SELF-ORGANIZING NETWORK SECURITY ENVIRONMENTS – A SIMULATIVE ANALYSIS

Falko Dressler, Lavanya Poondru, Tobias Limmer and Reinhard German

Computer Networks and Communication Systems
Dept. of Computer Science, University of Erlangen, Germany
{dressler,limmer,german}@informatik.uni-erlangen.de

Abstract: **We discuss the need for adaptive load control in network security environments in order to cope with the increasing bandwidth requirements. In earlier work, we developed a simple model to study the behavior of feedback loops for self-configuring security environments. We primarily considered the traffic between the monitoring probes, the IDS systems, and associated firewalls. We now enhanced the model to study fully self-organized network security environments in a simulation model. First simulation results outline both the feasibility of the general approach and the possibilities of the simulation model.**

Index Terms**: Network monitoring, self-organization, adaptive feedback loops, complex security environments.**

I. INTRODUCTION

Performance issues in distributed network security environments are being in the main focus of a number of research projects. Besides developing even faster computing systems, main focus is on algorithmic approaches. In the context of this paper, we consider attack detection and mitigation architectures. Basically, such systems rely on three components: Packet data is to be monitored, e.g. using a network tap, the collected data is further analyzed on an IDS system, and finally, reactive measures against detected attack traffic have to be initiated, e.g. by configuring firewall systems appropriately. With the increasing complexity of today's networks, the need aroused to deploy the named components in a distributed manner [9, 13]. Fig. 1 shows a typical distributed security environment.

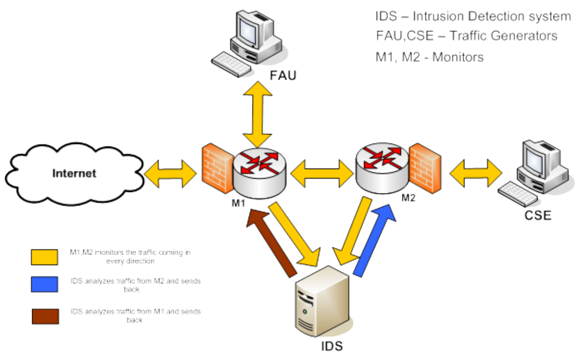
IDS – Intrusion Detection system
FAU,CSE – Traffic Generators
M1, M2 - Monitors
FAU
Internet
M1
M2
CSE
IDS
M1,M2 monitors the traffic coming in every direction
IDS analyzes traffic from M2 and sends back
IDS analyzes traffic from M1 and sends back

Fig. 1 – Distributed network security scenario consisting of routers with monitoring functionality, a central IDS, and distributed firewalls

In particular, multiple monitoring probes are employed to obtain information about the ongoing network traffic. This information is forwarded to the associated attack detection systems, which in turn analyze the data and close the loop by configuring firewall systems to counteract the identified attacks. Similar scenarios can be thought of for accounting or traffic engineering purposes.

Driven by the needs of network security installations but also by the demands of accounting and charging systems, network monitoring methods and techniques have been standardized by several organizations, first of all by the Internet Engineering Task Force (IETF). This includes the communication between the monitoring probes to the traffic analyzers, among the attack detection systems, and for the configuration of monitors and firewalls. In order to cope with the steadily increasing amount of data and the very high bandwidths in nowadays backbone networks, the reduction of monitoring data is a key issue for successful monitoring solutions [1, 11]. Thus, we focus in performance aspects of the monitoring and analyzing part.

The objective is to develop adaptive mechanisms for reducing the amount of monitoring data, or, to be more precise, to adjust the load in the entire monitoring system [8]. Based on earlier work on a simplified model to analyze these performance issues [7, 10], we developed a more complex model that allows to perform evaluations of distributed security environments. The primary goal is to monitor as much as possible in order to achieve more accurate results. A threshold is given by the processing capacity of all involved systems; therefore, an upper bound is defined. Additionally, this threshold depends on the kind of the data to be processed.

Using the structural model presented in [7, 10], we analyze its applicability in distributed environments. In particular, we created a new simulation model for the network simulator OMNeT++ [14], which allows a detailed performance evaluation. Using the simulation model, we performed a number of experiments that help to characterize the basic behavior of the use of feedback controlled adaptive monitoring techniques in distributed environments.

