Impact of Sanitized Message Flows in a Cooperative Intrusion Warning System

Jens Tölle, Marko Jahnke
Research Establishment for
Applied Science (FGAN)
Wachtberg, Germany
Email: toelle@fgan.de, jahnke@fgan.de

Nils gentschen Felde
MNM-Team
Ludwig-Maximilians-University
Munich, Germany
Email: felde@mnm-team.org

Peter Martini
University of Bonn
Computer Science IV
Bonn, Germany
Email: martini@cs.uni-bonn.de

***Abstract*— This paper discusses the side effects of sanitizing IT security event messages in a cooperative multi-domain Intrusion Warning System (IWS).**

To enhance detection capabilities of conventional IT security tools like Intrusion Detection Systems (IDS), virus scanners and packet filters, a centralized, so-called Intrusion Warning System can be deployed, which collects and analyzes event messages from the different domains. Additionally, the IWS informs the domains about potentially critical situations which might not be covered by the existing tools due to technical limitations, heterogeneous security policies or differences in configuration. The architecture of an IWS relies on centralized storage and analysis components, while the event messages are collected and preprocessed by distributed entities which are under the operational control of the respective domains.

In cooperation scenarios like military coalition environments (CEs, e.g. NATO, KFOR, SFOR), potentially confidential or sensitive information still needs to be concealed from the CE partners, as defined by existing information sharing policies. This also holds for the information contained in IDS event messages, since there might be specifications of network addresses and topologies, of products or vendors, of applications and security systems included in the messages. Thus, for enabling a CE wide cooperation of IT security systems, appropriate information sanitizing techniques need to be applied before sharing any security relevant information. This might lead to a negative impact on the centralized analysis capabilities, since potentially important information might be dropped from the messages.

In this paper, the impact of sanitizing event message flows in a cooperative IWS is studied by examining the behaviour of an IWS when feeding it with real-life event messages combined with artificial events from an internet worm spreading simulation. The worm detection capabilities of the analysis components are determined in a multi-domain setup for both situations, with and without applying information sanitizing mechanisms on the event message flow.

I. Introduction

Over the last years, Intrusion Detection Systems (IDS) have become more and more important in order to increase network security. The importance of these systems is pointed out by the great amount of effort that has been spent in research and development to improve IDSs.

Nowadays, cooperative approaches for intrusion detection are developed and their capabilities are studied. In cooperative IDS architectures, several parties share information about security relevant events and analyze the combined data in order to gain benefit from what has been detected by other domains. One possible architecture is a so-called Intrusion Warning System (IWS) that collects and preprocesses messages received from the locally deployed security tools in order to store and analyze the messages on centralized system components which are operated by a neutral trustworthy party.

In these multi-domain environments, trust is a crucial aspect. Sharing security relevant data may reveal leaks or weaknesses in deployed security systems and expose confidential information (e.g. network topology, deployed products and vendors). Besides, existing information sharing policies have to be obeyed. Thus, sanitizing parts of event messages – this means altering or even deleting messages sent out to the cooperative system – is an important issue for these kinds of cooperative security systems.

In this paper, we study the impact of sanitizing security event messages in an existing implementation of a multi-domain IWS. We have evaluated the impact on the detection performance by making the IWS analyze real-life data combined with artificial events, generated by an internet worm spreading simulation that mimics the specific characteristics of some real-world internet worms we observed during the last years. The investigations have been performed with a varying number of cooperation parties in different constellations of the CE. Additionally, different strategies for sanitizing messages within the IDS have been considered and the limits of the underlying cooperative detection approach have been examined.

The paper is organized as follows. First, related work in the area of distributed and cooperative IDSs is discussed in section II. In section III, the basics of the IWS and the scenario in which it has been deployed are described in detail. Further on, we present the procedure of the evaluation. In section IV, results are presented and will be discussed and explained in detail in section V. Finally, section VI summarizes and concludes the results of this work.

II. Related Work

Generally, distributed security mechanisms aim at detecting distributed large scale attacks on the network infrastructure and its components. Therefore, many novel frameworks and architectures have been proposed (e.g. [9], [6]). The majority of them focuses on securing certain elements of the network

topology such as for example routing protocols [1]. An important goal of distributed IDSs is trying to detect distributed Denial of Service attacks (DDoS, [7] [11]).

