Components for Cooperative Intrusion Detection in Dynamic Coalition Environments

Marko Jahnke, Michael Bussmann, Sven Henkel
Research Establishment for Applied Science (FGAN)
Research Institute for Communication, Information Processing and Ergonomics (FKIE)
Neuenahrer Str. 20, 53347 Wachtberg
Germany

{jahnke|henkel|bussmann}@fgan.de

Jens Tölle
University of Bonn
Institute for Computer Science, Dept. IV
Römerstr. 164, 53117 Bonn
Germany

toelle@cs.bonn.edu

ABSTRACT

We present a prototype of an Intrusion Warning System for combining event message flows of multiple domain-specific security tools in order to determine anomalies for early warning and response. Unlike other approaches for cooperating Intrusion Detection Systems (IDS), we suggest a modified star shape architecture for distributing attack information and feed back warning messages. We assume that there are almost no known properties, neither of the underlying information providing local security tools nor of their local security policies. Such heterogeneous environments are typical for dynamic coalitions like NATO. We extended a well-known hierarchical distributed IDS architecture to provide Meta IDS services with feedback to the local access points. The extensions include three major items: Early Anomaly Warning - A graph clustering based anomaly detector for the event messages is used as an adaptive early warning module for largely scaled activities, e.g. internet worms. Information Sanitizing - Event messages are anonymized when leaving the local domain, according to a domain-specific information sharing policy. Message Aggregation - Additional filters for data reduction and application of predefined correlation rules make the data flow feasible

1.0 INTRODUCTION

Since many networks in coalition environments (CE) - like NATO - are connected with each other using the public internet to transfer unclassified data - or even classified information, using appropriate security mechanisms - it is possible to save enormous amounts of costs for dedicated connections. Thus, those networks are exposed to many different threats.

One of the biggest threats are spreading internet worms. Obviously it is easier to detect largely scaled security related activities when having access to large security attack information (i.e. event messages or audit records) from different locations within a computer network (IDS, firewalls, virus scanners etc.). But combining different attack information sources becomes even more important when aiming at detecting coordinated attacks against large numbers of target systems, where there is a common strategy behind all activities.

Paper presented at the RTO IST Symposium on "Adaptive Defence in Unclassified Networks", held in Toulouse, France, 19 - 20 April 2004, and published in RTO-MP-IST-041.

So, one major goal from the CE security analyst's perspective is to combine all attack information in order to gain more input for this task. We define an Intrusion Warning System (*IWS*) for coalition environments as a system that gathers attack information from all available sources and generates warning messages about unusual system behaviour. There are different architectures to process coalition-wide audit data, such as *distributed cooperating IDS* and *centralized Meta IDS* which gain and process data from local security tools. This paper describes an architecture which meets general requirements of heterogeneous dynamic CEs. Additionally it discusses algorithms used for event message processing and anomaly detection and the status of their implementation.
The rest of this paper is structured as follows: Section 2 points out the essential issues for designing and building an IWS for CEs. Section 3 describes the overall architecture of our prototype system. Then, sections 5, 4 and 6 describe new architecture components which are necessary to solve the described issues. In section 7 we discuss the initial implementation and testing results. Section 8 outlines some of the related work that already has been done and points out the differences to our approach. Section 9 concludes on past activities and describes possible future directions.

2.0 ISSUES AND APPROACHES FOR A CE IWS

Before it is possible to design and build a robust and effective IWS for CE scenarios, several issues are to be solved. Some of these issues have been identified in a NATO/RTO working group report on IDS [1]. For our specific needs, we extend these issues to the following list of items:

- *Efficient Analyzer Algorithms*
  To get a benefit from collecting information distributed in the CE, we primarily need algorithms which help us to detect largely-scaled activities. On one hand, we need correlation techniques to detect coordinated activities. On the other, we also need anomaly detectors to act as an early warning system to feed the local security staffs with appropriate information.
- *Information Sharing Policies (IShPs)*
  One of the most important issues to be solved in order to realise coalition-wide exchange of attack information are differences between IShPs, since there may be information within IDS messages which are not to be exposed to even the closest coalition partners. An example for the information sanitizing is the anonymization of IP addresses, if a domain wants to keep its network topology secret. Another application would be hiding of information about utilized security products which is often contained within event messages. To solve this problem, we need flexible mechanisms for *attack information sanitizing*.
- *Security Policies*
  Also the security policies which are to be enforced by local IDS may differ in various ways (e.g. in one domain, port scans are treated as a potential attack, whereas in another domain, they are completely neglected). This leads to alternating pools of messages with different priorities, levels of trust etc.. To handle this, either the system can use mechanisms for “Message Normalization” to homogenize incoming data or use analyzer techniques, which work independently of these properties.
- *Architecture*
  There are different ways to implement an architecture for cooperative intrusion detection. In earlier approaches, such as described in a generic paper of Frincke et al. [5], the term “Cooperative IDS” was interpreted as establishing direct communication links between the IDS nodes (or, in general, local event message collectors) in the affected domains. For dynamic CEs, this model has several disadvantages, and thus it is not applicable:

