Adaptive Load Allocation for Combining Anomaly Detectors Using Controlled Skips

Mario Berger*[†], Felix Erlacher[†], Christoph Sommer[†] and Falko Dressler[†]
*Barracuda Networks, Innsbruck, Austria
[†]Computer and Communication Systems, Institute of Computer Science, University of Innsbruck, Austria
{mario.berger,erlacher,sommer,dressler}@ccs-labs.org

***Abstract*—Traditional Intrusion Detection Systems (IDS) can be complemented by an Anomaly Detection Algorithm (ADA) to also identify unknown attacks. We argue that, as each ADA has its own strengths and weaknesses, it might be beneficial to rely on multiple ADAs to obtain deeper insights. ADAs are very resource intensive; thus, real-time detection with multiple algorithms is even more challenging in high-speed networks. To handle such high data rates, we developed a controlled load allocation scheme that adaptively allocates multiple ADAs on a multi-core system. The key idea of this concept is to utilize as many algorithms as possible without causing random packet drops, which is the typical system behavior in overload situations. We developed a proof of concept anomaly detection framework with a sample set of ADAs. Our experiments confirm that the detection performance can substantially benefit from using multiple algorithms and that the developed framework is also able to cope with high packet rates.**

I. Introduction

Intrusion Detection Systems (IDS), dedicated network components that can alert a security administrator of potential attacks, have become more and more more popular in the last decade [1]. For detecting network traffic that might be part of an attack, traditional IDS often use a signature-based approach relying on rule sets [2]. However, if a novel attack has not yet been described in form of such rules, it is impossible to detect the malicious traffic. Therefore, Anomaly Detection Algorithms (ADAs) have been studied to identify network traffic deviating from normal behavior. Because attack traffic frequently shows different behavior, such traffic often also produces some kind of anomalies [1], [3]. Thus, ADAs can also help in detecting attacks on a network.

Many different ADAs have been developed for different types of anomalies, each having its own advantages and drawbacks [4]–[11]. In practice, it seems reasonable to make use of multiple ADAs for identifying attack traffic and combine their advantages. Because a detected anomaly can only be considered as a “suggestion” for ongoing attacks, further post-processing techniques are needed. Examples include ranking of anomalies by severity, automated forensics, or recording for later inspection by security experts.

The focus of this work, is on using multiple ADAs on one multi-core machine, together with controlled load allocation. To the best of our knowledge, there is no system available that extensively makes use of multiple ADAs on one machine in the contexts of high speed network monitoring and intrusion detection. The advantage of combining ADAs on one machine in comparison to clustering them over a network is that there is no network flow distribution offset.

Because anomaly detection can be very resource-intensive, especially in high-speed networks, a controlled load allocation scheme is required, which takes into account the complexities of different ADAs. In particular, slower algorithms using more complex operations should not prevent faster algorithms from being applied. Algorithms may be skipped if no more processing resources are available in order to ensure real-time monitoring in high-speed networks. The framework also features techniques for filtering anomalous traffic by certain criteria and for gathering statistics to evaluate ADAs that have been integrated into the framework.

We developed a modular framework supporting anomaly detection in computer networks. Our system has been implemented as part of Vermont, which is a highly flexible software toolkit used for network monitoring [12]–[14]. Vermont, being equipped for high speed flow monitoring, also provides all the necessary functionality to run single modules on different CPU cores in parallel. Our adaptive load allocation algorithm extends Vermont by integrating load distribution techniques using controlled skips. We integrated two well-known anomaly detection algorithms that we used in our experiments.

Our key contributions can be summarized as follows:

- As a novel concept, we make use of multiple anomaly detection algorithms on one machine to obtain better insights into the network traffic.
- We developed a controlled load allocation scheme supporting parallel execution of those algorithms on multiple CPU cores without slowing down the processing.
- For the first time, we integrate anomaly detection with high speed flow monitoring to provide additional information for efficient post processing.