Mirkovic, Robinson and Reiher suggest a system called *DefCOM* [7]. DefCOM is a distributed framework that enables the exchange of information among several different security mechanisms and services. This attempt has been proposed to overcome the problem of heterogeneity of existing systems which have already been deployed. The work enables cooperation of different parties uniting their security instances for the improvement of detecting widely distributed attacks.

Su and Ju describe an attempt to detect massively distributed denial of service attacks in a cooperative way in order to respond appropriately [11]. They developed a credibility algorithm to evaluate the trustworthiness of shared messages in order to improve the quality and precision of detection. The proposed IDS works in a cooperative manner meaning that multiple administrative domains are involved and extensive cooperation among the local IDS instances is necessary.

When cooperating in multi-domain environments, sanitizing event messages is necessary. Thus, mechanisms like *TCPdpriv* [13] have been developed. TCPdpriv is a free tool which performs a prefix-preserving IP address anonymization. IP addresses are mapped to pseudo-random anonymized IP addresses using tables. Xu, Fan, Ammar and Moon present an IP address pseudonymizer [15] [16] which is using short keys in order to pseudonymize IP addresses instead of tables.

Besides anonymization of IP addresses, Pang and Paxson present an anonymization methodology on application level [8]. Their proposal anonymizes both packet payloads and transactional information. Therefore, the system has to be aware of different application layer protocols such as HTTP, FTP or SMTP for example.

This paper aims at applying mechanisms for sanitizing event messages in a cooperative intrusion warning system and evaluating the impact of different sanitizing techniques onto the detection performance of the IWS. The work mentioned above focuses on either intrusion detection and warning or various information sanitizing techniques. None of the work has yet investigated on the effects of information loss on cooperative IWS in federated environments, such as i.e. given in a military context. Thus, we will focus on the impact of sanitizing event message flows in the remainder.

III. Testing Scenario

In this section, we describe our evaluation scenario. Further on, we will explain the anomaly detection mechanism in detail. Afterwards, the procedure of sanitizing messages within the system and the method of how to evaluate the impact of sanitized messages on the IWS is discussed.

A. IWS Setup

The scenario in which we applied and evaluated the IWS is shown in Figure 1. In this setup, three domains participate in connecting their local IDS instances to the common centralized IWS (the *Meta IDS*) in order to collect locally generated event messages and analyze them.

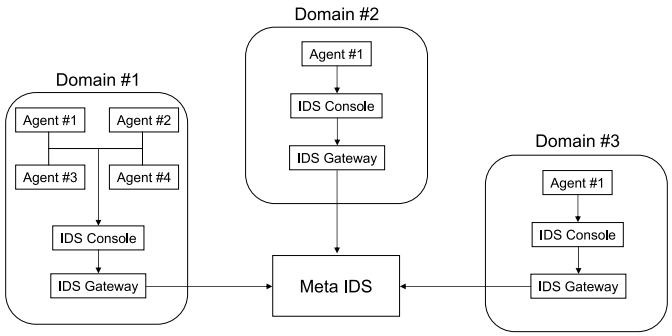

Fig. 1. General setup of the IWS and underlying IT security tools

In this case, each domain collects its security relevant information using a local installation of a conventional distributed IDS with an arbitrary number of agents and sensors. The setup of the agents, sensors and their deployment location within the networks is left to each domain's administrator's decision. Therefore, the central system has no prior knowledge of the local networks, their topologies, their security policies and the policy enforcment tools.

All messages generated by local agents are passed to their so-called *IDS Console*. This component evaluates, stores and presents the collected information to the security administrator. It has no additional information, neither about other cooperating domains, nor about their security policies and tools.

Messages are transferred within the system using the Intrusion Detection Message Exchange Format (IDMEF) [5], which is an XML-based format, in order to guarantee system interoperability and reusability. Authorization and authentication is realized using a public key infrastructure (PKI) and the TLS/SSL protocol (transport layer security/secure socket layer) as specified by the Intrusion Detection Exchange Protocol (IDXP) [17]. For more details on the implementation of the conventional distributed IDS, please refer to [6].

All messages received by the IDS console are passed on to the *IDS Gateway*. The gateway is responsible for sending the messages to the Meta IDS that preprocesses, collects and analyzes the information coming from all participating members. At this point, information sharing policies need to be enforced. The gateway offers the possibility to drop or alter messages using XML transformation capabilities. Within the gateway, regular expressions can be defined as part of the so-called *matching templates*, and if one of these regular expressions matches parts of an incoming message, it is modified according to the *transformation templates* specified in advance. In this case, transformation may include deleting or altering the information in a context sensitive manner before passing it on. The technique is called *Conditional Pattern Matching and Transformation* (CPMT) and it is implemented using XSLT (eXtensible Stylesheet Language Transformations). More details can be found in [4] and [14].

