Meta IDS Environments: An Event Message Anomaly Detection Approach

Jens Tölle, Marko Jahnke, Michael Bussmann, Sven Henkel
Research Establishment for Applied Science (FGAN)
Computer Networks Dept. (FKIE/KOM)
Neuenahrer Str. 20, 53347 Wachtberg, Germany
{toelle|jahnke|bussmann|henkel}@fgan.de

Abstract

This paper presents an anomaly detection approach for application in Meta IDS environments, where locally generated event messages from several domains are centrally processed. The basic approach has been successfully used for detection of abnormal traffic structures in computer networks. It creates directed graphs from address specifications contained within event messages and generates clusterings of the graphs. Large differences between subsequent clusterings indicate anomalies. This anomaly detection approach is part of an intrusion warning system (IWS) for dynamic coalition environments. It is designed to indicate suspicious actions and tendencies and to provide decision support on how to react on anomalies. Real-world data, mixed with data from a simulated internet worm, is used to analyze the system. The results prove the applicability of our approach.

Keywords: Meta IDS, Anomaly Detection, Event Messages, Graph Clustering

1. Introduction

Networks of cooperating partners are often connected with each other using the public internet. Therefore, these connections are exposed to many different threats. This paper is focused on an anomaly detection approach for event messages in an architecture of an intrusion warning system for dynamic coalition environments.

Within the intrusion warning system, anomaly detection methods are used to detect potential threats which are still unknown and whose signatures are not yet added to misuse databases.

The paper is organized as follows: Section 2 presents basics on cooperating intrusion detection systems and approaches of other researchers. Section 3 describes the structure of our system, followed by section 4 on our anomaly detection approach. Section 5 presents experiences with our approach when used with real-world data. The paper closes with some concluding remarks and a view to future work.

2. Related Work

Several intrusion detection approaches have been proposed, some based on sets of rules or specifications to separate normal from abnormal system behavior (e.g. [1], [2]), others learning normal system status during operation (e.g. [3], [4]).

Some network based anomaly detection approaches use traffic characteristics (e.g. [5]), others inspect payload of the network traffic (e.g. [6]). [7] uses an anomaly detection approach to detect worms propagating via email.

Graph based approaches to intrusion detection have been used since the beginning of the work on the Graph Based Intrusion Detection System GrIDS [8] which also looks at network traffic. This system already contained methods for the detection of worms.

In contrast to approaches which use sensor data as input, others look at event messages or audit trails, such as alert correlation techniques using data mining (see [9]) or they are based on more formal models of the network and its vulnerabilities (e. g. [10]).

Our approach applies graph based anomaly detection techniques to a multi-source event message flow, where no assumptions can be made on the message sources nor on the domain policies they are enforcing.

3. System Structure

Figure 1 presents the general architecture of our IWS prototype. It is based on our IDS infrastructure

Published in: Proceedings of the IEEE International Workshop on Information Assurance, University of Maryland, USA, April 2005.

framework which provides generic pluggable components.

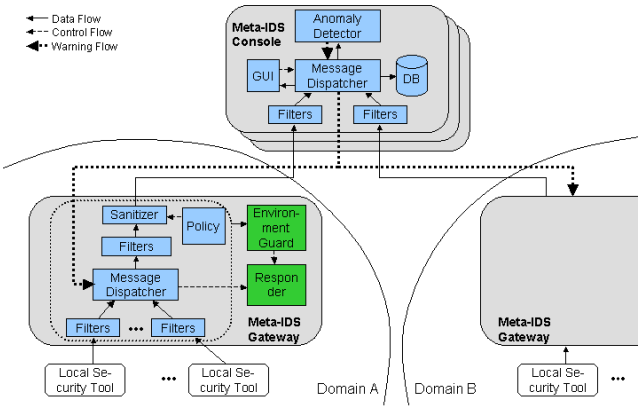

Abb. 2: Feinstruktur des IWS für Koalitionsumgebungen

Figure 1. System structure

The main task of an Intrusion Warning System is to collect and process information about potentially security relevant events from systems locally installed in the domains of the coalition environment. One possible architecture for such a system is the so-called *Meta IDS* which relies on centralized components for data storage and analysis. The term *Meta* is derived from the fact that an additional hierarchy level is introduced to conventional distributed IDS. For our application, we extended the Meta IDS architecture by introducing *Fall-back Consoles* and domain specific *Gateways*. The data model for event messages is given by the IDMEF recommendation [11] of the IETF IDWG. Accordingly, messages are encoded as XML documents and transmitted over network links via the IDXP profile for the BEEP protocol [12]. Any security tool that uses these protocols can be deployed to provide input to the IWS.

