Adaptive network monitoring for self-organizing network security mechanisms

Falko Dressler
Autonomic Networking Group, Dept. of Computer Sciences
University of Erlangen, Martensstr. 3, 91058 Erlangen, Germany
Phone: +49 9131 85-27914
dressler@informatik.uni-erlangen.de

Abstract

Network security has become a major part of the network infrastructure especially in the area of detection mechanisms for attack and intrusions. Apart from general measures for attack detection and prevention, the performance of the monitoring architecture employed by the attack detection constitutes an important asset for network security. In this paper, we focus of the amount of measurement data obtained on a monitoring probe that has to be transmitted to and analyzed by an attack detection system. We question the possibility of re-configuring the monitoring part in order to adapt to the computational resources of the analyzer as well as to the current network behavior. In order to cope with these goals, we model the overall security environment consisting of monitors, detection systems, and firewalls. In a simulation we show the speedup gained by adapting the parameters of the monitoring solution.

1. Introduction

Today's communication networks are threatened by an increasing number intrusion attempts, worms, and denial of service (DoS) attacks [19, 21]. Apart from general measures for attack prevention, the possibility to detect ongoing attacks in order to take appropriate countermeasures constitutes an important asset for network security. Therefore, the detection of intrusions, violations, and attacks is a significant task in nowadays communication networks. A brief history and overview to intrusion detection is, for example, given by Kemmerer [18] and Bace [1]. Especially, the number of DoS attacks and distributed denial of service attacks (DDoS) is increasing every day. Typically, important servers from government or industrial systems are attacked, yet we already see similar attacks against primary systems at universities. In addition, the effectiveness and malignance of such attacks is increasing [17].

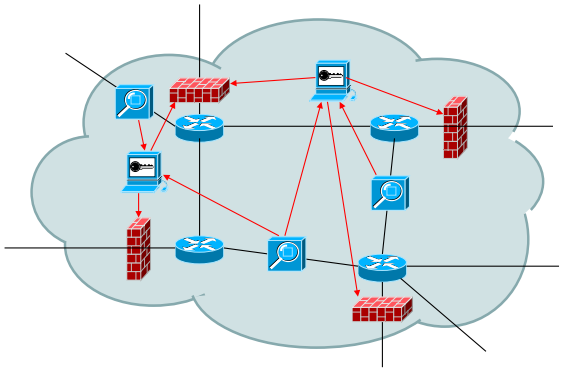

Fig 1. Network security environment. Besides of routers, a network contains monitoring probes, intrusion detection systems, and packet filters (firewalls)

Intrusion or attack detection only one part of a global network security environment. In Fig 1, an overview to a complete network security environment is provided. Intrusion detection systems build the intelligent core of a reactive system that is monitoring, analyzing, and counteracting ongoing attacks and intrusions. Obviously, the functionality depends on the quality of the interaction between involved parts.

In this paper, we focus on this interaction and especially on the performance aspects of the monitoring and analyzing part. In a simulation model we reconstructed a monitoring environment. In order to obtain meaningful results from the analysis, we used a packet trace from our lab server as input for the simulation. We modeled the complete environment by inspecting the traffic flow and configuring adequate firewall rules (blacklist). Additionally, a whitelist was employed that describes the traffic flows detected as legitimate traffic by the intrusion detection system. Finally, we analyzed the behavior of the system by modifying several parameters, such as some filters and sampling parameters, in order to show the possibilities of a self-configuring and self-optimizing network security environment.

The described adaptive solution can be adequately employed in an existing system, CATS (cooperating autonomous detection systems, [11]). Its goal is to identify ongoing attacks using autonomously working detection system that are

able to improve their detection performance through cooperation in a group of multiple detection systems.
The rest of the paper is organized as follows. The primary issues of network monitoring are discussed in section II. The complete network security environment is described in section III. Based on the shown basis mechanisms, the adaptive and self-organizing architecture is presented and discussed in section IV. This section also contains the simulation results obtained in comparative experiments. Finally, the conclusions are provided in section V.