- *Traffic Overhead*
  A fully connected network of n domains would require $n \cdot (n-1)$ communication links between the IDS nodes. This leads obviously to a lot more network traffic than necessary.
- *Separate Information Sanitizing Policies*
  Since information about attacks is transferred to each other domain separately, a separate policy for sanitizing information has to be created and maintained, in accordance to the established trust levels.
- *Continuous Reconfiguration*
  Since in a dynamic CE new domains may be integrated and the trust levels between the existing ones may change, continuous reconfiguration efforts are necessary for up to $n \cdot (n-1)$ information flows.
- *CE Command Structures*
  Since the conventional command structures in CEs like NATO are organized in a hierarchical way, individually communicating IDS nodes are contradicting.

An alternative for a peer-to-peer structure is a star shape architecture with bi-directional links from the single domains to the centralized event message processing unit. But it also has to be mentioned that using only one central unit is a single point of failure, and therefore fallback mechanisms have to be integrated in the architecture.

- *Data Format and Protocols*
  To use event messages, which have been collected by different security tools from different vendors, we need a commonly agreed data format and message transport protocol. Fortunately, the Intrusion Detection Working Group of the IETF (IDWG) has released proposals for this, such as the XML-based IDMEF format [3] and the IDXP profile [4] for the BEEP protocol [12]. Many vendors of commercial systems claimed to support these formats in order to be interoperable with other systems.
- *Secure and Reliable Communication*
  To deliver event messages in a secure manner, we need according communication channels. Thus, mechanisms for information integrity, confidentiality and authentication are necessary. These services can be provided by adding an additional crypto layer to the communication channels, such as SSL/TLS as a part of IDXP/BEEP, which itself provides a reliable message transport on top of TCP.

Although goals like a *Coordinated Intrusion Response* are very important for reacting on distributed attacks, they are beyond the scope of this paper.

3.0 GENERAL ARCHITECTURE

According to the objections to a peer-to-peer structure of message flows in section 2, we suggest a different architecture: a centralized flow of attack information with central information processing capabilities, and additional feed back links to the local IDS nodes for early warning information, as depicted in fig. 1.

This architecture is based on our IDS infrastructure framework which provides generic pluggable components. One or more central units (each called a Meta IDS Console) is deployed for processing all messages received from different domains. These component contain information storage capabilities as well as message filtering and processing modules. GUIs are deployed for offline message inspection and for real-time display of incoming messages. The central analyzing component is a message anomaly detector, which is described in section 4. To avoid a single point of failure, additional consoles are acting as a simple fallback solution for the primary console.