The Meta IDS console component processes all messages received from different domains. It contains information storage capabilities as well as different message filtering and processing modules. The central analyzing component of our Meta IDS console is the anomaly detector which is described in the next sections. For avoiding a Single-point-of-failure, a fallback instance of the console is running in the background. In the case of system component or network failures, it takes over all functionality from the main console. Different GUIs are provided for controlling the system, for real-time display of incoming messages and for offline database inspection.

Meta IDS gateways are under the control of the according domain. Their main task is to collect and preprocess all event messages from the event generating security tools that are deployed within the domain. The processed messages are sent over the network to the central system components for further analysis. The following message processing steps can be configured to be performed within a gateway, using extended XSLT [13] processors:

- *Event Normalization*: Messages from different security tools, expressing the same fact using different representations, can be transformed into the same format.
- *Information Sanitizing*: To respect the information sharing policy of the domain, messages can be sanitized, i. e. sensitive information contained within messages can be anonymized or just stripped.
- *Redundancy Filtering*: Multiple "similar" messages can be represented by summary messages; the similarity properties can be configured in a very flexible manner.
- *Offline Detection of pre-defined Event Combinations*: Correlated sets of event messages, pre-defined by according rule sets, can be detected directly in the message stream (i.e. without accessing an event database).

For more information on event message processing in Meta IDS environments, please refer to [17].

4. The Anomaly Detection Approach

Before describing the anomaly detection approach used in our system, it is important to note that anomaly detection approaches have some typical features. One major challenge is the risk of false alarms (*false positives*) and unreported serious events (*false negatives*). This holds for the method described in this paper as well. A careful selection of parameters according to the scenario is necessary to keep both false positive and false negative rates low. As usual, intrusions do not necessarily result in an observable abnormal system behavior, and observed abnormal system behaviour is not necessarily caused by an intrusion.

Therefore, the system described below is not suited as a stand-alone intrusion detection system. It is intended to assist the other intrusion detection methods – mainly misuse detection approaches – used in the different domains. The benefit of the usage is an indication of the health of the supervised networks and

the expectation to discover some kinds of yet unknown threats and threats which are not considered in signature databases of misuse detection systems.

4.1. Basic Ideas

The combination of event messages from different domains is intended to improve detection capabilities. One of the major challenges of anomaly detection approaches located in a centralized Meta IDS console is the fact, that almost no assumptions on the underlying event generating systems can be made.

It is difficult to make assumptions on the type, the quality, or the frequency of event messages coming from the domains. Obviously, the quality and quantity of the incoming event messages depends strongly on the system configuration. One of the main reasons for this is the local administration of the domains and the systems generating the event messages. The local domain security staff is allowed to choose the level of detail of the emitted event messages. This may lead to anonymization of event messages, omitting details or filtering of selected event messages. The domains may use different systems and different products. A result of this is the fact that different domains may send event messages differing in form and quantity as a reaction on the same event.

Nevertheless, the collected event messages can be used to detect abnormal system behaviour. The basic idea for surveying the current system state and for the detection of deviations (anomaly detection) is the continuous monitoring of the incoming event messages.

The method described in the following subsections was originally developed for monitoring network traffic and for detecting network traffic anomalies. The basic idea is described in [5].

4.2. Traffic Structure Recognition

In most networks, the typical structures of network traffic are quite stable. This does not mean that fundamental traffic parameters like data rates are stable. It is well known that these parameters are extremely bursty.

We concentrate on observing the general structure (communication pattern) of network traffic, and fundamental changes in these structures 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 to store it as a traffic matrix. This matrix can be seen as a graph $G = (V, E)$. Nodes $v_i \in V$ of graph G represent communicating devices while edges $e_{i,j} \in E$ represent the communication between device 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 resulting 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. The algorithm takes a data set of inputs and divides them into classes. 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 $G = (V, E)$ into cluster

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

This means that each cluster C_i is a subset of nodes of the node set V. No cluster is empty, and all the nodes belong to exactly one of the clusters.

In this case of application, a clustering algorithm which does not demand a predefined and fixed number of clusters as an input parameter is needed. Suitable clustering algorithms have to detect an appropriate number of clusters on their own.

Sudden variations of the clustering structure are regarded as anomalies. To discover these kind of variation, metrics are needed. Those metrics have to rate similarities (or dissimilarities) of consecutive clusterings $\Re_t$ and $\Re_{t+1}$. The method is not only used to compare current clusterings with their direct predecessors but with a larger set of preceding clusterings. This allows the detection of regular variations in traffic structure. Those variations may be caused by tasks which are executed at regular intervals and cause significant amounts of network traffic differing from the typical network traffic structure. Therefore, these changes in structure should not be regarded as abnormal system behaviour.