The rest of the paper is organized as follows. Section II introduces the need for adaptive network monitoring in the context of network security environments, especially for efficient attack detection. Then, Section III outlines the basic approach for adaptive parameter optimization using two separate feedback loops. The analyzed scenarios are presented in Section IV followed by a discussion of selected simulation results in Section V. Finally, Section VI concludes the paper.

II. NETWORK MONITORING

Major focus of the network monitoring community is on flow monitoring that allows to reduce the amount of monitoring data to a huge extent. Flows are data records that describe network traffic. In most applications, they contain aggregated data that summarizes connections or packets that were transferred over the monitored network. A typical configuration would be using the IP 5-tuple `<source IP, dest IP, source port, dest port, protocol>` as flow keys, i.e. attributes describing the flow. Furthermore, relevant statistical data can be added such as the flow start and end times or the number of bytes of all packets belonging to the flow. If flow records are configured to represent single packets, parts of payload data may also be contained.

A wide number of application scenarios exist that relies on flow monitoring. Besides simple anomaly detection methods like top-N lists, more intelligent traffic summaries [12] or application identification methods [5] are being investigated.

In order to reduce the amount of packet information to be analyzed at the attack detection level, filtering and sampling techniques can be employed. It can be shown that such methods – even though reducing the granularity of the monitored data – can greatly help to detect attacks and misbehavior in high-speed networks using distributed monitoring [6].

For the transmission of monitoring data, the IETF developed a protocol and information model named IP Flow Information Exchange (IPFIX) [3, 4]. The IPFIX protocol is built on a template based system for information exchange, making it very flexible with regards to changing the default information fields of the exported flow records. Because of the template based solution of flow records, it is fairly easy to tailor an IPFIX based flow information export system to network operators specific needs. The transmission of collected flow information, i.e. the data export in IPFIX terminology, occurs in regular intervals controlled by an active and an inactive timeout, i.e. a measure for the maximum time of storing a flow at the monitor and the maximum gap between two consecutive packets. Therefore, the protocol is well-suited for transmission of huge amounts of monitoring data between monitoring probes and analyzing stations as well as for dynamic flow data mediation.

III. FEEDBACK-LOOP BASED ADAPTATION

In this section, we briefly review the feedback based approach presented in [7, 10]. This method is based on a bio-inspired technique that creates appropriate feedback loops for adapting the parameters in the monitoring environment depending on the current load in the network. Usually, two different feedback loops need to be used in combination: *positive feedback* for short-term amplification and *negative feedback* for long-term regulation. The intrusion detection reports detected attacks to the firewall that in turn is blocking this traffic and reduces the number of packets to be monitored. Additionally, the IDS reports legitimate traffic to the monitor. This monitor stops reporting the packets belonging to these flows and, therefore, reduces the number of packets to be analyzed. Obviously, both configurations cannot be permanent. Sources sending legitimate traffic might begin to send attack packets at any time. Also, malicious systems may be patched or otherwise "corrected" and should not be starved by our firewalls.

The adaptation scheme is based on two building blocks. First, we define blacklists and whitelists representing flow patterns of detected attack traffic and legitimate traffic, respectively. In particular, the blacklists represent the firewall systems that completely filter (detected) attack traffic. On the other hand, whitelists are used to reduce the amount of monitoring data. Usually, it is not necessary to report traffic information about clearly identified legitimate connections. Secondly, we use feedback loops for system control. The adaptation is done by calculating appropriate timeouts. TO_{black} corresponds to the firewall system and TO_{white} to the monitoring probe, i.e. each entry in the blacklist is associated a timeout when to expire the particular entry. Similarly, each whitelist entry is managed.