2. Network Monitoring

Network monitoring has become a major research issue. This is because the available bandwidths grow much faster than the processing speed of the monitoring probes. Solutions have been developed that allow to reduce the processing requirements at the analysis. The primary idea behind all these concepts is to split the monitoring and the subsequent analysis into two independent tasks. This became possible because not all packet information are required for network analysis (whether for accounting or security reasons).
The first concept is the netflow accounting. The key idea is to store information about netflows and the corresponding statistics instead of individual packets. In this context, a netflow is defined as a unidirectional connection between two end systems as defined by the IP 5-tupel (protocol, source IP address, destination IP address, source port, destination port). Doing this, a single measurement data set contains information of one up to several thousands of individual packets. For the transmission of the monitoring data to an analyzing system, a special protocol was developed called netflow.vX, where X stands for the version number. Netflow.v9 is the latest version and standardized by the IETF [6]. Its successor is IPFIX (IP flow information export). It was developed by the corresponding IPFIX working group at the IETF and provides sufficient information for a distributed deployment [8, 24]. First implementations are available that support netflow.v9 as well as IPFIX, e.g. nprobe [9] and Vermont [13].
Even if this methodology works well under normal conditions (usual connections consist of about 7.7 packets per flow [20]), there is a major problem during DDoS attacks. Usual attacks forge the IP address of the attack packets, which results in the creation of individual flows per packet. Thus, in attack situations, netflow accounting or IPFIX do not scale well, i.e. they overflow the connection between the monitoring probe and the intrusion detection system (regardless of the computational expense at the analysis). To cope with this problem, recently an aggregation mechanism was introduced [12] that allows to aggregate individual flows into so called metaflows. This aggregation mechanism allows a free scaling of the amount of monitoring data and provides the basic functionality to build adaptive self-optimizing netflow accounting solutions.
The remaining problem is based on the principle of netflow accounting: the lost of payload data. For intrusion detection reasons, this information is often required and, therefore, the applicability of netflow accounting and IPFIX is limited (nevertheless, it offers sufficient information for anomaly detection based on network statistics). To support the selection of single but complete packets and transporting them to an analyzer, PSAMP (packet sampling) was developed in the PSAMP working group at the IETF. It allows the free combination of filters and samplers [16, 25]. Filters are used for deterministically selection packets based on matching field in the IP packet. Samplers are statistical algorithms that select packets using a given sampling algorithm, e.g. count based, and the corresponding parameters. The measured packets are exported to the analyzer using a protocol similar to IPFIX [7, 10].
In conclusion it can be said, that PSAMP allows to monitor and to export complete packets providing sufficient information for subsequent intrusion detection. Additionally, the sampling algorithms, filters, and parameters can be freely defined and re-configured allowing a full-adaptive behavior. The monitoring can be optimized to provide all required data to the analyzer and no more than it is able to process.

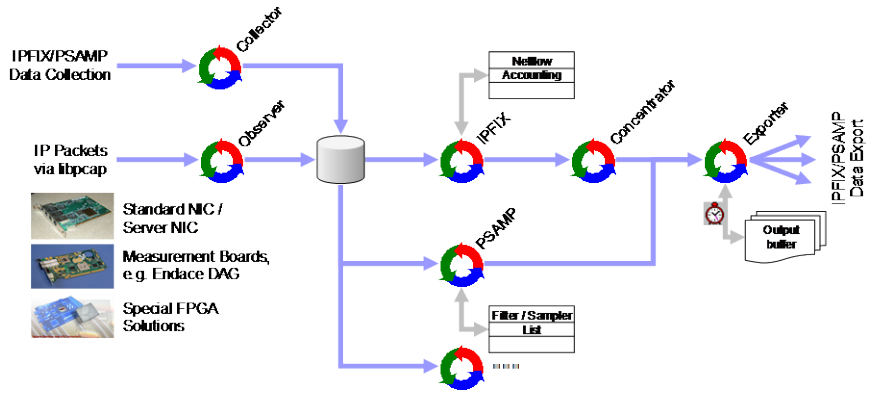

Fig 2. Vermont monitoring architecture [13]. Packets are monitored using standard network interfaces or dedicated hardware. Netflow accounting and packet sampling functionality is provided

In Fig 2, the Vermont architecture is shown. It allows two ways to capture packet data from the network: by using a directly connected NIC, and by employing IPFIX exporters, which send the collected information using the IPFIX protocol. The packet monitoring and sampling layer is responsible for capturing of received packet data. Moreover this layer may preprocess the packet data. Filters or sampling algorithms may be applied to reduce the amount of packets being further processed.

