts. A more elaborate taxonomy could take into account several other attributes as well.

ISBN 0-87145-192-1/$15.00

VI. References

[1] Alfonso Valdes and Keith Skinner: "An Approach to Sensor Correlation", an essay presented at the RAID symposium 2000, Toulouse, France; 2000.

[2] Benjamin Morin, Ludovic Mé, Hervé Debar and Mireille Ducassé: "M2D2: A Formal Data Model for IDS Alert Correlation"; in Proceedings of Recent Advances in Intrusion Detection 2002, LNCS 2516, p. 115-137; Springer-Verlag; 2002.

[3] Peng Ning, Yun Cui and Douglas S. Reeves: "Analysing Intensive Intrusion Alerts via Correlation"; in Proceedings of Recent Advances in Intrusion Detection 2002, LNCS 2516, p. 74-94; Springer-Verlag; 2002.

[4] Alfonso Valdes and Keith Skinner: "Probabilistic alert correlation"; in Proceedings of Recent Advances in Intrusion Detection 2001, LNCS 2212, p. 54-68, ISBN 3-540-42702-3; 2001.

[5] Oliver Dain and Robert K. Cunningham: "Fusing a Heterogeneous Alert Stream into Scenarios"; in Proceedings of the 2001 ACM Workshop on Data Mining for Security Applications, p. 1-13; 2001.

[6] Giocanni Vigna and Richard A. Kemmerer: "NetSTAT: A network-based intrusion detection system"; Journal of Computer Security, 7/1999, p. 37-71.

[7] Steven J. Templeton and Karl Levitt: "A requires/provides model for computer attacks"; in Proceedings of New Security Paradigms Workshop, p. 31-38; 2000.

[8] Peng Ning, Yun Cui and Douglas S. Reeves: "Constructing attack scenarios through correlation of intrusion alerts"; in Proceedings of the 9th ACM conference on Computer and Communications security; 2002.

[9] Frédéric Cuppens and Alexandre Miége: "Alert correlation in a cooperative intrusion detection framework"; in Proceedings of the IEEE Symposium of Security and Privacy; 2002.

[10] Xinzhou Qin and Wenke Lee: "Statistical Causality of INFOSEC Alert Data"; in Proceedings of Recent Advances in Intrusion Detection 2003, LNCS 2820, p. 73-94; Springer-Verlag; 2003.

[11] Hervé Debar and Andreas Wespi: "Aggregation and correlation of Intrusion-Detection Alerts"; in Proceedings of Recent Advances in Intrusion Detection 2001, LNCS 2212, p. 85-104; ISBN 3-540-42702-3; 2001.

[12] Cristopher Krügel, Thomas Toth and Clemens Kerer: "Decentralized Event Correlation for Intrusion Detection"; in Proceedings of the 4th International Conference on Information Security and Cryptology, p. 114-131; Springer-Verlag; 2001.

[13] D. Curry and Hervé Debar: "Intrusion Detection Message Exchange Format Data Model and Extensible Markup Language (xml) Document Type Definition"; Internet Draft, draft-ietf-idwg-idmef-xml-10.txt; 2003.

[14] Intrusion Detection Working Group: "IDWG meeting report"; at IETF November 2002, Atlanta, Georgia, USA; http://www.ietf.org/proceedings/02nov/192.htm; 2002.

[15] Clifford Kahn, Phillip A. Porras, Stuart Staniford-Chen and Brian Tung: "A common intrusion detection framework"; http://www.isi.edu/gost/cidf/papers/cidf-jcs.ps; 1998.

[16] Richard A. Kemmerer and Giovanni Vigna: http://www.computer.org/computer/sp/articles/kem/; referenced on 21.10.2003.

[17] Ulf Lindqvist and Erland Jonsson: "How to systematically classify computer security intrusions"; in Proceedings of the IEEE Symposium on Security and Privacy, p. 154-163; 1997.

[18] S. Kumar: "Classification and detection of computer intrusions"; Ph.D. thesis, Purdue University; 1995.

[19] C. E. Landwehr et al.: "A taxonomy of computer security program flaws"; ACM Computing Surveys, vol. 26, p. 211-254, 9/1994.

[20] Jeffrey Undercoffer, Anupam Joshi and John Pinkston: "Modeling Computer Attacks: An Ontology for Intrusion Detection"; in Proceedings of Recent Advances in Intrusion Detection 2003, LNCS 2820, p. 113-135; Springer-Verlag; 2003.

ISBN 0-87145-192-1/$15.00