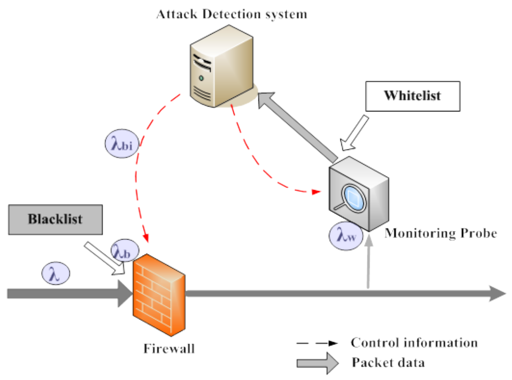

Fig. 2 – Basic model for the adaptation of the system load using a feedback-based approach [7]

Fig. 2 shows the basic model. As depicted, different traffic rates (denoted as λ_k) are continuously measured. Table 1 lists all these input parameters that are used to calculate the timeout values.

Table 1: Input parameters, i.e. data rates

λ	Arrival rate
λ_b	Arrival rate of "black" packets
λ_{bi}	Arrival rate of "black" packets belonging to the i-th flow
λ_w	Arrival rate of "white" packets
λ_{IDS}	Maximum capacity of the IDS

The timeout values TO_{black} and TO_{white} are continuously adapted according to the current system state, which is reported among the participating systems, i.e. the monitoring probes, the attack detection systems, and the firewalls. In particular, these values are calculated according to Equations (1) and (2) as shown below.

$$TO_{black} = C_1 \underbrace{\frac{\lambda_{bi}}{\lambda}}_{t_1} + C_2 \left(\underbrace{\frac{\lambda_b}{\lambda}}_{t_2} + \underbrace{\frac{\lambda}{\lambda_w}}_{t_3} + \underbrace{\frac{\lambda}{\lambda_{IDS}}}_{t_4} \right) \qquad (1)$$

$$TO_{white} = C_3 \left(\underbrace{\frac{\lambda}{\lambda_{IDS}}}_{t_5} + \underbrace{\frac{\lambda}{\lambda_b}}_{t_6} \right) \qquad (2)$$

The single terms are discussed in the following. In principle, all terms belonging to one timeout are summed up and scaled by a constant.

- t_1 Ratio of the ith attack flow to the overall attack rate. Used for penalizing previously discovered attack flows. This term must be scaled separately using C_1 because it is usually very small.
- t_2 Similar to t_1 but defined the ratio of arriving attack traffic to the overall throughout. The larger it is, the more aggressive the attack.
- t_3 This term describes the safety of the arriving traffic: the larger the amount of "white" packets, the smaller the requirement for large timeouts at the firewall.
- t_4 This term is a measure for the overload of the attack detection system.
- t_5 The same as t_4 but used at the monitor.
- t_6 Similar to t_3 but defining the risk of arriving packets.

Each term is represented by a fraction of two rates. Therefore, each has to be read as "$t_i = 0$ if $\lambda_k = 0$". In our first experiments, we evaluated appropriate values for these constants. In a next step, the constants themselves can be adapted to the current scenario. Further details about the model and the calculations can be found in [7].

IV. SIMULATION MODEL AND SCENARIOS

For the simulative analysis of the performance in the distributed case, we implemented a simulation model in OMNeT++ together with its INET Framework extension. OMNeT++ is an event-based network simulator. Scenarios in OMNeT++ are represented by a hierarchy of reusable modules written in C++. Modules' relationships and their communication links are stored as Network Description (NED) files and can be modeled graphically. Simulations are either run interactively, in a graphical environment, or are executed as command-line applications. The INET Framework provides a set of OMNeT++ modules that represent various layers of the Internet protocol suite, e.g. the TCP, UDP, IPv4, and ARP protocols.

We implemented IPFIX based monitoring probes as well as firewall systems capable to maintain the described whitelist and blacklist functionality. The IDS is modeled as a stochastic process that identifies flows as malicious, legitimate, or unknown according to a given fixed probability or a probability distribution. This is an abstraction from real world behavior but we tried to model the ratios as correct as possible – and we also evaluated differences in the system behavior for varying probability distributions of attack detection.

The basic scenario that we analyzed is depicted in Fig. 1 and a screenshot of the main network in the simulation is shown in Fig. 3. Three networks are connected by two routers. In particular, we observed traffic from/to the Internet that comes from the computer science department (CSE, right). In addition, the "normal" university Internet traffic is analyzed (FAU, top).