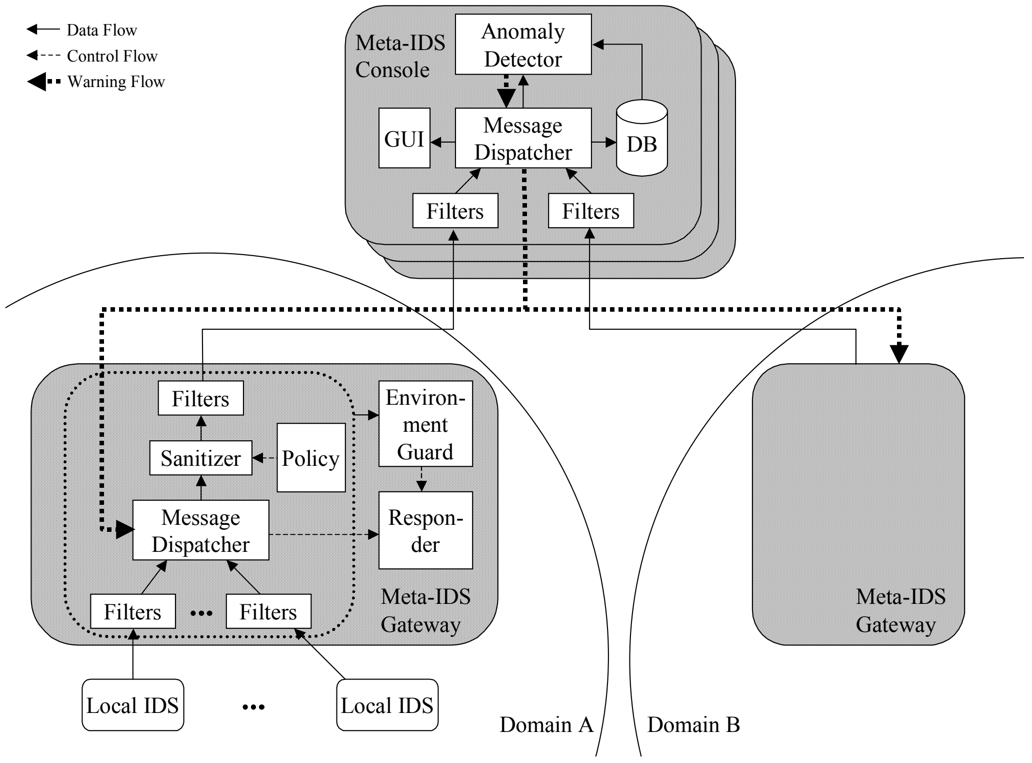

Figure 1: Architectural components of the meta IDS

Consoles receive event messages from several Meta IDS Gateways (GWs) which collect all messages occurring within a specific domain and distributes warning information back to the GWs. Thus, our GWs provide a suitable bi-directional interface to the domains; section 5 provides more information on our GW concept. Since the communication behaviour of GWs is similar to IDS agents, all previously developed infrastructure related countermeasures against Denial-of-Service attacks can be applied (e.g. component redundandencies, message overload protection, process environment integrity checks and process observation, see [6]) .

In our system, the IDMEF recommendation [3] of the IETF IDWG is used as a common data model for representing attack information as event messages. Accordingly, messages are encoded as XML documents and transmitted over network links via the IDXP/BEEP protocol [4].

4.0 ANOMALY DETECTION IN THE EVENT MESSAGE DATA MODEL

The structure of the entire system allows the usage of a method for detecting unusual activities. The event messages generated by the systems located in each domain are sent via the gateways to the meta IDS. The collected event messages are the base for an approach for detecting abnormal system behaviour. Due to the idea of the entire system, it is difficult to make assumptions on the type, the quality, or the frequency of event messages. The basic idea for surveying the current system state and for the detection of deviations (anomaly detection) is the continuous monitoring of the arriving event messages. The method used here was originally developed for monitoring and detecting network traffic anomalies (see [13]). In this application context, the method works as follows:

The typical structures of network traffic are quite stable. Fundamental changes are unusual. This allows to gather the traffic in regular intervals (this can be done using monitoring devices or traffic sniffer, in switched networks it is necessary to use a monitoring port of the switch) and stored as a traffic matrix. This matrix can be seen as a graph $G = (V, K)$. Nodes (vertices) $v_i \in V$ of graph G represent communicating devices while edges $K_{i,j} \in K$ represent the communication between node v_i and v_j. The edges are weighted according to the intensity of the communication during the measurement period.

Those graphs can be partitioned into subgraphs using graph clustering algorithms. The clustering of the graph represents the typical communication structure of the monitored network. *Clustering* means finding a mapping of each node to one out of a set of several clusters. This exclusive classification is also called partitioning of the object set. Each object of an object set is assigned to exactly one cluster. In a more formal description, clustering is the partitioning of a graph $G = (V,K)$ into n cluster

- $C_i \subseteq V, C_i \neq \emptyset, 0 \leq i \leq n-1$ with
- $C_0 \cup C_1 \cup ... \cup C_{n-1} = V$ and
- $\forall 0 \leq i, j \leq n-1, i \neq j : C_i \cap C_j = \emptyset$

Sudden variations of these structures are regarded as anomaly. To discover these kind of variation, metrics are needed. Those metrics have to rate similarities (or dissimilarities) of consecutive clusterings $\Re_t$ and $\Re_{t+1}$.

This method used in the area of traffic structures needs some adaptations for the event message model presented in this section. Most of the event messages arriving at the meta IDS are suitable for building a graph. All the event messages containing detailed information on the destination (the attacked system) and the origin (assumed originator) belong to this category. Such a message generates a new edge k in the set of edges K of our message graph G. The edge k connects the nodes representing the origin v_{origin} and the destination $v_{destination}$.

Depending on the level of detail of the event messages, some gateways may just indicate the affected system but not specify the (assumed) originator. Depending on the configuration of the underlying security tools and meta IDS gateways, this information may not be available. In order to consider these event messages in the event message graph, a *pseudo node* representing these event message types may be assigned to the domains. An edge between such a pseudo node and a node representing an affected system represents an event message not containing an originator address.

The graph partitioned with graph clustering algorithms describes the typical structure of the incoming event messages. Variations from the typical message structure are seen as an anomaly. Those anomalies are reported as warning messages. The comparison measures used for anomaly detection in network traffic structures can be used in this domain as well.