4.3. Application to Event Message Context

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 e in the set of

edges e of our message graph G. The edge e 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 ID systems and 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.

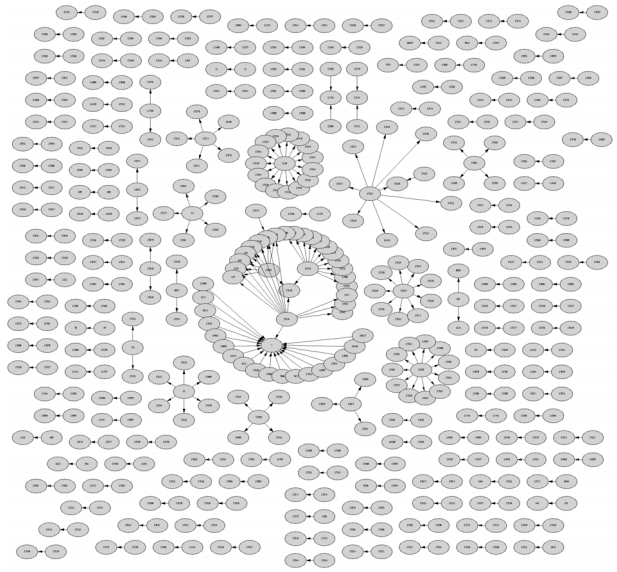

Figure 2. Graph representing the event message structure

The resulting graph describes the typical structure of the incoming event messages.

Figure 2 shows an example of a typical event message structure graph. Each node of this graph represents a unique IP address. Two nodes are connected via an edge if there is at least one entry in the event message database concerning those IP addresses during an analysis interval. The analysis interval used here is 300 seconds.

The circular structures are generated by a set of event messages, where a lot of different IP addresses are the (assumed) originators and one address is always the destination (imagine a distributed attack or an attack with randomly chosen fake source addresses). In this case, the edges point to the central node. Similar structures are generated, when one originator accesses a lot of other nodes (imagine a vulnerability scan). In this case, the edges start at the central node and point to the outer nodes.

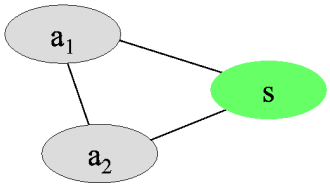

Figure 3. Idea: Graph with additional port nodes and edges

A possible enhancement of this method with no in-depth description in this paper is the additional consideration of extra information contained in parts of the event messages.

Whenever event messages contain information on services (e.g. port numbers), it is possible to enhance the event message graph as follows:

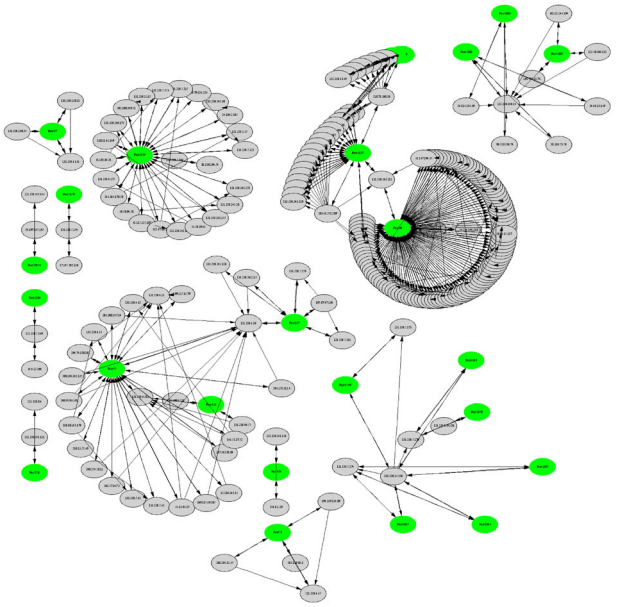

Figure 4. Example graph with additional port nodes and edges

For each event message indicating an activity affecting addresses a_1, a_2 and service s, an additional node representing the service s and two additional edges from a_1 to s and from a_2 to s are added. The figure 3 illustrates this concept, figure 4 shows an application to real world data. This enhancement often allows easy recognition of affected services in case of abnormal system behaviour.