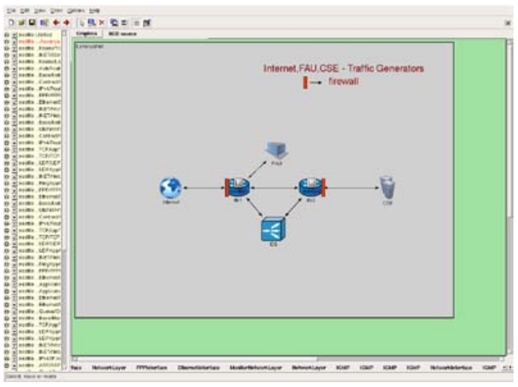

Fig. 3 – Basic scenario used for the simulations: three networks are connected by two routers, which also provide monitoring and firewall functionalities

In order to obtain as accurate measures as possible, we used trace-driven input modeling techniques. All the experiments were performed on a set of trace data that we collected in advance and stored it in an anonymized form. The trace file used for simulations have the following formatting:

```
10:19:31.048660 10.10.37.59.1398
                10.10.37.230.22 tcp 120
10:19:31.048924 10.10.37.230.22
                10.10.37.59.1398 tcp 88
10:19:31.083884 10.10.37.1.520
                10.10.37.255.520 udp 52
```

The first value corresponds to a time stamp, which is also used to advance the time in the event-driven simulation environment appropriately. Then, the source IP address/port and destination IP address/port of the packet are denoted. Finally, the protocol and the packet size are appended. A separate traffic generator module has been implemented that reads the trace files, generates new packets within

OMNeT++, and advances the simulation time appropriately.

In order to analyze the security system's performance, a number of measures have been recorded. Main focus was on the different arrival rates, i.e. all the λ_k. Furthermore, we observed the number of entries in the blacklists and whitelists in order to obtain information about the resource consumption on firewalls and monitors, respectively. The adaptive feedback algorithm can be best analyzed by evaluating the timeouts for each entry in both the lists.

In total, we analyzed four scenarios. First, we re-created the simple setup used in [7] for cross-validation of the simulation model. The simulation results outlined the validity of the implementation. Furthermore, and the main contribution of this paper, we analyzed three distributed setups. We started with the straightforward approach as depicted in Fig. 1. The central IDS analyzes the traffic reported from monitor M1 and M2 separately and creates corresponding reports for the respective firewalls. In the next scenario, we mixed the monitoring traffic before it is analyzed. This is a reasonable assumption because hierarchical monitoring environments can be expected in large-scale networks [2]. In turn, the whitelists and blacklists installed on both monitors and firewalls contain the same information. Lastly, we added more intelligence to the IDS in terms of knowledge about the network topology. This can be anticipated in real networks as well exploiting network management data. With this scenario, we tried to reduce the resource consumption of blacklists on the respective firewalls. It turned out that this approach does not work as expected as we could not provide bidirectional flow information. Thus, we concentrate on the first two scenarios in the next section in which we present and discuss the simulation results.

V. RESULTS AND DISCUSSION

For the simulative analysis of performance aspects in the monitoring probes and the firewalls, which in turn provide insights into the general behavior of the adaptive load management system, we performed a number of experiments. Each experiment was based on the same trace of network packets to provide comparability. The trace includes 20min of packet data. Simulation control was applied by performing at least five runs for each experiment to analyze stochastic effects of the simulation.

Most of the results are shown as boxplots. For each data set, a box is drawn from the first quartile to the third quartile, and the median is marked with a thick line. Additional whiskers extend from the edges of the box towards the minimum and maximum of the data set. Data points outside the range of box and whiskers are considered outliers and drawn separately.

We analyzed the impact of a number of parameters on the behavior of the whitelist and blacklist management. In particular, we evaluated the influence of the export interval that is configured at the flow processing entities, i.e. the monitoring probes. We analyzed equal export interval timing for both the monitors, ranging from 5s to 40s as well as different export interval timings, e.g. a 5s export interval for M1 and 10s for M2.

Furthermore, we studied the impact of the detection ratio of the IDS. In real systems, this ratio will largely vary depending on the current attack situation. Therefore, we analyzed static detection ratios as well as a stochastic behavior, i.e. a uniform distribution for the detection ratios.