II. Fundamentals and Related Work

Anomaly detection refers to the process of identifying properties within a certain data set that deviate from normal behavior. Usually this is done by applying an application-specific ADA on a data set that should be examined for anomalies. Anomalies may be classified by type, severity or any other relevant property and therefore often are assigned attributes like floating-point scores or numeric class values. The computational complexity of ADAs has to be considered because this might be a limiting factor. On the other hand, signature-based IDS achieve high

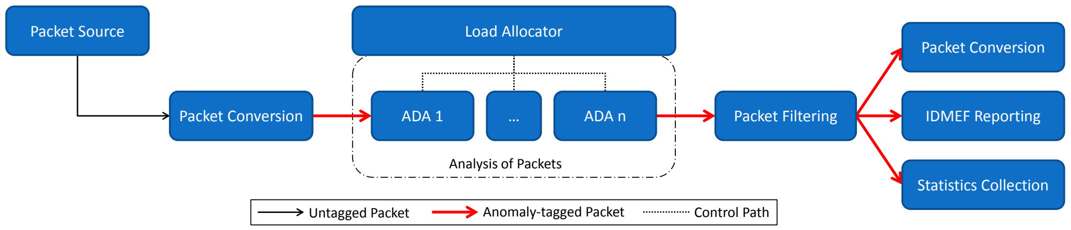


Figure 1. Architecture overview. A controller feeds packets to as many Anomaly Detection Algorithms (ADAs) as system load tolerates, each of which can tag packets with an anomaly score for downstream processing.

detection rates for intrusion patterns of known exploits, but they provide only poor performance in detecting novel attacks [6].

Examples of anomaly detection systems include *SPADE* [7] and *NIDES* [10]. Depending on the used algorithms, each system might have its own weaknesses which prevent certain anomalies from being detected. Algorithms that only rely on IP addresses and port numbers for anomaly detection in particular, are not sufficient to provide reliable attack detection.

Because both signature-based and anomaly-based techniques have their own drawbacks, there are good reasons to combine both approaches into a hybrid IDS. An experimental system has been proposed in [8]. As a key concept, a *weighted signature generation* scheme has been developed in order to classify anomalies by assigning *anomaly scores* and *normality scores* to each network connection.

Approaches such as Time Machine [15] or Front Payload Aggregation (FPA) [16] suggested the use of flow monitoring techniques with added payload information to speed up the network monitoring performance. As an example, the first N bytes of packet payloads can be embedded into the flow data, which allows significantly reducing the amount of data that has to be processed by an IDS but still gives access to parts of the payload information. It turned out that attacks are frequently not detectable using this concept as the attack pattern only shows up after some initial higher layer protocol exchange. Dialog based Payload Aggregation (DPA) goes well beyond Time Machine and FPA as it focuses on the relevant parts of the entire session instead of the first N bytes [14].

We make use of these ideas, both anomaly detection and DPA, to design an attack detection framework that is able to cope with high-speed networks well beyond data rates of $1\,\mathrm{Gbit\,s^{-1}}$ using commodity hardware. Our system is implemented in the Versatile Monitoring Toolkit (Vermont) [12].

III. Architecture

Figure 1 outlines the architecture of the developed framework, including the underlying data flow. Any packet source, which can be either live traffic or a trace, may be used to provide the input data for the anomaly detection framework, and hence the implemented algorithms.

A. Packet Analysis

In order to classify anomalies, packets have to be tagged with information provided by the different ADAs. Usually, these algorithms generate score-based values that are assigned

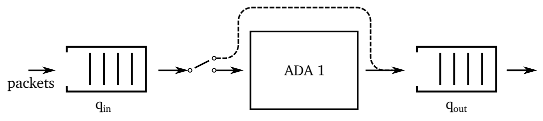


Figure 2. The controlled load allocation scheme can skip individual ADAs if load gets too high.

to the individual packets. We therefore tag network packets with an anomaly score and add a list for all these values. Thus, we allow each ADA to tag the packet structure internally. Such anomaly tags also contain a unique algorithm instance key. This is required to distinguish between anomaly scores from different algorithms in the post-processing step and also allows to use multiple instances of the same algorithm, even with different configurations.

It is important that our framework is able to support algorithms of different types, including semi-supervised algorithms requiring an initial training time interval and unsupervised algorithms which do not require any user interaction. As a proof of concept, we implemented two basic packet-based anomaly detection algorithms: Packet Header Anomaly Detection (PHAD) [4], which examines header field values of different protocols at link, network, and transport layer; and Network Traffic Anomaly Detector (NETAD) [5], which also incorporates packet payloads for anomaly detection. The choice fell on these two to have one ADA inspecting only protocol headers and one focusing also on the payload. All implemented ADAs are implicitly part of the framework's controlled load allocation scheme.

B. Controlled Load Allocation Scheme

The key challenge in using multiple packet-based anomaly detection algorithms is that they strongly differ in their computational complexities and, therefore, the processing rates can be very different. For optimal performance, faster algorithms should be able to analyze a larger amount of packets compared to slower algorithms. The main objective of our controlled load allocation scheme is thus to provide anomaly detection on an "as-much-as-possible" basis. If enough CPU cycles are available, the allocator should not interfere so that every single packet can be analyzed by all algorithms.