When using anomaly detection methods, a major challenge is the sensitivity to false alarms (*false positives*) or unreported serious events (*false negatives*). This holds for the methods used here as well. A careful selection of parameters according to the usage scenario is necessary to keep false positive and false negative rates low. This method is currently being integrated into the overall prototype system, as described in section 7.

5.0 POLICY-CONFIGURABLE EVENT MESSAGE GATEWAYS

To provide an event message sanitizing service, domain-specific gateways (GWs) are needed. Their basic structure is outlined in the lower left part of figure 1. They perform the following tasks:

- Collecting all local event messages and passing them to the central meta IDS, independently of the local security tools that generate attack information.
- Inspecting message flow for sensitive information and filter (sanitize) messages according to the current IShP (see sect. 2).

- Accessing warning information from the meta IDS and providing it to the local security staffs and their supporting systems.
- Optionally perform filter operations on messages, such as normalizing message contents and adjusting priority values.

GWs are functionally controlled by the meta IDS, but configuring the IShP is a matter of the local domain and the responsible security staffs. Since IShPs are one of the most important issues concerning IDS in CEs, a more formal specification of the information sanitizing process within the GW is useful. It can be expressed as an event matching and transformation problem:

Given the local IShP, it should be possible to generate a set of conditional transformation rules in the shape of $R = (\{ E_0^M \}, \emptyset, E^T)$ as described in appendix A. But unfortunately, a useful information sanitizing process - as needed for our GWs - cannot solely rely on static text substitution. For example, when replacing IP addresses with fixed values, all information about network topology is lost. Obviously, this may hamper the intrusion detection process, especially when using traffic-related anomaly detection. Thus, we need a more flexible way of defining transformation rules: *submatching references*.
Let $s(E_0^M)$ be a set of qualified subexpressions in the matching template E_0^M, and $s(E_0^M, e)$ the set of substrings of a given event e which actually match the subexpressions in E_0^M. Now, if we extend the transformation template to

$$E^T : \Sigma \times s(E_0^M) \times P \to P(\Sigma),$$

we have the possibility to construct new events, based on submatchings of the old events. An example for this is the 'substitute' command of the standard Unix `sed(1)` tool (e.g. the command

```
s/192\.22\.([0-9]{1,3}\.[0-9]{1,3})/191\.72\.\1/
```

leads to the substitution of the 2-byte-prefix of matching IP addresses).
To extend the possibilities for the transformation rules even more, it is obviously useful to involve current system parameters like the time of the last matching, the current system time or the IP address of the gateway host as parameter set *P*.

6.0 EVENT MESSAGE AGGREGATION MODULES

In our approach, two kinds of message aggregation should help us in making high amounts of messages feasible: *redundancy filtering* and application of predefined rules for sets of correlated events (*combination detection*). Filter modules for event aggregation can be applied at different places in the architecture, as shown in figure 1. Like the message sanitizing techniques, as described in the previous section, we can reduce it to an event pattern matching and transformation problem.

6.1 Redundancy Filtering

In order to reduce the number of event messages, it is obviously necessary to avoid redundancies. Several messages with a “similar” content shall be merged into one single message, that represents the merged ones sufficiently. To specify the merging process of the aggregated messages, we can again utilize transformation templates E^M as described in section 5.

The definition of event “similarities” must be configurable, since it strongly depends on domain specific parameters, like deployed security tools and the security policy that has to be enforced. Therefore, we extend event matching templates E^M accordingly. Since we do not only need absolute (initial) matchings, but even *relative matchings* (i.e. depending on previous ones, defining the difference), we have to enhance the matching process by introducing a second matching template E_1^M that refers to submatchings of the initial matching template E_0^M. Thus, for the set matching templates $E^M = \{E_0^M, E_1^M\}$, we have

$$E_0^M : \Sigma \to \mathrm{B}, \qquad E_1^M : \Sigma \times s(E_0^M) \to \mathrm{B}$$

Example: If all messages with identical classifications and source/target IP addresses are to be filtered, and if the message creation time is within a time range of one second,

- E_0^M has to contain qualified subexpressions for the message creation time and for source and target address to be able to refer to them later,
- E_1^M must contain a conditional expression for the values of source and target address, which are identical to the according values of the event e that matches the initial matching template E_0^M (referred as $e \dashv E_0^M$), and
- E_1^M must contain a conditional expression for the values of creation time, which are not more than 1 second away from the creation time of the initial matching event.