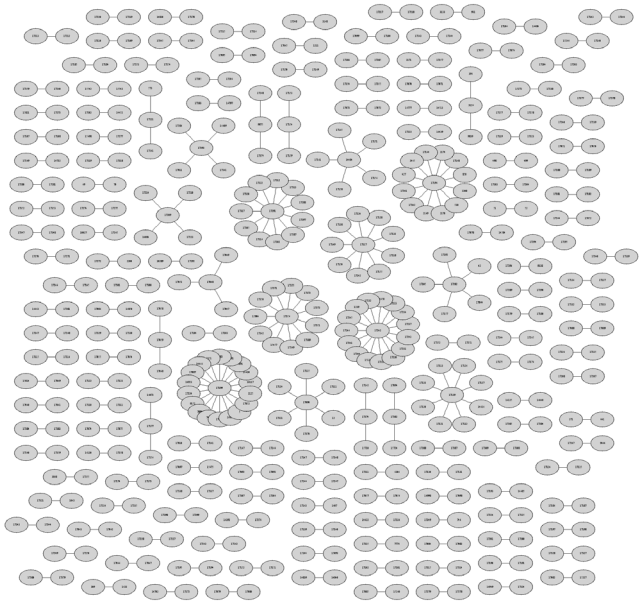

Figure 5. Clustered graph representing the typical message structure

Figure 5 presents the result of an application of a graph clustering method to the graph presented in figure 2. Again each node represents an IP address. Some of the less important edges are removed. This graph now represents the typical event message structure. This structure and its deviations are quite stable in longer time-frames.

Variations from the typical message structure are regarded 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.

The difference (or distance) between two clusterings of a set of objects is calculated as follows. The way of calculating is inspired by the hamming distance, giving a value indicating the difference of two bit strings as the number of elementary operations needed to transform one into the other. In the same manner, our distance d is defined as the number of elementary operations to transform a clustering $\Re_1$ into another clustering $\Re_2$. Elementary operations are splitting of a partition, combining two partitions and moving a subset from one partition to another. Note: The third elementary operation is not really necessary (and therefore *elementary*), because it can be replaced with a sequence of the first two operations.

The formula used in our anomaly detection approach to measure the distance of two clusterings is:

$$d(\Re_1, \Re_2) = |\{(P_1, P_2) \in \Re_1 \times \Re_2 \mid P_1 \cap P_2 \neq \emptyset\}| - \sum_{P \in \Re_1 \sqcup \Re_2} \min\{|\Re_{1|P}|, |\Re_{2|P}|\}.$$

In this formula, it holds $\Re_1 \sqcup \Re_2 := \{P \in 2^M \setminus \emptyset \mid a \in P \wedge \exists P' \in \Re_1 \cup \Re_2 : b \in P' \Rightarrow b \in P, \forall a, b \in M\}$ and $\Re_{|M'} := \{P \in 2^M \setminus \emptyset \mid \exists P' \in \Re : P = P' \cap M'\}$, where 2^M denotes the power set of M.

P_i is a cluster from the clustering $\Re_i$, $i \in \{1, 2\}$. The formula above is a compact way of counting the minimum number of elementary operations to transform the first clustering $\Re_1$ to $\Re_2$. The first part of the formula counts the number of tuples of clusters of consecutive clusterings containing identical nodes. This number is an upper limit for the number of necessary operations. The second part of the formula removes unnecessary elementary operations.

Other means of calculating the distance between two clusterings were developed, e.g. based on the number of nodes belonging to the same cluster in one clustering and to different clusters in the other clustering, or based on a kind of correlation between two clusterings. These methods are also suitable for the purpose of detecting anomalies in the structure of event messages.

5. Results

The implementation was tested with original event message data from one of our networks. Event messages from different security tools (including snort [1], logsurfer [16], examining virus scanner, firewall and other log files) were combined in the Meta IDS and analyzed by the anomaly detection instance.

The structure of our system allows real-time analysis of the incoming event messages. For reasons of simplicity and in order to carry out tests concerning the sensitivity of the anomaly detection methods, we used several databases of recorded event messages, each spanning one week. The results presented in the next subsections are outcomes from one of these databases.

5.1. Worm spreading

The spreading of a worm is a well-known dangerous threat. If newly discovered vulnerabilities are exploited, then signature databases of misuse detection systems need some time to be updated. The distribution of sensors in our Meta IDS allows the early detection of this kind of incidents.

Note: Intended use of this system is in coalition environments. On the one hand, combining event messages of several domains can result in faster detection of potential threats. On the other hand, this method allows to warn the other domains which are possibly not protected against a present attack. This makes it useful to analyze firewall logs even if the own domain is not vulnerable to an attack due to their firewall configuration.