3. Network Security Architecture

The primary goal of this section is to show all required parts of the complete network security architecture as shown in Fig 1. First, the most impressing threats, in our discussion we focus on denial of service attacks, are briefly described followed by some information about intrusion detection systems, countermeasures, and a quick overview to the CATS architecture.

3.1. Denial of Service Attacks

Denial of service attacks focus on the prevention of an offered service. This can be done in two ways: first, by exhausting network resources on the path towards the target server and, secondly, by exhausting resources of the victim server.

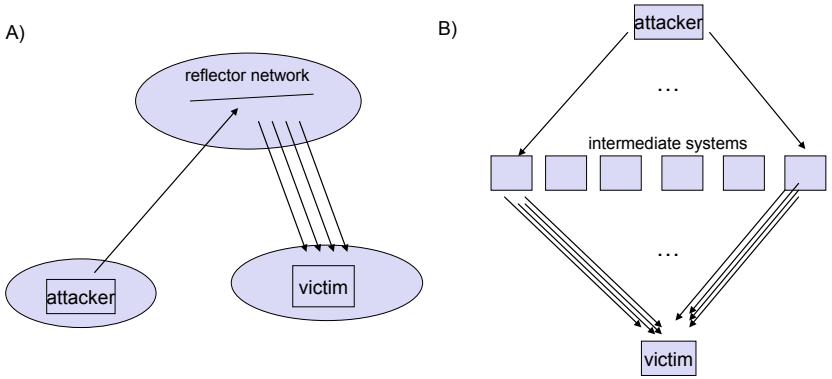

Fig 3. Typical denial of service scenarios. A) ICMP flood using a reflector network; B) TCP SYN flood using intermediate systems, e.g. compromised / malicious systems

An example for the first scenario is a distributed ICMP flood attack. A TCP SYN flood attack is an example for the second scenario. Both scenarios are depicted in figure 4. Further information on distributed denial of service attacks can be found in [4, 21]. Both scenarios are described in the following. Obviously, we concentrate on the behavior of the attack concerning a successful monitoring.

ICMP flood attack

An ICMP flood attack can take place in two ways. First, so called broadcast pings can be employed utilizing an unsecured reflector network for forwarding the ICMP echo request messages towards a victim network as shown in Fig 3 A). Secondly, IP address spoofing can be used by sending ICMP echo requests to multiple stations in the network with the IP address of the victim inserted in the source IP address of each packet. All the receivers of these ICMP request will answer by sending an appropriate response to the victim, from which they think the request was coming from. The result of this attack is an overload of the network paths near the victim. Therefore, normal service requests suffer from the artificial network congestion and cannot be served in an adequate time. The working principle of such ICMP flood attacks leads to packet flows containing a single packet each (unique IP destination address but containing different IP source addresses).

TCP SYN flood attack

The goal of a TCP SYN flood attack as shown in Fig 3 B) is to exhaust local resources at the victim. TCP is a connection oriented transport protocol. Thus, in order to transmit data, a connection has to be established first. This is done by sending a TCP SYN packet which is answered by an SYN+ACK. After the reception of the SYN packet, a half-open connection remains until it is timed out or the SYN+ACK is being answered.
Benefiting form this working principle of TCP, TYP SYN flood attacks employ compromised computers as a relay for a particular attack. All the relay hosts are commanded to send as many TCP SYN packets as possible to the victim. Resources required for state information of half-open connections are exhausted quickly, preventing the victim from receiving legitimate service requests.

3.2. Intrusion Detection

Intrusion or attack detection systems work on monitored packet data. The capability of a detection system to detect anomalies and concrete attacks in a local context is evident. The capability of the detection system to detect anomalies and attacks in a global context is also important. The key properties of detection mechanisms are listed in the following.