B. Anomaly Detection Scheme

The centralized Meta IDS has several capabilities to evaluate the incoming data. Besides a detection engine for pre-defined event correlations and an analysis of message flow parameters, an anomaly detection approach is used to detect deviations from a previously learned behavior profile. Further evaluation procedures are focused on the according graph based anomaly detection module called CBAD ("**C**luster-**B**ased **A**nomaly **D**etector"), as described in [4].

CBAD collects incoming event messages during a period of time Δt. The length of Δt has to be specified in advance (in this case $\Delta t = 300$ seconds). According to the content of the messages collected during an interval t_i of length Δt, a graph $G = (V, E)$ is constructed. The vertices V of the graph G are associated with the IP addresses mentioned by the IDMEF messages. A directed and weighted edge $(v_1, v_2) \in E$ between two nodes $v_1, v_2 \in V$ is established, if a security relevant message within the current interval contains both the IP address associated with $v_1 \in V$ as (probable) source of the event and the one associated with $v_2 \in V$ as destination. The weight of the edge is adjusted with respect to the severity of the event message. Additionally, a third node $v_3 \in V$ is created and associated with the TCP/UDP port mentioned in the IDMEF message and bidirectional edges $(v_1, v_3), (v_2, v_3) \in E$ are introduced as well. This procedure allows the investigation for the reason of a detected anomaly later on.

At the end of the interval t_i the resulting graph G_i is clustered using graph clustering algorithms. In this case, we use the "*MajorClust*" algorithm as described in detail in [12]. Figure 2 shows the clustering algorithm in pseudocode.

MAJORCLUST
Input: Graph $G = (V, E)$
Output: Function $c : V \mapsto N$, which maps each node $v \in V$ to a cluster

(01) $n = 0, t = false$
(02) $\forall v \in V$ DO
(03) $\quad n = n + 1$, $c(v) = n$
(04) WHILE $t = false$ DO
(05) $\quad t = true$
(06) $\quad \forall v \in V$ DO
(07) $\quad\quad$ IF $|\{u : \{u, v\} \in E \wedge c(u) = i\}|$ is max THEN
(08) $\quad\quad\quad c^* = i$
(09) $\quad\quad$ IF $c(v) \neq c^*$ THEN
(10) $\quad\quad\quad c(v) = c^*$, $t = false$
(11) $\quad$ END
(12) END

Fig. 2. Pseudocode "Majorclust", see [12]

This results in a clustering that partitions the nodes of the original graph to subsets. Members of the same cluster are strongly connected in the original graph (i.e. the edges have a high weight), while weaker connected nodes of the original

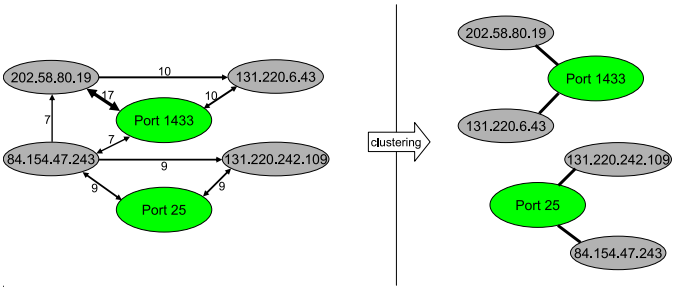

Fig. 3. Example of a graph and its clustering (no real-life data)

graph are assigned to different clusters by the clustering process. This procedure reveals the basic structure of the original graph constructed from the message flows. This allows to detect *sudden changes* in the structure that may be interpreted as anomalies, as described in detail below.

Figure 3 shows an example of a graph and its clustering. On the left hand side a sample graph is shown with nodes representing both IP addresses and ports. The nodes are connected by directed and weighted edges. The result of the clustering algorithm is presented on the right hand side. Examples for clusterings of real-life data can be found in Figures 8 and 9.

For the detection of *sudden changes* between subsequent graph clusterings, a distance between clusterings needs to be defined. In this case, the distance between two consecutive clusterings C_i and C_{i+1} is defined as δ_i, which is the number of elementary changes that have to be performed in order to transfer clustering C_i into C_{i+1}. Elementary changes are to split off a node from its cluster or to merge a node with another cluster. Each node that does not exist in both of the clusters C_i or C_{i+1} increases the value of δ_i by one as well.