In contrast to (stateless) conditional transformations, where a separate matching of all templates $E^M{}_i$ is required for a transformation, we need two buckets

$$B_i(t) = \{e \mid e \dashv E_i^M\}, i = 0,1 \qquad \text{with} \qquad 0 \leq |B_0(t)| \leq 1, |B_1(t)| \geq 0 \forall t \in T$$

for previously matched events, at least to determine which template is currently to be matched next. Additionally, we need the storage time t_{B_0} of the first event which has been stored in bucket B_0 to be able to indicate the end of the aggregation phase after a maximum storage time Δt_{B_0} has been exceeded. The transformation function does not only depend on the current event e, but also on the previously stored ones, i.e. we define

$$E^T : P(\Sigma) \times P(\Sigma) \times s(E_0^M) \times P \to P(\Sigma)$$

and for the transformation function f_R a redundandency filtering process algorithm that does not involve external parameters. Therefore it is straightforward:

Input: R=$(\{E_0^M, E_1^M\}, \varnothing, E^T)$, Δt_{B_0} and a sequence of events $e(t)$

Output: A sequence of event sets $\{e\}$, where similar events are summarized.

Algorithm:

```
$B_0(0)$ := $\varnothing$ ; $B_1(0)$ := $\varnothing$ ; $t_B$ := 0 $t := 1$
while true do
    read $e(t)$
    if $t_{B_0} \neq 0$ and $t - t_{B_0} \geq \Delta t_B$ then
        output $E^T(B_0(t), B_1(t), s(E_0^M))$
        $B_0(t)$ := $\varnothing$ ; $B_1(t)$ := $\varnothing$
        $t_{B_0}$ := 0
    fi
    if $e(t) \not\dashv E_0^M$ and $e(t) \not\dashv E_1^M$ then
        output $\{e(t)\}$
```

```
            fi
            else if $|B_0| \neq 0$ and $e(t) \dashv E_0^M$
            then
                  $B_0(t)$ := $\{e(t)\}$
                  $t_B$ := $t$
            fi
            else if $|B_0| > 0$ and $e(t) \dashv E_1^M$
            then
                  $B_1(t)$ := $B_1(t-1) \cup \{e(t)\}$
            fi
            $t$ := $t+1$
      done
```

Note that this algorithm does not require that similar events are coming in sequentially, without having different messages in between them. Additionally, it is either possible to use multiple buckets $B_{i,j}(t)$ in parallel, but the number of buckets must be limited in order to avoid overload situations and DoS attacks. It is also possible to modify the later output event *e*' each time a matching occurs, instead of calculating *e*' = $E^T(B_0, B_1, s(E_0^M))$ after the end of the aggregation phase. Since then only one bucket for one event is needed, a lot of memory is saved.

6. 2 Combination Detectors

We define *event combinations* as correlated sets of event messages. Detectors for these combinations apply predefined rules for the events and their correlation. Note that our approach does not include the generation of correlation rules between event messages, but once they are specified (e. g. by a procedure as described by Julisch [7]), we are able to apply them on the message flow in a very flexible manner. Combination detectors are a generalized case of the redundancy filters as described above, since instead of having one absolute and one relative matching, we need a set of relative matching templates $E^M = \{E_1^M, \ldots, E_n^M\}$ to describe an event combination. Every relative matching may not only refer to the submatchings of the initial matching E_0^M, but also of the

$$E_i^M : \Sigma \times \prod_{j=0, j\neq i}^{n} s(E_j^M) \rightarrow \mathrm{B}, \forall i = 1, \ldots, n$$

We assume multiple buckets $B_i, 0 \leq i < n$ for the different matching templates. Thus it shall be possible to refer to all of the stored events in the buckets by extending E^T to:

$$E^T : \prod_{i=0}^{n} P(\Sigma) \times \prod_{j=0}^{n} s(E_j^M) \times P \rightarrow P(\Sigma)$$

With this approach, we can model different correlations which are important to detecting security relevant situations. To see how this works, we look at correlations between two matching templates E_0^M and E_1^M :

- *Classification Correlation*:
  Defining expressions in E_0^M and E_1^M which describe a subset of possible event classifications (e.g. enumerations or ranges of attack database IDs).

- *Temporal Correlation*:
  Defining a subexpression in E_0^M containing the time value and an arithmetic expression in E_1^M that defines the correlation (e.g. a maximum detection time difference).
- *Location Correlation*:
  Defining a subexpression in E_0^M containing the address value and an arithmetic expression in E_1^M that defines the correlation (e.g. belonging to the same subnet).

Of course, arbitrary combinations of all of the above may be defined to specify more complex correlations, e.g. *More than 100 TCP packages from address A to address B on port 22, containing potential NoOp instructions and one packet from B to A from port 22, containing text "uid=0(root)" which comes right after the last packet of the earlier type* which might be a serious indication for a well-known attack against the SSH login daemon.