The most important simulation parameters are listed in Table 2. This includes also the settings for the static parameters in the adaptation algorithm, i.e. C_1, C_2, and C_3 as well as the processing capabilities of the attack detection system, i.e. λ_{IDS}.

Table 2: Simulation parameters

Export interval	5s – 40s; uniform or different for both monitors
Detection ratio	Static: 0.01/0.1 – 0.2-0.2 (black/white) Stochastic: uniformly distributed
C_1, C_2, C_3	9×10^8, 236, 120
λ_{IDS}	60 packets per second

In a first step, we evaluated the arrival rates at the systems with respect to the analyzed scenario. Fig. 4 depicts the different arrival rates at the monitoring probe M1: λ, λ_w, and λ_b. As can be seen, the variance of the data rates related to the currently installed whitelists and blacklists is rather small. Also, there is only a marginal difference between the two scenarios.

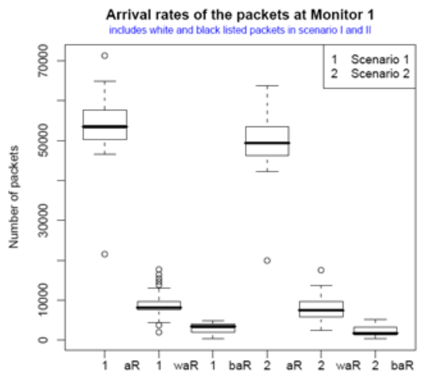

Fig. 4 – Arrival rates at M1: λ (aR), λ_w (waR), and λ_b (baR)

In comparison, the rates at the monitoring probe M2 are depicted in Fig. 5. In this case, there is a recognizable difference between scenario 1 and scenario 2. Obviously, it makes sense to install identified information about attack and legitimate traffic at both monitors and firewalls. This allows to block attack traffic as early as possible as it enters the local network.

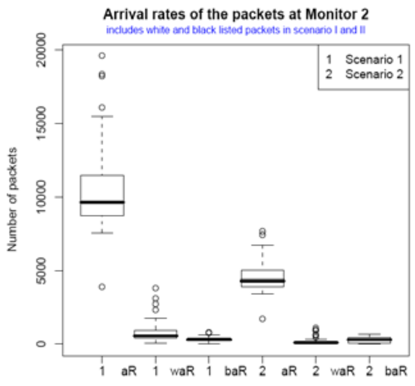

Fig. 5 – Arrival rates at M2: λ (aR), $λ_w$ (waR), and $λ_b$ (baR)

We further analyzed the influence of the export interval. For this, we stepwise increased the export interval from 5s to 40s. It turned out that the export interval has almost no influence on the traffic rates. Nevertheless, it is clear that short lasting attacks cannot be successfully counteracted if using long export intervals because the overall response time increases accordingly.

Our focus was therefore to investigate the resource demands on monitors and firewall. In general, the size of whitelists and blacklists not only contributes to increased memory consumption. The major concern is the length of the lists in terms of computational search time. In case of high variance of the length of the lists, the search operations will take unpredictable time and, therefore, lead to an unstable operation of the overall security system.

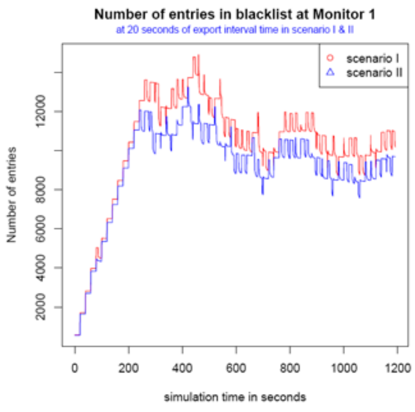

Fig. 6 – Number of blacklist entries at M1

In Fig. 6, the number of blacklist entries at monitor/firewall M1 is depicted. As can be seen, the maximum is at about 12.000 entries, which is a reasonable size even for linear search operations. Also, after a short startup phase, the size of the blacklist converges to a stable state. Another observed effect is the independency of the scenario, i.e. whether all monitored traffic equally contributes to the attack detection analysis and, in turn, to the installed whitelists and blacklists; or if each monitor-firewall combination is handled separately.