The definition of a *sudden change* in the structure of the clusterings is as follows: If δ_i exceeds an acceptance interval resulting from the smoothed average $\overline{\delta}$ of the past clustering differences, which means $\delta_i \notin [\overline{\delta} - d \cdot \sigma_{\overline{\delta}}, \overline{\delta} + d \cdot \sigma_{\overline{\delta}}]$ (d is a constant defining the width of the acceptance interval and in our study $d = 4.5$ proved to be a suitable value), $\sigma_{\overline{\delta}}$ being the standard deviation of the smoothed average $\overline{\delta}$, the structure of the message stream is interpreted as abnormal. Therefore, an anomaly report will be generated and published. If $\delta_i \in [\overline{\delta} - d \cdot \sigma_{\overline{\delta}}, \overline{\delta} + d \cdot \sigma_{\overline{\delta}}]$ the current situation is considered to be normal. A more detailed description of the amonaly detection technique can be found in [2].

C. Evaluation Method

In the scenario, we have three different domains sharing event messages in a cooperative manner, using IDS gateways for passing and filtering the messages (see 1). By putting the gateway under the control of the security staff of the according domain, it is possible to enforce the information sharing policies directly at the domain's border. In this work, altering address specifications in messages and its impact on the analysis of the data is considered and evaluated. A very

simple example for sanitizing IPv4 addresses is to delete their last two bytes. In this paper all IP addresses belonging to one of the cooperating parties are anonymized this way, e.g. the following mappings would be applied using this anonymization technique:

$$123.123.123.123 \mapsto 123.123.--.--$$

as well as

$$123.123.234.234 \mapsto 123.123.--.--$$

Thus, all IPv4 addresses with the same first two bytes will be mapped to the same anonymized pseudo IP. Since the addresses are the most important part of the messages with regard to the anomaly detection approach examined here, the impact on the detection performance is significant.

The IWS has been deployed in the scenario shown in Figure 1 and a stock of real-life data has been collected for further study. For evaluation purposes messages from a simulated spreading of internet worm instances (see [3]) have been injected into the stream of messages. These messages would have been generated using regular firewall logging mechanisms during the time of the global spreading of real-life worms. In order to achieve close-to-reality-situations, *rate limiting* functions (like the ones that are used in hardware firewall packet filters) have been implemented in order to reduce the amount of information. In this case rate limiting means that a *maximum* of one message per second entering the stage of common data evaluation (*1Hz rate limiting*) is guaranteed. If more messages arrive at the system, an adequate amount is dropped randomly in order to obey the maximum rate.

The basis for the upcoming evaluation is the point in time where the manually inducted anomaly can be detected in both situations, with and without sanitization. Further, the influence of the number of coalition partners participating has to be investigated. Therefore, different constellations of parties cooperating in collecting and examining data have been considered. Additionally, a lower bound for the amount of information needed in order to detect anomalies properly was determined.

D. Worm Model

For evaluation purposes, we are in need of information about the spreading of worm instances. In this case we used a simulation of the *Scalper* worm, as examined in [3]. Other worm propagation algorithms such as for example various spreading strategies of different CodeRed variations have been tested as well. The simulation delivers information about the attempts of worm instances to access arbitrary IP addresses in order to infect the corresponding systems and thus generates log files the way firewalls would have created in case of a real-life worm spreading.

FreeBSD.Scalper or just *Scalper* [10] tries to find new infectable systems by scanning ranges of IP addresses systematically. Therefore, the first two bytes of potentially new targets are randomly generated before the last two bytes are scanned systematically. This leads to easily detectable phenomena in case of not sanitized message flows in the cooperative IDS. But despite that fact, the type of sanitization chosen here (deleting the last two bytes of each IP address mentioned and being associated with one of the partners' IP address space, see section III-C) will make the anomaly detection much more difficult, as IP addresses are crucial for the anomaly detection approach used here (see section III-B).

In the following section, typical results of the measurements taken are presented and the limits in detecting the inducted anomaly caused by Scalper are pointed out.

IV. Results

The address spaces of the three domains examined in this scenario are different in their size; one is a complete class B network, while the others have a smaller range. This is a very important aspect when examining an internet worm spreading because of the different numbers of firewall hits obtained from worm instances trying to find the next vulnerable computer system. Having a large IP address space will cause more attempts in comparison to a smaller address range because the probability of being targeted by active worm instances is higher.