In this example, we focus on active worms. In order to analyze our system, we merge real sensor data from different systems with artificially generated firewall log messages. These firewall log messages are generated using a worm simulator we built for analyzing worm spreading behaviour, pretending worm spreading outside the protected domains.

The firewalls of the domain's networks deny access and log attempted access to non-existent or blocked machines inside the protected networks. Input parameters of our worm simulator are as follows:

- The worm spreading mechanism. In this case, the spreading mechanism of the worm CodeRed v2 (see [15]) is used. The choice of the spreading mechanism has influence on the way a worm instance generates addresses of potential new victims.
- The number of infectable systems. This value has strong influence on the spreading time. We use a value of 360000 infectable systems. This is a realistic number of infectable systems, because it is the approximate number of hosts infected by CodeRed v2. In the future, the number of systems infected by worms could even be higher. A worm attacking an unpatched vulnerability in a widely deployed server software, e.g. a web server, could result in disastrous numbers of infected hosts.
- And in addition: timeout values, if a computer does not answer or is not reachable; number of simultaneous threads looking for infectable hosts; and some more parameters of minor importance.

The simulator contains a supplementary model on the distribution of reachable IP addresses, the region where the corresponding nodes are located and approximated signal delays between network nodes.

5.1.1. Application of the Simulator The parameters chosen for the spreading of a worm result in approximately 7600 artificially generated firewall log messages in addition to 166000 real firewall log messages in the database of several days of life event message data.

These generated log messages are distributed to approximately 96 analysis time-frames (8 hours × 12 time-frames/hour).

The typical structure of the spreading of a worm leads to an extremely low number of additional event messages in the beginning. This early phase of the lifetime of a worm is hard to find, because the total number of eye-catching events is low. Here the Meta IDS approach helps in combining event messages from several domains.

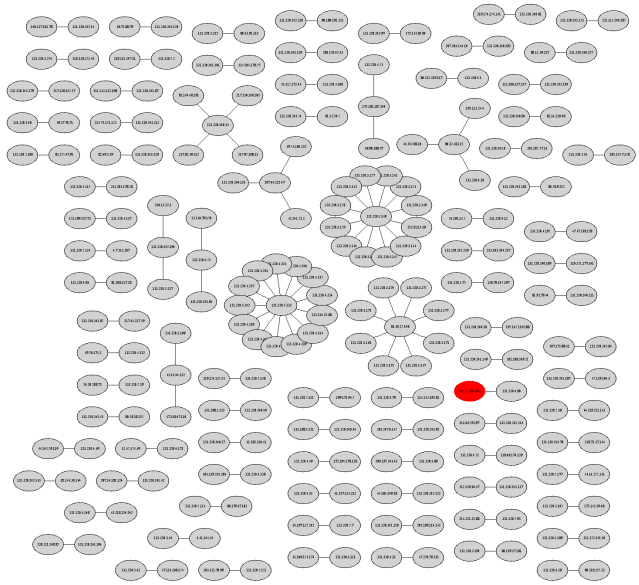

Figure 6. Worm spreading, start

5.1.2. Illustration of Worm Behaviour The graph in figure 6 presents the situation when the first infected node from the outside tries to contact a node inside the protected network. The nodes representing the IP addresses of infected systems are marked in figures 6 to 9. This is done manually to illustrate the behaviour of the worm, it is not part of the anomaly detection system.

In figure 6, you see exactly one node being the first contact between the protected network and the worm. One out of 194 event messages is caused by the worm in this time-frame.