Similar effects can be observed for the behavior of the whitelists as installed at monitor/firewall M1 is depicted in Fig. 7. Again, the maximum number of whitelist entries (about 70.000) is of reasonable size for search operations and the size of the list converges quickly to a stable size.

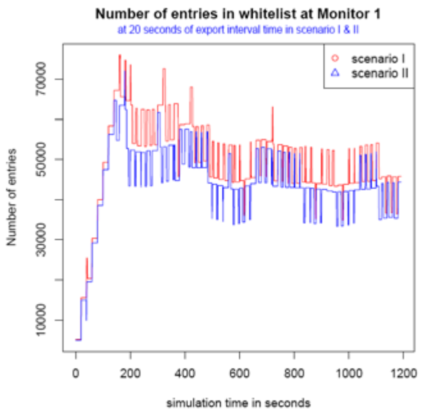

Fig. 7 – Number of whitelist entries at M1

We also analyzed the behavior at the monitor/firewall combination M2. The numbers of blacklist and whitelist entries is depicted are shown in Fig. 8 and Fig. 9, respectively. Similarly to the behavior of M1, the size converges after a short startup phase. Due to the much smaller packet rate, the maximum size of the lists is really small (blacklist: 400; whitelist: 3.000). Also, the deviation is much smaller compared to M1.

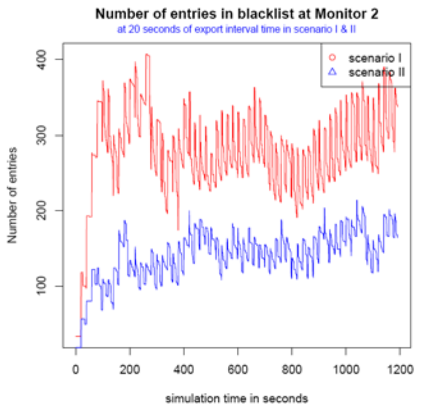

Fig. 8 – Number of blacklist entries at M2

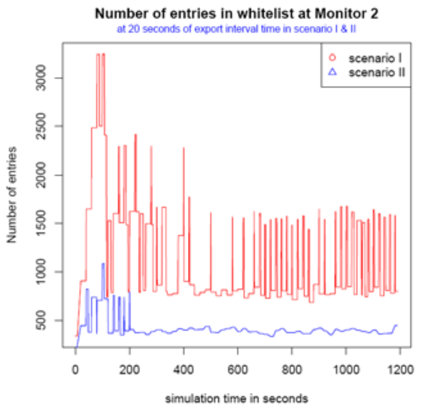

Fig. 9 – Number of whitelist entries at M2

From all these measurements, it can be seen that the feedback loops lead in all analyzed cases to a quick adaptation of the load of all involved systems and, in turn, successfully prevents overload while achieving optimal detection rates. Of course, we conducted further simulations, e.g. for different export intervals and different detection rates at the IDS, that confirmed the presented data.

VI. CONCLUSION

We analyzed the need for adaptive load control in distributed network security environments. In conclusion, it can be said that, in order to cope with the increasing bandwidth requirements, the system load of deployed monitoring probes, attack detection systems, and firewalls need to be and can be controlled in a fully self-organizing manner. We figured out that a feedback-loop based approach that we developed and analyzed in a simple model also performs well in a distributed environment.

The capability of adapting the systems' parameters during runtime makes this approach useful for most monitoring scenarios. Using an amplifying positive feedback loop and protecting a negative feedback loop, self-organizing behavior of the overall system is achieved.

The capabilities of the implemented OMNeT++ model were only explored to a small degree. The model can further be used and extended for many simulative analyses of the performance of methods for efficient monitoring and attack countermeasures. Explicitly to mention are the implemented blacklists and whitelists as well as the IPFIX functionality.

A future extension of the adaptive feedback system is the implementation of bidirectional flows. This can be used to optimize the distribution of whilelist and blacklist entries to appropriate systems in the network, e.g. exploiting topology information as available from a network management system. Another possibility is to have a sensor at the detection system, which maintains the state of the detection system.