In this section we will present the results obtained by introducing a simulated spreading of Scalper. First, we will show the plain results without sanitizing any messages before we proceed using sanitized message streams. After that, we will point out the limits of the system by showing an example in which a false negative is obtained. In all the cases presented here, rate limiting functionalities with a maximum message rate of 1Hz are enabled as described in section III-C.

A. Non-Sanitized Message Flows

Figure 4 shows the evaluation of the data retrieved from all three cooperation partners with injected messages of a simulated spreading of the Scalper worm. All incoming messages are not modified, meaning that especially no sanitization or message hiding takes place.

In this case the anomaly caused by Scalper is represented by the high peak beginning at 12:39 p.m (timeframe 186). Additionally, two minor exceedings of the acceptance interval can be observed in timeframe 136 and 157 before the impact of Scalper is recognized. These two smaller anomalies result from one out of the three sets of real-life data collected in advance in which an anomaly can be found. These anomalies do not influence the measurements because the simulated data is introduced into the message stream after these two anomalies have taken place.

B. Sanitized Message Flows

The evaluation of the information gathered from all three participating parties while sanitizing incoming messages (see section III-C) is shown in Figure 5. The anomaly caused by the injected event messages about the spreading of the Scalper worm can still be observed clearly, while being less distinctive than in the previous case (Figure 4). The anomaly starts at