7.0 INITIAL RESULTS

The architecture is implemented as a prototype system, which is called "MSIDI" (Meta SIDI), which is based on our earlier IDS prototype "SIDI" (Survivable Intrusion Detection Infrastructure, [6]), using its C++ class framework. Using instances of SIDI for providing event messages to the meta IDS and additionally developed gateways, all necessary infrastructure and management components are implemented, as well as the majority of the message filtering and processing features mentioned within this paper.

7.1 Anomaly Detector

Our approach for detecting anomalies in event message flows has been implemented, currently only looking at the source/target address properties of messages. At present, the detector is being integrated in the overall prototype system.

The evaluation of the detection approach is currently done using several data sources. One of those data sources is a selection of more than 400 Mbytes of real event messages representing the original data traffic of one of our test configurations. An additional test system is a system used for tests with artificially generated event messages mixed with real event messages. This test system can be used with input data, e.g. generated by a system simulating worm spreading. This simulator simulates the spreading of real worms (Code Red v2, Code Red II,...) or worms using modified or new spreading methods.

7.2 Information Sanitizing

Our approaches for information sanitizing, for redundancy filtering and for event combination detection presented in sections [5] and [6] are all reduced to the conditional event message transformation (see appendix A). A common way of performing complex transformations on XML formatted data is the application of XSLT Stylesheet [14]. Unfortunately, this recommendation does not include REs, and no suitable implementations of according extensions are available at the moment. Thus, for our first meta IDS prototype, we implemented an XML processor which recognizes POSIX.1 REs in XSLT templates, expands Boolean/arithmetic expressions for additional conditions and applies the matching and transformation on IDMEF messages.

An equivalent to the IP address prefix substitution example from sect. 5 is the following *transformation sheet* (integrated specification of matching and transformation template):

```
<address>
192\.22\.([0-9]{1,3})$v1$\.([0-9]{1,3})$v2$
  <condition>($v2<255)</condition>
  <transform>
```

```
    <xsl:copy>
     191.72.<xsl:value-of select="$X"/>.<xsl:value-of select="$Y"/>
    </xsl:copy>
  </transform>
</address>
```

The matchings of the last two bytes of the IP address `192.22.X.Y` are further on referred as variables `$X` and `$Y`. The additional condition for performing the transformation is that `$Y` is not the traditional broadcast suffix 255.

Using these basic transformation techniques, we have implemented the information sanitizing functionalities of the event message gateways as described in section 5. We are utilizing the GNOME libxml2 and libxslt libraries for XML processing to create highly configurable C++ filter modules which can be easily plugged into our infrastructure components.

To evaluate the efficiency of our filtering approaches, we created a number of transformation sheets and counted the number of messages per time frame which our sanitizing gateway is able to transform.
Using a PIII/1GHz PC with 256MB RAM, running Debian GNU/Linux, the message gateway is able to handle the amounts of message transformations, as shown in figure 2 (messages supplied by one domain IDS, no compiler optimizations). Our test results confirmed the assumption, that the way of specifying matching and transformation templates has a big impact on the filtering performance. Wherever applicable, different matching templates should be condensed into one. In this case, it is not necessary to analyze incoming messages up to their ends before the next matching template is applied.

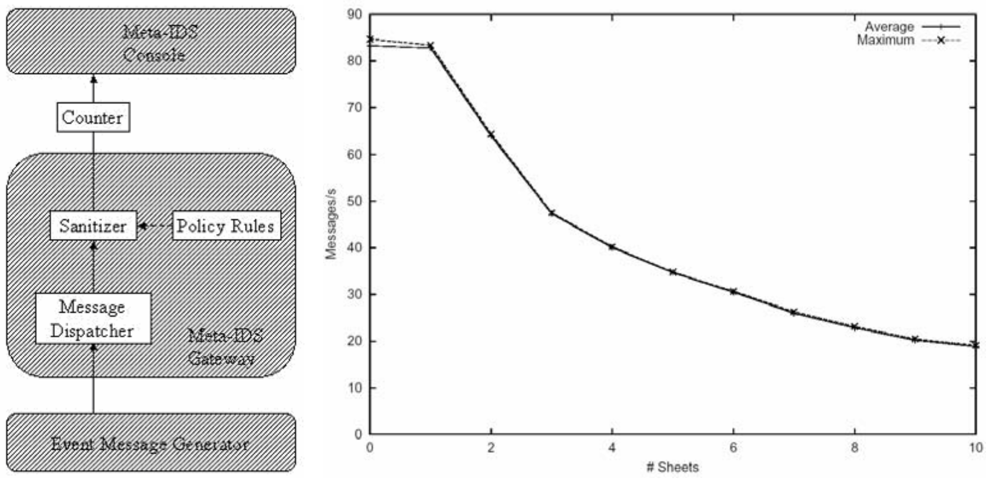

Figure 2: Scenario and results of the throughput tests for the information sanitizing gateway.