The solution that provides us with the most flexibility at low complexity is the processing scheme as depicted in Figure 2, we used *sequential processing with packet skipping*. Figure 2 shows the principles of skipping an ADA stage (simply discarding such packets would not be desirable, as there might be another (faster) algorithm in the pipeline that has enough resources to process the packet). Every ADA is implemented within a single (lightweight) module that can run on a separate CPU core, which nicely fits into Vermont's overall architecture. The sequential processing is traditionally organized in form of a pipeline. We therefore added a controller (cf. Figure 1) that is able to identify congestion and to throttle the amount of packets that are processed by the slowest algorithm instances. This is done by skipping packets and immediately pass them to the next pipeline stage without adding an anomaly score entry. Instead of using a window-based approach for skipping packets (which may have even a negative impact if all ADAs have similar computational complexities), we investigated a probabilistic approach. This allows fine-grained adjustments of skip rates.

The main pipeline controller performs two different tasks: *identification of congestion* that are caused by one or more pipeline stages and *throughput control* by adjusting the skip probabilities in the individual pipeline stages. For identifying congestions in the packet processing, we have defined a *congestion criterion*:

$$d(q_{in}) > t_{in} \quad \wedge \quad d(q_{out}) < t_{out} \tag{1}$$

The congestion criterion relies on the fill level of the input $d(q_{in})$ and the output queue $d(q_{out})$ together with threshold values for each queue. If the input queue exceeds a defined threshold t_{in}, while the filling degree of the output queue is below a certain threshold t_{out}, that the pipeline is congested.

Throughput control is performed by using a *feedback control algorithm* that is invoked in regular time intervals. The algorithm initially uses a learning phase during which throughput quotas are determined for each individual pipeline stage based on its throughput performance. From these quotas, skip probabilities are derived for every pipeline stage in order to balance the amount of input data that is going to be processed by the ADAs. The skip probabilities calculated by the feedback control algorithm follow a Multiplicative Increase / Multiplicative Decrease (MIMD) approach based on quotas.

C. Post-Processing of Packets

After packets have been analyzed by algorithms, we have different options for post-processing the anomaly-tagged packets. The framework's filtering engine can be used to select packets based on the anomaly scores. If an algorithm generates score-based results, only packets with scores that exceed a certain threshold are of interest. However, an ADA may also be used to categorize packets (this can basically be interpreted as assigning an integer $c \in S$ from a finite set of classes to each single packet). In this case, the filtering engine can be configured to select packets matching a certain class.

For analyzing the performance and the behavior of the implemented ADAs, we implemented a statistics collector that provides information about the distribution of the generated anomaly scores for certain packets.

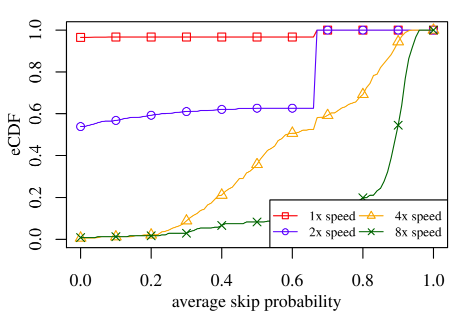
IV. Evaluation

We thoroughly evaluated the performance of our proposed system that combines multiple Anomaly Detection Algorithms with our load allocation scheme. We stepwise evaluated all core components that have been developed as part of the anomaly detection framework. Our aim is to show that there is a substantial benefit of using multiple ADAs, even when only using three different instances of two distinct algorithms. Furthermore, we demonstrate the capability of the controlled load allocation scheme to ensure high detection rates in high-speed networks without dropping packets due to the computational complexities of the used ADAs.