Future work will also include experimental validations in a testbed setup.

REFERENCES

[1] Georg Carle, Falko Dressler, Richard A. Kemmerer, Hartmut König, Christopher Kruegel, and Pavel Laskov, "Manifesto - Perspectives Workshop: Network Attack Detection and Defense," Proceedings of Dagstuhl Perspectives Workshop 08102 - Network Attack Detection and Defense 2008, Schloss Dagstuhl, Wadern, Germany, March 2008.

[2] Atsushi Kobayashi, Haruhiko Nishida, Christoph Sommer, Falko Dressler, Emile Stephan, and Benoit Claise, "IPFIX Mediation: Problem Statement," IETF, Internet-Draft (work in progress) draft-ietf-ipfix-mediators-problem-statement-00.txt, May 2008. (http://tools.ietf.org/html/draft-ietf-ipfix-mediators-problem-statement-00)

[3] Jürgen Quittek, Stewart Bryant, Benoit Claise, Paul Aitken, and Jeff Meyer, "Information Model for IP Flow Information Export," IETF, RFC 5102, January 2008. (http://www.rfc-editor.org/rfc/rfc5102.txt)

[4] Benoit Claise, "Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of IP Traffic Flow Information," IETF, RFC 5101, January 2008. (http://www.rfc-editor.org/rfc/rfc5101.txt)

[5] Manuel Crotti, Maurizio Dusi, Francesco Gringoli, and Luca Salgarelli, "Traffic Classification Through Simple Statistical Fingerprinting," *ACM Computer Communication Review (CCR)*, vol. 37 (1), pp. 5-16, January 2007.

[6] José M. González and Vern Paxson, "Enhancing Network Intrusion Detection with Integrated Sampling and Filtering," Proceedings of 9th International Symposium on Recent Advances in Intrusion Detection (RAID 2006), vol. 4219, 2006, pp. 272-289.

[7] Falko Dressler, "Bio-inspired Promoters and Inhibitors for Self-Organized Network Security Facilities," Proceedings of 1st IEEE/ACM International Conference on Bio-Inspired Models of Network, Information and Computing Systems (IEEE/ACM BIONETICS 2006), Cavalese, Italy, December 2006.

[8] Y. Hu, Dah-Ming Chiu, and John C.S. Lui, "Adaptive Flow Aggregation - A New Solution for Robust Flow Monitoring under Security Attacks," Proceedings of 12th IEEE/IFIP Network Operations & Management Symposium (IEEE NOMS 2006), Vancouver, Canada, April 2006, pp. 424-435.

[9] Moheeb Abu Rajab, Fabian Monrose, and Andreas Terzis, "On the Effectiveness of Distributed Worm Monitoring," Proceedings of 14th USENIX Security Symposium, Baltimore, MD, USA, July 2005.

[10] Falko Dressler and Isabel Dietrich, "Simulative Analysis of Adaptive Network Monitoring Methodologies for Attack Detection," Proceedings of IEEE EUROCON 2005 - The International Conference on 'Computer as a Tool', Belgrade, Serbia and Montenegro, November 2005, pp. 624-627.

[11] Holger Dreger, Anja Feldmann, Vern Paxson, and Robin Sommer, "Operational experiences with high-volume network intrusion detection," Proceedings of 11th ACM conference on Computer and Communications Security (ACM CCS 2004), Washington, DC, USA, October 2004, pp. 2-11.

[12] C. Estan, S. Savage, and G. Varghese, "Automatically Inferring Patterns of Resource Consumption in Network Traffic," Proceedings of ACM SIGCOMM 2003, Karlsruhe, Germany, August 2003, pp. 137-148.

[13] Frédéric Cuppens and Alexandre Miège, "Alert Correlation in a Cooperative Intrusion Detection Framework," Proceedings of IEEE Symposium on Security and Privacy, Oakland, California, USA, May 2002.

[14] András Varga, "The OMNeT++ Discrete Event Simulation System," Proceedings of European Simulation Multiconference (ESM 2001), Prague, Czech Republic, June 2001.