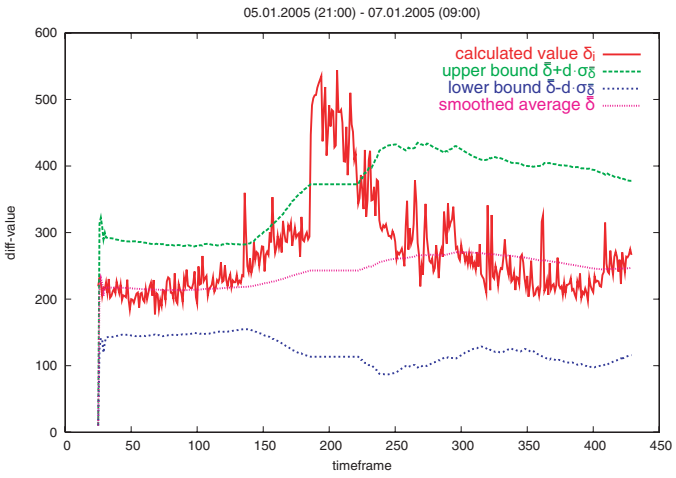

Fig. 4. Three parties with **non**-sanitized message flows

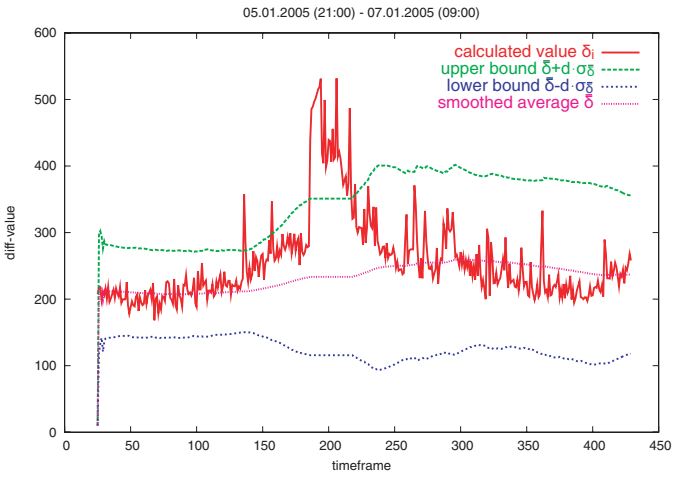

Fig. 6. Two parties with **non**-sanitized message flows

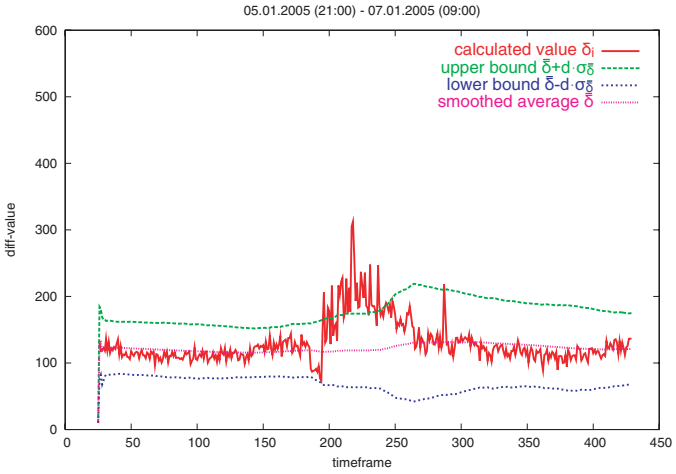

Fig. 5. Three parties with sanitized message flows

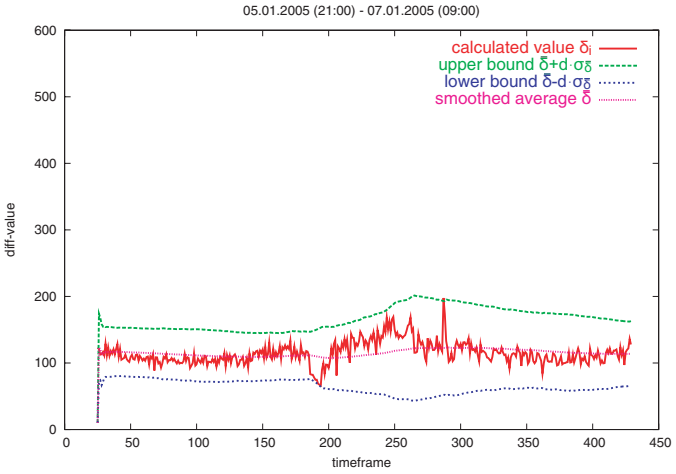

Fig. 7. Two parties with sanitized message flows

02:25 p.m (timeframe 207). In comparison to the previous case, the system has detected the worm approximately two hours later. Additionally, in contrast to Figure 4, the detection algorithm did not report the two smaller anomalies at the beginning of the set of data.

C. Limits of the System

In this section, we will present a case in which the IWS fails to detect the artificially inducted anomaly in order to point out the limits of the system. The set of data used in this case comes from only two of the participating domains. Figure 6 shows the analysis of the set of data with not sanitized messages while Figure 7 shows the analysis of the same data with sanitized information.

In Figure 6, one can see the high peak originating from the injected messages about the spreading of Scalper. Further, two smaller peaks can be observed. They are a result of an anomaly contained in one of the real-life data sets as already described above. Again, these two anomalies do not influence the further evaluation.

In Figure 7, all three anomalies that could be detected before, are not visible any longer. Despite that fact, a great decrease in the clustering differences can be observed. Additionally, the clustering differences are much more stable. This results in a smaller acceptance interval and finally leads to a situation interpreted as regular. It is not being reported as an anomaly by the anomaly detector.

Considering Figure 4 and 5, the observations regarding the differences between sanitized and non-sanitized message stream holds. But in contrast to the case considered in this section, the anomaly caused by Scalper could still be detected.

For further discussion and an explanation of the main causes for this behavior see the next section.

V. Discussion

In the previous section, results have been presented that show two different configurations of cooperating parties with both sanitized and non-sanitized message streams evaluated by the cooperative IWS.

Comparing the two results of three parties working together (Figure 4 and 5) one can clearly observe that in both cases the artifically inducted anomaly could be detected. Taking a closer look at Figure 4 and 5, one recognizes that the anomaly is much more obvious in the case of not modified data (Figure 4) than in the case of sanitized information (Figure 5). Besides, the detection of Scalper was delayed by nearly two hours when using sanitized message flows. This leads to the conclusion that in this case the loss of information just leads to a loss of performance in identifying anomalies.

Taking a look at the level of the measurements taken every timeframe (namely the clustering differences, refer to section III-B) one recognizes that sanitization leads to much more stable measurements with smaller values. An explanation can be given by taking a closer look at the type of sanitization considered in this case. Because of deleting the last two bytes of each IP address associated with one of the cooperating partners the number of total nodes in the resulting graphs and thus in the clusterings decreases tremendously. Figures 8 and 9 show two clusterings of a sample set of data from exactly the same point in time. The lighter nodes represent IP addresses while the darker nodes stand for the corresponding ports. Figure 8 shows the clustering resulting from non-sanitized messages, while the same situation is presented in Figure 9, but sanitizing all relevant data first.