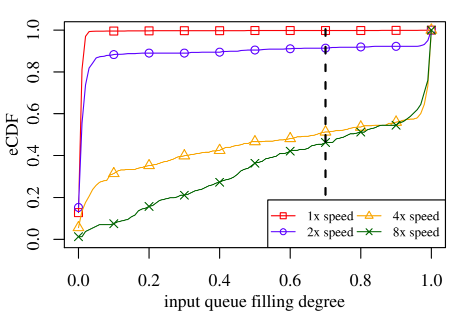
For all the experiments, we used network traces as input data to be able to compare results of different test runs. The *anomaly trace* has been generated by recording traffic from the network link of a "typical" home computer together with background traffic generated by the operating system and other installed applications. We artificially injected anomalies of 12 different types by either generating entirely new packets or by modifying recorded packets. The *attack trace* is based on the 1999 DARPA Intrusion Detection Data Sets [17]. For our evaluation, we have created a network trace by concatenating traffic from this data set, including attack-free traffic for training of the algorithms (if required) and traffic containing 8 different types of attacks. The *load allocation trace* has been captured from the Gigabit Internet uplink of a university. Thanks to the relatively high data rate of $450\,\mathrm{Mbit\,s^{-1}}$ and by replaying the trace with different speed multipliers, we have been able to stress our anomaly detection framework, including the implemented algorithms and the developed controlled load allocation scheme. Finally, the *detection rate trace* is an artificially created trace consisting of three types of packets, all identified by the used IP address: the first type represents normal traffic, the second and third type represent anomaly traffic. The two types of anomaly traffic each contain exactly 500 packets that are equally distributed over the trace in bursts of 10 packets. The duration of the entire trace is 568 s (if replayed at a rate of 75 kpps; with the used small packet size this results in $35\,\mathrm{Mbit\,s^{-1}}$).

A. Anomaly Detection Algorithms

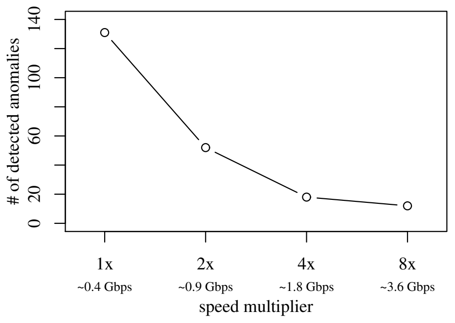
First, we verified that the implemented algorithms work as expected by performing several tests with the anomaly and attack detection trace. Because a detailed analysis of the implemented ADAs has already been provided in the publications of the authors, the aim of our evaluation is to show that anomaly detection (and therefore attack detection) in general can benefit from using multiple algorithms simultaneously. For this evaluation the anomaly detection framework has been configured to use a total number of three ADA instances, namely the semi-supervised PHAD algorithm and the NETAD algorithm in both, semi-supervised and unsupervised (auto) operation mode. Based on the overall distribution of anomaly scores we defined three sensitivity levels so that only packets with anomaly scores in the top 0.01% / 0.1% / 1% pass the filter.



(a) skip probabilities

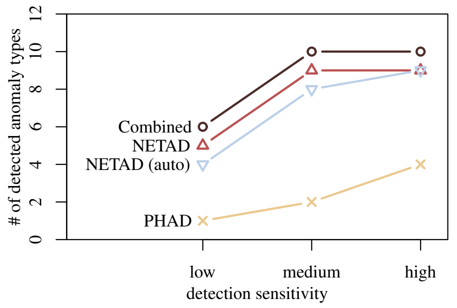


(b) input queue filling degree

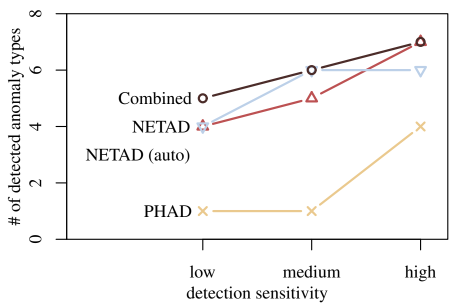


(c) detection quality

Figure 4. Impact of packet rate on skip probability, input queue filling degree, and detection quality (load allocation trace).



(a) anomaly trace



(b) attack trace

Figure 3. Combining ADAs can increase detection quality, particularly at low sensitivities.

Figure 3 shows the number of anomaly types (anomaly trace) and attack types (attack trace) that could be detected by using multiple algorithms instead of one. The maximum number of detectable anomaly/attack types is 12 for the anomaly trace and 8 for the attack trace, but not all anomalous packets passed the filter. There were no false positives in the detected anomalies.

From that we can conclude that using multiple algorithms has a positive impact on the overall detection performance, even when only using a small number of different algorithms. If additional algorithms would be integrated into the framework we expect that detection performances could be increased further. Clearly, more of our anomalies and attacks can be detected if lower anomaly score thresholds are configured, but this may also increase the number of false positives which is not desired in practice. In practice detection sensitivity will always be low. Figures 3a and 3b confirms this is exactly where the combination of ADAs shows the highest benefits.