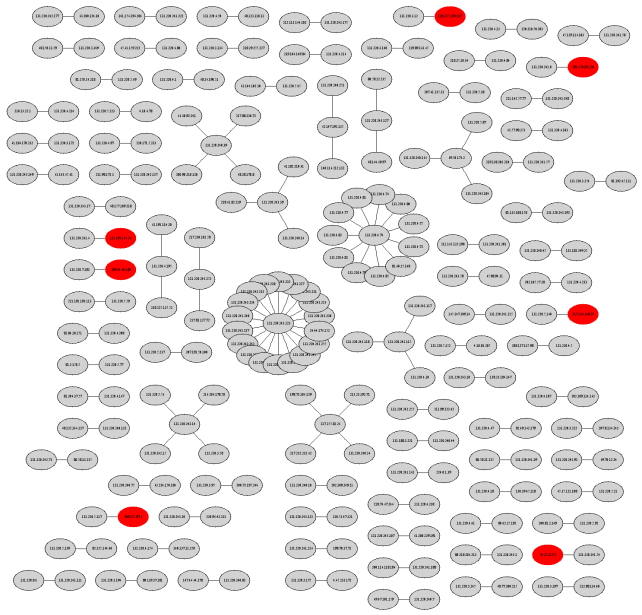

Figure 7. Worm spreading, after one hour

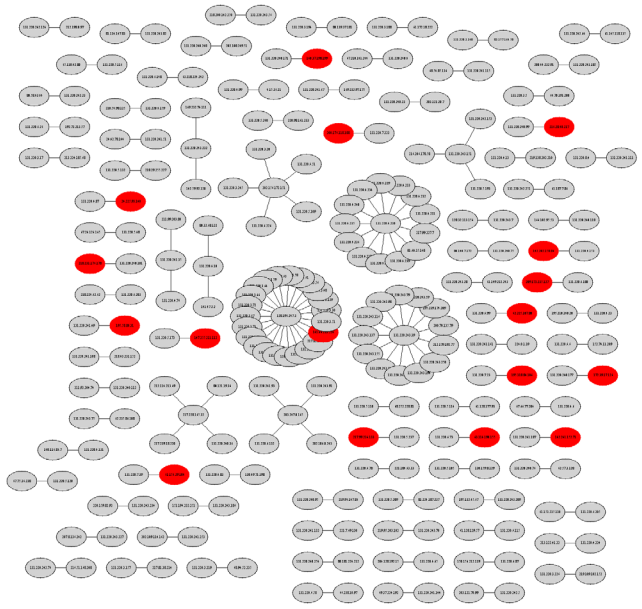

Figure 8. Worm spreading, after two hours

Figure 7 presents the situation one hour (or twelve time-frames) later.

The amount of infected nodes trying to contact the protected network increased. Seven nodes tried to contact systems behind the firewall. The total number of event messages decreased to 181 during this time-frame.

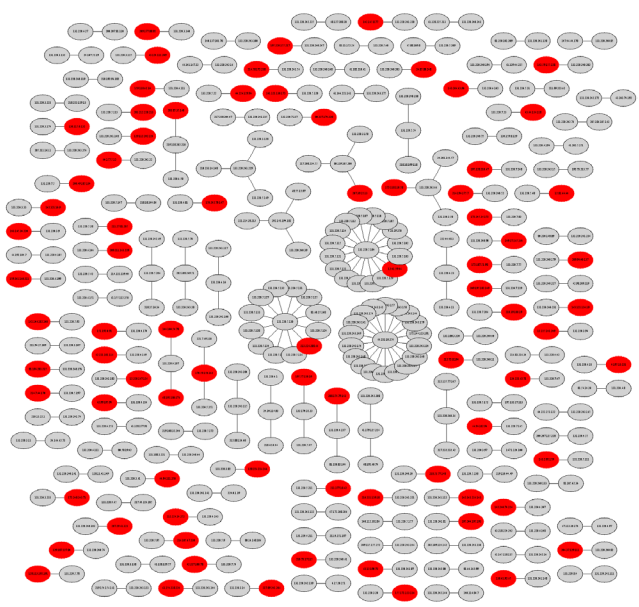

Figure 9. Worm spreading, after three hours

Figure 8 illustrates the situation two hours after the first contact between an infected node and the firewall. The number of infected nodes trying to reach systems behind the firewall has drastically increased.

One hour later, three hours after the first contact with the worm, the situation is even worse. See figure 9.

Two hours after the start of the worm, 185 event messages arrived during the time-frame. This number decreased to 179 messages one hour later.

This change in the structure of incoming event messages is easy to see for a human observer. An early and automatic detection of this kind of variations would be very helpful. Thus, the next section describes the approach to automatic detection of abnormal changes in the event message structure.

5.1.3. Automatic Detection In order to detect abnormal changes in the structure of the event messages, we calculate the distance of (sets of) consecutive clusterings. A visual inspection of the clusterings (for example the left hand sides of figures 6 and 8) does not indicate strong deviations in the event message structure. This seems to be obvious, because the mapping of event messages to our graph mainly adds one-to-one connections.