The most noticeable difference is the significantly higher number of nodes in figure 8. The lower total number of nodes in figure 9 is caused by mapping larger sets of different IP addresses to only one anonymized address per set. The total number of three darker nodes representing port numbers does not differ in both graph clusterings, because only IP addresses are sanitized. Due to the more complex structure of the initial communication graph, the clustering in figure 8 consists of two clusters, one representing a larger set of IP addresses of systems communicating via port 135, and a second cluster representing stations communicating via ports 25 and 21. The much simpler graph of sanitized IP adresses leads to a single cluster representing all communication activities in figure 9.

It is obvious that even in this very small example the number of vertices decreases, resulting in a much simpler and smaller structure of the clustering. For that reason the clustering differences decrease in comparison to the case of unmodified information and the values are stabilizing which complicates the detection of anomalies dramatically.

Finally, information hiding may lead to false negatives, meaning that existing anomalies cannot be detected any more. An example has been presented in section IV-C and its corresponding Figures 6 and 7. A typical situation of a false negative is pointed out, which in this case leads to an even worse situation than just not detecting the anomaly. As one can see, the current value of clustering differences does not exceed the upper boundary of the acceptance interval, but instead the slowly increasing clustering differences enlarge the acceptance interval. This situation makes it even more complicated to detect anomalies later on.

To solve this problem, the very tight restrictions put upon

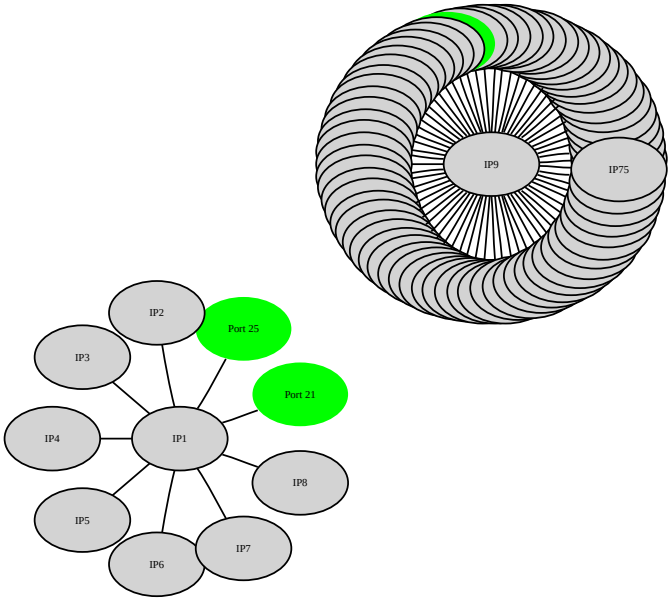

Fig. 8. Comparison of clusterings – **non**-sanitized messages

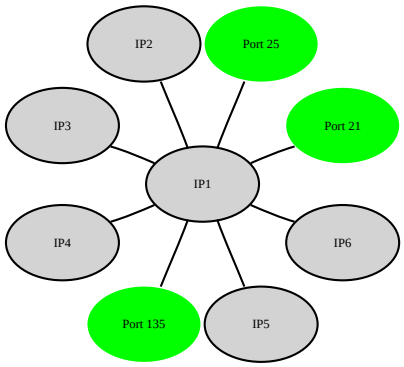

Fig. 9. Comparison of clusterings – sanitized messages

the system could be loosened a little. By increasing the maximum rate of messages to above 1Hz for example, the detection of the inducted anomaly is possible again. This highlights that we have reached the lower boundary in regard to the restrictions that can be put upon the Meta IDS. This case has been reproduced in different situations with different types of information hiding strategies, such as deleting different parts of IP addresses and/or statically mapping ports/IPs. Besides, other types of anomalies like various CodeRed spreadings have been evaluated as well and the results presented here have been proven to be representative.

VI. Conclusion and Further Work

This paper describes the effects of sanitizing event messages in a multi-domain coalition environment, where locally deployed Intrusion Detection Systems are cooperating in a so-called Intrusion Warning System (IWS). After motivating the necessity for anonymizing and pseudonymizing IDS messages, the Meta IDS architecture of the examined IWS has been

described, followed by a detailed description of the deployment scenario. We described our evaluation environment and procedures – based on simulations of worm spreading effects – as well as the impact on the detection. Further on, we discussed our results, explained the reasons for these phenomena and pointed out the limits of our IWS approach.