B. Controlled Load Allocation Scheme

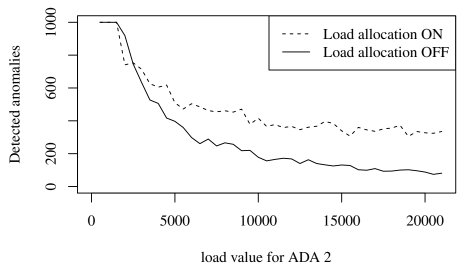
In a first step, we have evaluated the implemented feedback algorithm which is used to continuously adjust the skip probabilities in the individual pipeline stages. Figure 4a gives an overview of the average skip probability ($\bar{p} = (p_1 + p_2 + p_3)/3$) by showing the CDF plots for different speed multipliers. As we can see, the average skip probability is almost zero when using a speed multiplier of 1. This indicates that all algorithms are able to keep up with the traces' average packet rate of $450\,\mathrm{Mbit\,s^{-1}}$. As the result of a data rate burst within the input data, congestion has been observed once by the controller, which has caused a temporary increasing of skip probabilities. Furthermore, we can observe that average skip probabilities get higher if the speed multiplier is increased and therefore the pipeline suffers from congestion. The curve for a speed multiplier of 4 shows an approximately linear distribution of the average skip probability. For speed multipliers of 4 and 8 we can see that the probability for processing packets is almost zero, which means that the pipeline is suffering from congestion all the time.

Figure 4b shows the filling degree of the input queue which has been configured to use a threshold value of $t_{in} = 0.7$ for the congestion criterion. There have been almost no congestions (and thus $d(q_{in}) \leq 0.7$) when replaying the input data with the original speed, because (according to the average skip probability of 0.0) every single packet could be analyzed. Also when doubling the average packet rate, the pipeline does not suffer from congestion with a probability of approx. 0.9.

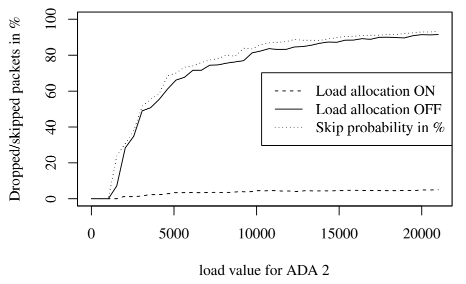
In Figure 4c we can see that the data rate burst which has caused a temporary congestion (and therefore an increase of skip probabilities) when replaying the input data with original speed only had a small impact on the number of detected anomalies. Clearly, the overall detection performances decrease if speed multipliers get larger. If we are doubling the speed, only about 40 % of the original anomalies could be detected. For a speed multiplier of 8, we have been able to detect only 12 out of 133 anomalies. This relationship can also be observed in the increasing values of skip probabilities (cf. Figure 4a).

C. Behavior under Stress

To obtain more insights on the behavior under heavy load, we analyzed the packet detection rate using the described



(a) detected anomalies



(b) dropped packets and skip probability

Figure 5. Performance with increasing ADA load

detection rate trace. For this experiment, we implemented a quite simple ADA, which only checks the IP source address of the incoming packets and assigns a high anomaly score if it matches. This guarantees that the number of detected anomalies is not dependent on the packet arrival times or changes if dropped packets lead to other internal system states. The most important feature is that its CPU load can be manipulated. This mechanism is used to investigate increasing skip probabilities.

The test configuration consists of two of the above simple ADAs, one configured to give a high anomaly score to the first type of anomaly traffic and the second is configured to give a high anomaly score to the second type of anomaly traffic. Thus, 50 % of the anomalies are supposed to be detected by the first ADA and the other 50 % by the second ADA. The test has been repeated multiple times, always increasing the load on the second ADA (load value in the presented results).

Figure 5 shows the results of the experiment. As can be seen, the anomaly detection systems without controlled load allocation drops substantially more packets with increasing load (more than 90 % at high load) with strongly decreasing detection rates (about 8 % at high load). In contrast, the proposed controlled load allocation scheme achieves much higher detection rates. Instead of random packet drop, packets skip only one (the more computationally expensive) ADA while being processed by the other ADA. The resulting detection rate is substantially higher (335 out of 1000 anomalies) while the number of dropped packets is negligible.