Local context – Attack detection in a local context, i.e. based on information from packet data received at the detection system only (or a directly connected monitoring probe) is a straightforward process integrated in almost all detection systems. This capability requires no intercommunication or interaction with other detection systems. Both types of attacks (see Fig 3) can be detected by a system near the victim. Nevertheless, only a system near the attacker is able to detect the source of the attack if IP address spoofing techniques are used. Additionally, traceback mechanisms can be deployed to identify the source of a spoofed IP packet. Unfortunately, such mechanisms have significant resource requirements.

Global context – Using a global context, i.e. information gathered at multiple points in the network, allows improved detection of ongoing attacks in the network. Both scenarios described before can be detected with a global context. Therefore, this capability is an essential requirement. The communication overhead introduced by the different interacting detection systems is an important performance measure of the complete system. Distributed monitoring is also a basis for such a global context. The described monitoring architecture allows such an operation. The focus of this paper is to question if the performance of such a complex system can be tuned in an adaptive way leading to a self-organizing monitoring and attack detection system.

Knowledge-based detection – Knowledge-based detection was the first kind of attack detection deployed in the Internet. While statistical conclusions are not possible, well-known attacks can be detected efficiently using this methodology. Sufficient information about the packet payload is required to give the detection algorithm all the required information.

Anomaly detection – The capability to employ anomaly detection mechanisms is a further requirement for highly accurate attack detection. With the increasing capacities of network links, pure knowledge-based detection systems suffer from their inability to process every single data packet. By employing statistical methods for anomaly detection, high-speed detection engines can be realized. Anomaly detection also allows to detect new kinds of attacks, or slightly modified variants of known ones, that cannot be detected by knowledge-based systems.

3.3. Countermeasures

Finally, a successful network security environment must contain appropriate mechanisms for counteracting ongoing attacks or for preventing them. Primarily, packet filters, or firewalls, are employed for preventing unwanted traffic entering a network domain. Regarding the performance of the network monitoring and the subsequent attack detection, such firewalls have a large impact. Based on the detected attack packets and the resulting filtering rules, much less traffic will arrive at the monitoring probes and, therefore, much less traffic must be handled by the intrusion detection system. In this paper, we summarize all detected attacks in a blacklist that, finally, represents the firewall system.

3.4. CATS – Attack Detection using Cooperating Autonomous Detection Systems

The objective of this section is to describe an approach for attack detection using cooperative autonomous detection systems. This system, CATS [11], is one of the first approaches that provide an architecture clearly split into the mentioned three parts: monitoring, analysis, counteracting. Additionally, aspects of distributed operation are included into the monitoring part as well as into the analyzing part.

The architecture of an individual detection system is depicted in Fig 4. It consists of an outer part for network monitoring and an inner part for detection. The network monitoring part is responsible for capturing packets and flow statistics from the network, either directly using a connected network interface, or by employing monitoring probes and the standardized protocols IP flow information export (IPFIX) [5, 23] and packet sampling (PSAMP) [7, 15]. This part also performs necessary preprocessing of the gathered data, such as packet filtering or generation of statistical flow measurements needed by the detection part. It is further divided into a layer for packet monitoring and sampling and a layer for statistical measurements. The detection part is divided into two detection engines, one providing statistical anomaly detection and the other applying knowledge-based detection mechanisms. The required packet data and statistical measures are provided by the network monitoring part.

The main reason for separating the network monitoring part and the detection part is to allow for a multi-hierarchy monitoring environment for capturing packets and flow statistics. The metering NSLP protocol [14] can be employed for the configuration of the monitoring environment. This allows for deploying one detection system that analyzes data monitored at different points of the network. Furthermore, a detection system can become itself a source of information to other detection systems by exporting monitoring data.