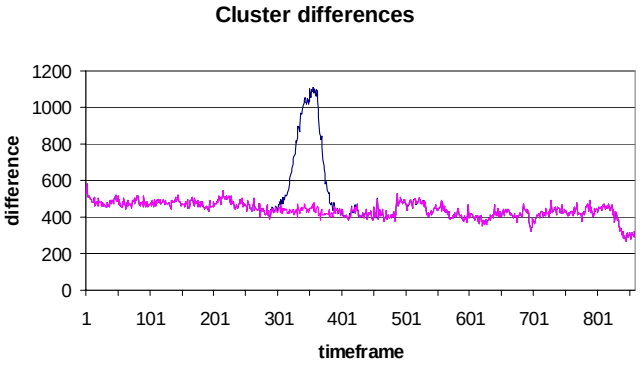

Figure 10. Cluster differences

When considering information concerning the number of new and missing nodes when starting a new time-frame, the distance calculation delivers a strong warning when the worm is spreading. We use an estimation of standard deviation of distances between consecutive graph clusterings. When the distance exceeds the smoothed average plus a factor times deviation, a warning is generated.

The graph in figure 10 presents the distance values between consecutive clusterings. The increase of distances during the worm activities is easy to see. The graph contains both the clean real-world data (lower values during worm attack) and the distance values including the worm attack (peak between time-frames 300 and 400).

5.2. CRC32

The CRC32 attack (see [14]) is an attack against the Secure Shell ssh which is widely used for encrypted communication. Older versions are vulnerable to a special kind of attack. The data base of event messages contains data about an attempted attack on a system under supervision. This attack consisted of a larger number of data packets from the attacker to the attacked node. Each of those packets was reported with an event message, since the potentially dangerous payload contains typical NoOp instructions.

The figure 11 shows the graph during the attack. The typical structure of the event messages has not changed, but the event messages concerning the marked nodes have shown an abnormal increase in frequency of occurrence. Therefore, this was regarded as anomaly and reported. The coloring of the affected nodes is updated automatically when thresholds concerning the weights are exceeded in order to help the administrator to locate

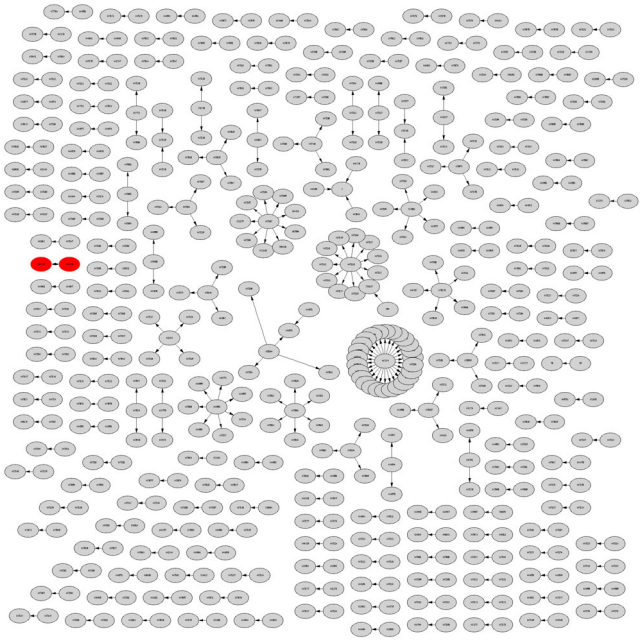

Figure 11. Event message structure during CRC32 attack

security incidents (The visualization of the event message graphs and of the clusterings is performed using Graphviz, see [18]).

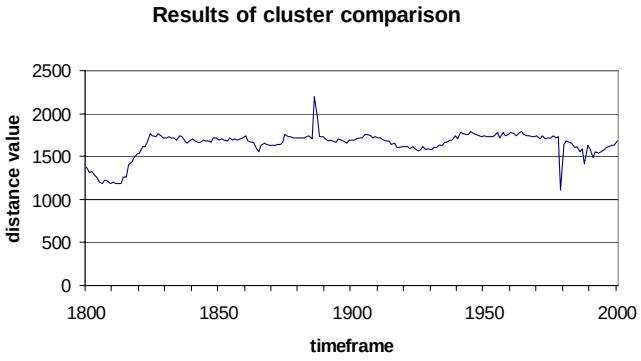

Figure 12. Results of cluster comparison

In this case, the basic comparison of consecutive event message graphs or clusterings is not sufficient to detect these kind of attacks. See figure 12. There is only a slightly higher distance value when the attack was attempted (time-frame 1886). Therefore, other methods have to be used.