V. Conclusion and Future Work

We proposed an anomaly detection framework to optimize attack detection in computer networks that has been implemented as part of an existing network monitoring toolkit. The framework features a controlled load allocation scheme and can be used with different types of ADAs. However, neither the framework nor anomaly detection in general intends to replace existing (signature-based) techniques for attack detection. Instead, anomaly detection should serve as a supplement to traditional techniques for detecting novel attacks that have not yet been described by signatures or rules.

Our framework provides an efficient way for packet-based anomaly detection in computer networks, even in high-speed networks. Although it currently only implements two different ADAs, we could show that detection performances can benefit from using multiple algorithms. As a next step, the integration of additional ADAs should be considered in order to further improve the overall detection rates. The adaption of existing algorithms to protocols not yet supported might also be desired, particularly when protocols like SCTP and DCCP become more popular. To provide a mixture of packet-based and flow-based processing, also anomaly detection for packet payloads stored within flow data (like Internet Protocol Flow Information Export (IPFIX) records) may be considered as a further enhancement.

References

[1] F. Sabahi and A. Movaghar, "Intrusion Detection: A Survey," in *IEEE ICSNC 2008*, Sliema, Malta, October 2008, pp. 23–26.

[2] M. Roesch, "Snort: Lightweight Intrusion Detection for Networks," in *USENIX LISA 1999*, Seattle, WA, November 1999, pp. 229–238.

[3] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly Detection: A Survey," *ACM Computing Surveys*, vol. 41, no. 3, July 2009.

[4] M. V. Mahoney and P. K. Chan, "PHAD: Packet Header Anomaly Detection for Identifying Hostile Network Traffic," Florida Tech., Technical Report CS-2001-4, 2001.

[5] M. V. Mahoney, "Network Traffic Anomaly Detection Based on Packet Bytes," in *ACM SAC 2003*, Melbourne, FL, March 2003, pp. 346–350.

[6] K. Wang and S. J. Stolfo, "Anomalous payload-based network intrusion detection," in *RAID 2004*, Sophia Antipolis, France, September 2004, pp. 203–222.

[7] S. Biles, "Detecting the Unknown with Snort and the Statistical Packet Anomaly Detection Engine (SPADE)," 2006. [Online]. Available: http://www.computersecurityonline.com/spade/SPADE.pdf

[8] K. Hwang, M. Cai, Y. Chen, and M. Qin, "Hybrid Intrusion Detection with Weighted Signature Generation over Anomalous Internet Episodes," *IEEE Transactions on Dependable and Secure Computing*, vol. 4, no. 1, pp. 41–55, January 2007.

[9] M. Cova, D. Balzarotti, V. Felmetsger, and G. Vigna, "Swaddler: An approach for the anomaly-based detection of state violations in web applications," in *RAID 2007*, Queensland, Australia, September 2007.

[10] D. Anderson, T. F. Lunt, H. Javitz, A. Tamaru, and A. Valdes, "Detecting Unusual Program Behavior Using the Statistical Components of NIDES," SRI International, Tech. Rep. SRI-CSL-95-06, May 1995.

[11] W. Lu, M. Tavallaee, and A. A. Ghorbani, "Detecting Network Anomalies Using Different Wavelet Basis Functions," in *CNSR 2008*, Halifax, Canada, May 2008, pp. 149–156.

[12] R. T. Lampert, C. Sommer, G. Münz, and F. Dressler, "Vermont - A Versatile Monitoring Toolkit for IPFIX and PSAMP," in *IEEE/IST MonAM 2006*, Tübingen, Germany, September 2006, pp. 62–65.

[13] T. Limmer and F. Dressler, "Adaptive Load Balancing for Parallel IDS on Multi-Core Systems using Prioritized Flows," in *IEEE ICCCN 2011*, Maui, HI, July/August 2011, pp. 1–8.

[14] ——, "Improving the Performance of Intrusion Detection using Dialog-based Payload Aggregation," in *IEEE INFOCOM 2011, GI Symposium*, Shanghai, China, April 2011, pp. 833–838.

[15] S. Kornexl, V. Paxson, H. Dreger, R. Sommer, and A. Feldmann, "Building a Time Machine for Efficient Recording and Retrieval of High-Volume Network Traffic," in *ACM IMC 2005*, Berkeley, CA, October 2005, pp. 267–272.

[16] T. Limmer and F. Dressler, "Flow-based Front Payload Aggregation," in *IEEE LCN 2009: WNM Workshop*, Zurich, Switzerland, October 2009, pp. 1102–1109.

[17] "DARPA Intrusion Detection Data Sets." [Online]. Available: http://www.ll.mit.edu/mission/communications/ist/corpora/ideval/data/index.html