We have shown that sanitization of event message streams in cooperative IDS – which is a basic requirement in many deployment scenarios – is generally applicable without significant loss of detection capabilities. Many operational parameters of the underlying IDSs need to be adjusted carefully (e.g., the message emission count of packet filters). The loss of information as a result of the sanitization procedure may lead to a delay of the detection time. In the examined internet worm spreading process with three cooperating IDS instances and based on real-life data, the detection delay was around two hours.

It is worth to mention that local detection capabilities are not influenced by sanitizing event messages. Thus, global detection with sanitized data can only improve and never degrade the overall (i. e. combined) detection capabilities compared to scenarios without multi-domain IWS.

As further research activities, large-scaled evaluation setups should be examined where more than 10 IDS instances are cooperating. Additionally, the effectiveness of detecting other anomalies beside worm spreadings – especially small-scaled and synchronized attacks – need to be evaluated.

REFERENCES

[1] S. Cheung and K. Levitt: "Protecting Routing Infrastructures from Denial of Service Using Cooperative Intrusion Detection", Proceedings of the 1997 workshop on New security paradigms, p. 94-106, ACM SIGSAC, Langdale, Cumbria, United Kingdom, 1997.

[2] N. gentschen Felde: "Leistungsfähigkeit von Anomalieerkennungsverfahren in domänenübergreifenden Meta-IDS", Master's thesis, University of Bonn, Germany, 2005.

[3] H. Schmidt: "Simulation und Erkennung der Ausbreitungsstruktur von Würmern", Master's thesis, University of Bonn, Germany, 2002.

[4] M. Jahnke, J. Tölle, M. Bussmann, and S. Henkel: "Components for Cooperative Intrusion Detection in Dynamic Coalition Environments", in: Proceedings of NATO/RTO IST Symposium on Adaptive Defence in Unclassified Networks, Toulouse, France, 2004.

[5] The Internet Engineering Task Force - Intrusion Detection Working Group: "The Intrusion Detection Message Exchange Format", 2006. `http://www.ietf.org/internet-drafts/draft-ietf-idwg-idmef-xml-16.txt`

[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, 2002.

[7] J. Mirkovic, M. Robinson, P. Reiher, and G. Kuenning: "Alliance Formation for DDoS Defense", in: Proceedings of the New Security Paradigms Workshop, ACM SIGSAC, 2003.

[8] R. Pang and V. Paxson: "A High-Level Programming Environment for Packet Trace Anonymization and Transformation", ACM SIGSOMM, 2003.

[9] S. Snapp, J. Brentano, G. Dias et al.: "DIDS (Distributed Intrusion Detection System) – motivation, architecture, and an early prototype", in: Proceedings of the 14th National Computer Security Conference, 1991.

[10] Symantec Security Response: "FreeBSD.Scalper.Worm". `http://securityresponse.symantec.com/avcenter/venc/data/freebsd.scalper.worm.html`

[11] H. Su and J. Ju: "Cooperative Detecting and Responding to a DDoS Attack", Parallel and Distributed Computing and Systems – PDCS 2004, MIT Cambridge, USA, 2004.

[12] Benno Stein, Oliver Niggemann: "On the Nature of Structure and Its Identification", Dept. of Mathematics and Computer Science, University of Paderborn, "Lecture Notes in Computer Science", Springer Verlag, pages 122-134, Berlin Heidelberg New York, 1999.

[13] TCPdpriv `http://ita.ee.lbl.gov/html/contrib/tcpdpriv.html`

[14] J. Tölle, M. Jahnke, M. Bussmann, and S. Henkel: "Meta IDS Environments: An Event Message Anomaly Detection Approach", in: Proceedings of IEEE Workshop on Information Assurance, University of Maryland, USA, 2005.

[15] J. Xu, J. Fan, M. Ammar, and S. Moon: "On the Design and Performance of Prefix-Preserving IP Traffic Trace Anonymization", ACM SIGCOMM Internet Measurement Workshop, 2001.

[16] J. Xu, J. Fan, M. Ammar, and S. Moon: "Prefix-Preserving IP Address Anonymization: Measurement-based Security Evaluation and a New Cryptography-based Scheme", IEEE International Conference on Network Protocols (ICNP), 2002.

[17] The Internet Engineering Task Force - Intrusion Detection Working Group: "The Intrusion Detection Exchange Protocol (IDXP)", 2002. `http://www.ietf.org/internet-drafts/draft-ietf-idwg-beep-idxp-07.txt`