8.0 RELATED WORK

Cooperative IDS have been examined in many publications:

An early generic cooperative intrusion detection paper of Frincke et al. [5], identified many of the cooperation issues, such as policy conflicts, information sharing and filtering. In these premises and in the context of allowing different local audit tools to be used within the affected domains, the paper can be considered a basis for our work. A major difference to our work is the described architecture, which relies on direct communication links between the distributed audit data collectors.

The approach of Kim et al. [8] has several architectural analogies to our work. Unfortunately their approach relies on the fact that the suggested architectural components are installed all over the place which is to be monitored. This is not suitable for dynamic CEs, since every single domain has its own principles for selecting products and vendors.

The cooperating software entities, described in [9] as well as in [15], have similar concepts to our meta IDS gateway, but again, the architecture is on a peer-to-peer basis.

A lot of papers describe approaches for detecting correlations between alerts (e.g. [2], [10], [11]). Since our system only provides mechanisms for detecting predefined correlations of events, off-line correlation techniques are necessary to define the according detection rules. Due to our very general approach in specifying such rules, it should be feasible to import externally generated correlation rules.

9.0 SUMMARY AND FUTURE ACTIVITIES

This paper has presented an architecture to act as a cooperative intrusion warning system for dynamic heterogeneous coalition environments. Due to its structure, it meets several requirements for dynamic CEs, which cannot be met by other approaches relying on directly communicating domain-specific units.

The most important extensions are a central message processing unit where an anomaly-based detection module is included for providing early warning information about largely-scaled attacks, and integrated modules for information sanitizing and data reduction. For a prototypical implementation, we extended a previously developed distributed IDS infrastructure.

The anomaly detector module is based on a graph clustering technique which has been successfully deployed for IP traffic analysis. All event message processing tasks have been reduced to problems of Conditional Pattern Matching and Transformation (CPMT), which we have implemented as extended XML/XSLT Stylesheet processors. Thus, we are able to filter the message flow in a very flexible manner.

Currently, we are focussing on

- exploring the impact of other information contained in event messages on the graph-based anomaly detection approach,
- distinguishing the different reasons for detecting anomalies and extracting useful information for warning messages, and
- the implementation of the redundandency filter and the combination detector by extending the CPM techniques.

ACKNOWLEDGEMENTS

Parts of this work have been sponsored by the German Federal Office for Information Management and Information Technology of the Bundeswehr (Bundeswehr IT-Office). The authors would like to thank Michael Schmeing (FGAN/FKIE) for valuable comments on the topic.

References

[**1**] R. Coolen and H.A.M. Luiijf.
Intrusion Detection Systems - Generics and State of the Art.
Technical report, NATO/RTO Information Systems Panel, Task Group on Information Assurance (TGIA, IST-008), 2001.

[2] F. Cuppens and A. Miege.
Alert Correlation in a Cooperative Intrusion Detection Framework.
In *Proceedings of the IEEE Symposium on Research in Security and Privacy*, pages 202-215, Oakland, CA, May 2002. IEEE Computer Society, Technical Committee on Security and Privacy, IEEE Computer Society Press.

[3] D. Curry and H. Debar.
Intrusion Detection Message Exchange Format - Data Model and Extensible Markup Language (XML) Document Type Definition.
IETF Internet Draft draft-ietf-idwg-idmef-xml-10.txt, January 2003.
IETF IDWG.

[4] B. Feinstein, G. Matthews, and J. White.
The Intrusion Detection Exchange Protocol.
IETF Internet Draft draft-ietf-idwg-beep-idxp-07.txt, October 2002.
IETF IDWG.

[5] D. A. Frincke, D. Tobin, J. C. McConnell, J. Marconi, and D. Polla.
A Framework for Cooperative Intrusion Detection.
In *Proc. 21st NIST-NCSC National Information Systems Security Conference*, pages 361-373, 1998.

[6] M. Jahnke.
An Open and Secure Infrastructure for Distributed Intrusion Detection Sensors.
In *Proceedings of the NATO Regional Conference on Communication and Information Systems (RCMCIS'02), Zegrze, Poland*, October 2002.

[7] K. Julisch.
Mining Alarm Clusters to Improve Alarm Handling Efficiency.
In *Proceedings of the NATO/RTO IST Workshop on Inforensics and Incident Response, The Hague, The Netherlands*, October 2002.

[8] Byoung-Koo Kim, Jong-Su Jang, and Tai M. Chung.
Design of Network Security Control System for Cooperative Intrusion Detection.
Lecture Notes in Computer Science, 2344:389-, 2002.