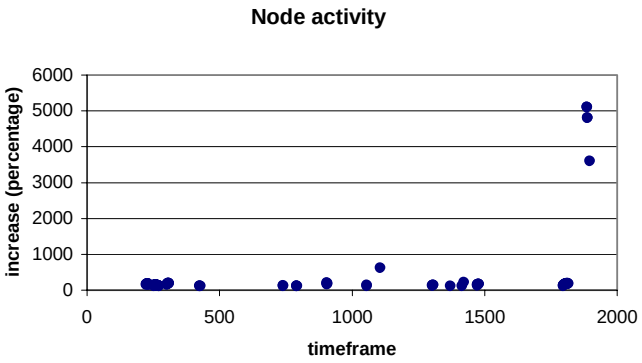

Figure 13. Activity of nodes

Figure 13 presents an analysis of the activity of nodes. A node n is *hyper-active*, if a noticeable number of event messages with the node n as assumed origin or destination are received during one time-frame. If the system discovers an activity which is larger than a fixed value a times the average of the last observations, than a log entry in an activity log file is produced. The graph in figure 13 uses a value of $a = 100$. Each dot in this figure represents one noticeable activity. During the analysis time, we had 94 observations, indicating a larger number of event messages concerning a total of 16 IP addresses (source and destination). Most of the dots indicate just a marginal increase in the number of incoming event messages per time frame. The sudden increase in event messages concerning the two stations being involved in the CRC32 attack is easy to see.

6. Conclusions and Further Work

This paper has presented an approach for detecting anomalies in the event message flow of an intrusion warning system for dynamic coalition environments. It generates graphs from the information about assumed source and target nodes of an action, which is contained within event messages. The algorithm builds clusterings on the graphs for every time-frame. The comparison of clusterings of one graph for subsequent time-frames deliver different distance values; sudden increases of these distance values indicate an anomaly.

According to the requirements for its application, the detection algorithm has the following properties:

1. It is possible to detect largely spread activities like internet worms if traffic related event messages (e. g. from packet filters / firewalls) are available.
2. Activities which cause an unusual number of messages of the same type are also detected (e. g. the SSH CRC32 NoOp attack).
3. Generally, the approach is independent of the event message generating domain-specific security tools, their configuration and the security policy they are enforcing.

For the near-term future, some important things need to be examined further:

- How reliable is the determination of the reason for an anomaly?
- What information can be extracted from an anomaly in order to provide decision support on how to react?
- What happens, if parts of the input messages are sanitized (e.g. anonymized), which might be the case for different application contexts?

These questions will be examined in a multi-domain evaluation environment, operating with real-world data.

7. Acknowledgements

The authors would like to thank Alexander Rink, who was involved in this research work during his time as a master candidate at University of Bonn, Germany.

References

[1] Snort (Open Source Network Intrusion Detection System) homepage – http://www.snort.org

[2] V. Paxson. Bro: A System for Detecting Network Intruders in Real-Time, Computer Networks, 31(23-24), pp. 2435-2463, 1999.

[3] Next-Generation Intrusion Detection Expert System (NIDES) – http://www.sdl.sri.com/projects/nides/

[4] Event Monitoring Enabling Responses to Anomalous Live Disturbances (EMERALD) – http://www.sdl.sri.com/projects/emerald/

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

[6] K. Wang, S. J. Stolfo. Anomalous Payload-based Network Intrusion Detection, Seventh International Symposium on Recent Advances in Intrusion Detection (RAID), September 2004.

[7] A. Gupta, R. Sekar. An Approach for Detecting Self-Propagating Email Using Anomaly Detection. Sixth International Symposium on Recent Advances in Intrusion Detection (RAID), September 2003.

[8] S. Staniford-Chen, S. Cheung, R. Crawford, M. Dilger, J. Frank, J. Hoagland, K. Levitt, C. Wee, R. Yip, D. Zerkle. GrIDS – A Graph-Based Intrusion Detection System for Large Networks. The 19th National Information Systems Security Conference, 1996.

[9] W. Lee, S. J. Stolfo. Data Mining Approaches for Intrusion Detection. 7th USENIX Security Symposium, January 1998.

[10] B. Morin, L. Mé, H. Debar, and M. Ducassé. M2D2: A formal data model for IDS alert correlation. volume 2516, page 115, 2002.

[11] D. Curry, 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-11.txt, January 2004. IETF IDWG.

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

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

[14] CRC32 vulnerability – http://www.kb.cert.org/vuls/id/945216

[15] CERT information on Code Red worm – http://www.cert.org/advisories/CA-2001-23.html

[16] Logsurfer homepage – http://www.cert.dfn.de/eng/logsurf/

[17] M. Jahnke, M. Bussmann, S. Henkel, and J. Tölle. Components for Cooperative Intrusion Detection in Dynamic Coalition Environments. NATO/RTO IST Symposium on Adaptive Defence in Unclassified Networks, April 2004.

[18] Graphviz homepage – http://www.research.att.com/sw/tools/graphviz/