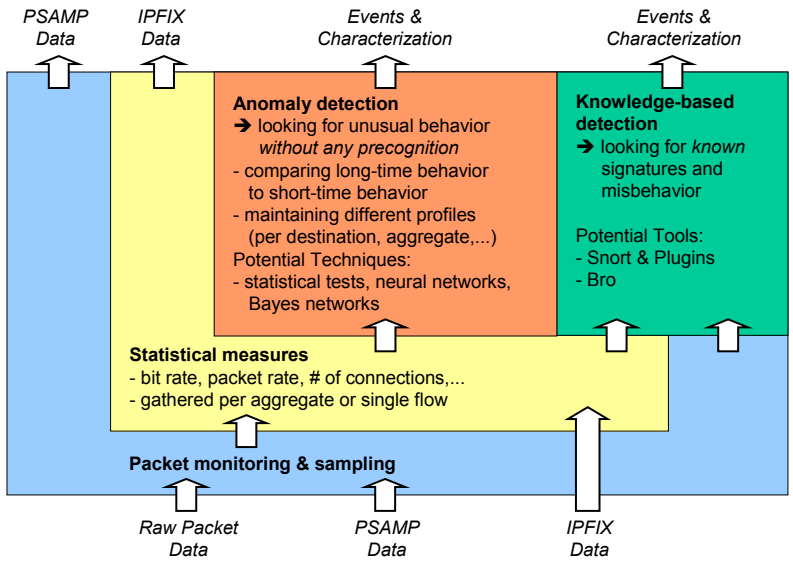

Fig 4. Architecture of our novel autonomous detection system

In the following subsections, the network monitoring part and the detection part of the detection system are described in more detail. This and additional information on CATS can be found in [11].

Packet monitoring and sampling layer

The architecture of our detection system allows two ways to capture packet data from the network: by using a directly connected NIC, and by employing PSAMP exporters, which send the collected information in a standardized way. The packet monitoring and sampling layer is responsible for capturing of packet data received via NICs or PSAMP. Moreover this layer may preprocess the packet data. Filters or sampling algorithms may be applied to reduce the amount of packets being further processed.
Within the detection system, the collected packet data is used for two purposes. First, it can be directly passed on to the detection part in order to look for known attack signatures. Secondly, it can be forwarded to the statistical measurement layer that generates flow statistics from the packet data. Additionally, the detection system can export packet data to other detection systems using PSAMP.

Statistical measurement layer

The statistical measurement layer generates statistical flow measures based on the packet data received by the packet monitoring and sampling layer, and the flow statistics received via IPFIX. Examples for statistical measures are the number of bytes and packets per flow or per aggregate, the number of connections per time, and the number of similar connections. The resulting statistical measures build the basis for further anomaly detections. For instance, an unusually high connection rate may indicate a distributed denial of service attack where typically each connection consists of only a single packet.
The statistical measurement layer does not only provide the data for the local detection mechanisms. It may also export the generated flow statistics via IPFIX. Using the terms of IPFIX [7], this corresponds to the functionality of an exporter or concentrator.

Attack detection

In the detection system, we integrate two separate, independently working detection engines – an anomaly detection engine and a knowledge-based detection engine – in order to achieve high detection rates. The detection of an attack results in the generation of an event that is combined with additional information for characterizing the attack. This information can be exchanged with other detection systems in order to improve the detection performance. On the other hand, it can be used to trigger appropriate countermeasures.
The anomaly detection works on statistical data received from the lower statistical measurement layer. This detection process is looking for unusual behavior without any precognition. It compares long-time behavior to short-time behavior and maintains different profiles, e.g. per destination, aggregate, and others. Potential techniques are statistical tests, neural networks, and Bayes networks. The architecture of our autonomous detection system allows to integrate a variety of other detection algorithms.
The knowledge-based approach represents the second main pillar of our detection engine. This engine searches the packet stream for known signatures and misbehaviors. Open-source tools such as snort [2, 3] and Bro [22], which are widely used in the Internet community, build the basis for this part of the detection.

Cooperation of multiple autonomous detection systems

So far, an individual, autonomously operating detection system can achieve a good detection rate by incorporating features from different approaches: knowledge-based detection and anomaly analysis. Additionally, the possibility to use a nearly unlimited monitoring network allows to gather packet data from