[9] G. Koutepas, F. Stamatelopoulos, V. Hatzigiannakis, and B. Maglaris.
An Adaptable Inter-Domain Infrastructure Against DoS Attacks.
In *Proc. of the International Conferences on Advances in Infrastructures for e-Business, e-Education, e-Science, e-Medicine on the Internet (SSGRR 2003)*, L'Acquila, Italy, January 2003.

[10] Benjamin Morin, Ludovic Mé, Hervé Debar, and Mireille Ducassé.
M2D2: A formal data model for IDS alert correlation.
volume 2516, page 115, 2002.

[11] P. Ning, Y. Cui, and D. Reeves.
Constructing attack scenarios through correlation intrusion alerts.
Technical Report TR-2002-12, Department of Computer Science, North Carolina State University, August 14 2002.

[12] M. Rose.
RFC 3080: The Blocks Extensible Exchange Protocol Core.
http://www.ietf.org/rfc/rfc3080.txt, March 2001.

[13] J. Tölle and C. de Waal.
A Simple Traffic Model Using Graph Clustering For Anomaly Detection.
Proc. of Applied Simulation and Modelling (ASM) Crete, Greece, June 2002.

[14] W3C.
W3C Recommendation 16: XSL Transformations (XSLT) Version 1.0.
http://www.w3.org/, 1999.

[15] Q. Zhang and R. Janakiraman.
Indra: A Distributed Approach to Network Intrusion Detection and Prevention.
Technical report, University of Washington, St. Louis, 2003.

A NOTATION

Since many of the information flow problems in distributed IDS can be modelled as variations of conditional pattern transformations for event message, it is useful to describe the problem in a more formal fashion:

Let Σ be the set of event messages or *events*. An *event matching template* E is an expression that expands to a set of events i.e. $Expand(E) \in P(\Sigma)$. An event e *matches* a template E (noted as $e \dashv E$), if and only if $e \in Expand(E)$. The *matching function* of E is intuitively given as

$$E : \Sigma \to IB, \quad E(e) = \begin{cases} true, & \text{if } e \in Expand(E) \\ false, & \text{if } e \notin Expand(E) \end{cases}$$

Let $P = \{(p_1, \dots p_m)\}$ be a set of arbitrary given time-variant parameter values. A *conditional transformation rule* R is a triple

$$R = (E^M, P, E^T)$$

with a set of *matching templates*

$$E^M = \{E_i^M \mid E_i^M : \Sigma \to IB, i = 0, \dots, n\}$$

and a *conditional transformation template*

$$E^T : \Sigma \times P \to P(\Sigma).$$

Let T be the time interval of interest, $e(t) \in \Sigma \forall t \in T$ a sequence of events and $p(t) \in P$ a sequence of parameter values. The *conditional transformation function* of the rule R is

$$f_R : \Sigma \times T \to P(\Sigma) \text{ with}$$

$$f_R(e(t), t) = \begin{cases} E^T(e(t), p(t)), & \text{if } e(t) \dashv E_i^M, \forall i = 0, \dots, n \\ \{e(t)\}, & \text{otherwise} \end{cases}$$

i.e., depending on the matches of the templates E_i^M to the input event $e(t)$ at time t, zero or more output events are generated by the filter. Note, that this mechanism is stateless, i.e. no input values are stored within the filter.

As a simple example: If event messages, encoded in the IDMEF format (see [3]), shall be transformed in other messages, which have a modified target address value, the according transformation rule is

$$R = (\{E_0^M\}, \varnothing, E^T)$$

i.e. the transformation depends only on the matching of the input event e onto E_0^M, otherwise e is not modified. Now we can define E^M as an XML/IDMEF formatted message with regular expressions (REs) as:

```
<Alert>
 <Target >
  <Node category="dns">
   <Address category="ipv4-net-mask">
    <address>
     192\.22\.[0-9]{1,3}\.[0-9]{1,3}
    </address>
    <netmask>
     255\.255\.255\.255
    </netmask>
   </Address>
  </Node>
 </Target>
 <Classification origin="bugtraqid">
  <name>124</name>
 </Classification>
</Alert>
```

Then E^T could be constructed as

```
<Alert>
 <Target>
  <Node category="dns">
   <Address category="ipv4-net-mask">
    <address>
     191.72.1.1
    </address>
   </Address>
  </Node>
 </Target>
</Alert>
```

If $e(t) \dashv E^M$ (i.e. messages concerning the “teardrop” attack against nodes with an address prefix of “192.22.”) arrives at the filter, a new message e' is generated, which contains all elements of $e(t)$, with the exception of the target address, which will be constantly 191.72.1.1, and the old message is discarded.