multiple points in the network. In this section we show how the detection quality can be enhanced further by loosely coupling multiple autonomous detection systems to cooperating ones, which additionally improves the overall detection quality.
In multi-gigabit networks, the capacity of monitoring probes and detection systems is limited. Frequently, sampling algorithms are employed for coping with the high data rates, which also drop packets belonging to an attack. We address the problem by creating state information for all suspicious data flows. Starting from a first assumption of an ongoing attack, the detection system has to refine the analysis in order to confirm or reject the assumption. In a first step, the aggregation level will be decreased until the potential attack flows can be isolated and a corresponding filter rule can be formulated. Subsequently, the filters at the monitoring probe and the detection systems can then be programmed in order to capture and analyze all packets belonging to the suspicious flows. Ideally, sampling algorithms are applied only on packets that do not belong to suspicious flows. In case that deeper analysis confirms an initial assumption, the state information is sent to other detection systems. Other systems that have not yet detected the same attack flow proceed as if the state information was a local assumption, trying to confirm or reject it by refining the analysis. As a result, the detection system can either affirm that it observes the same attack flow, or it dismisses the state information. Therefore, the cooperation of detection systems allows to improve the detection rate significantly and helps to identify the path of the attack flows through the network. In summary it can be said, that CATS allows an adaptive monitoring and subsequent analysis. In the following section, we show the advantages of such an self-organizing system by evaluating the number of packets at the monitoring probe and at the detection system.

4. Adaptive Re-Configuration

In the context of this section, we discuss the possibility of an adaptive re-configuration of the monitoring environment dependent on the current situation in the network. The final goal is to show the possibility of a self-organizing monitoring architecture that still fulfills its role to collect as much as possible packets concentrating on the really necessary data sets. First, the problem is described including the primary objectives. Secondly, a model is provided which fulfills the requirements and builds a basis for the simulative verification which is presented as the final part in this section.

4.1. Problem Description and Objectives

It is important to prevent the attack detection system from being overloaded by the monitoring probes in order to prevent the detection system from becoming a target itself and to increase the availability of the overall system. Even though each subsystem can perform attack detection autonomously, an overloaded single system might miss the important packets that build a primary attack accompanied by a large amount of meaningless packets. To achieve this goal, an autonomous behavior is considered with the following capabilities:
Self-Configuration – A first requirement for autonomous behavior is the capability of self-configuration. Starting from a master configuration, or even starting from scratch, the system must be capable to set all required configuration parameters, such as the current location of the probe or the type and number of neighboring entities to which communication relationships are to be applied.
Self-Maintenance – Self-maintenance is the process of adapting the configuration parameters to the current situation. Autonomously working entities must be capable to adapt to a changing environment. This adaptation, typically realized by reconfiguration of runtime parameters, comprises of changes in the resource management and in the configuration of tasks and processes.
Self-Healing – Self-healing is an important function of autonomously working entities. In the case of problems, mechanisms must be available which determine the kind of problem and initiate a healing process. For example, if the system faces memory shortages, the attack detection must be modified by selecting algorithms and parameters which require less memory (while typically resulting in a lower detection rate).
Self-Optimization – Finally, self-optimization is an important requirement for autonomous systems. In this context we understand self-optimization as the ability to optimize the detection quality. This can be achieved by exchanging information about already identified attacks or suspicious network connections and also by statistically forwarding parts of collected data packets and network statistics to neighboring probes.

4.2. Model and Solution

In Fig 5, a model is shown that represents the considered architecture including all necessary components. While focusing on the monitoring part, we want to reduce, or more precisely, to adapt the rate of packets sent to the attack

detection system. This can be done in at least three ways as shown in the following.

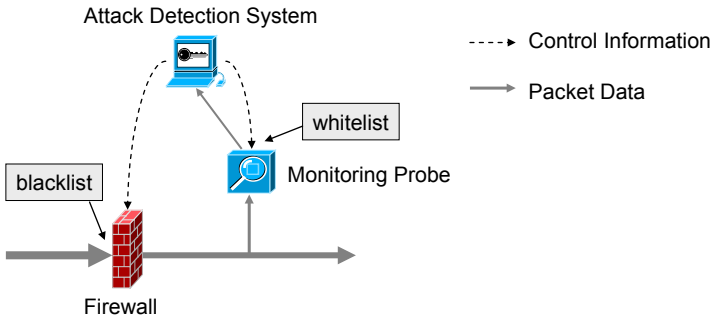

Fig 5. Model for Simulations. Shown are the main components (monitor, attack detection, firewall) and the control and data flows

Compression/encoding – Monitored packet data can be encoded in a way reducing the number of transmitted bytes to the absolute minimum. Netflow accounting / IPFIX as mentioned before can help at this place. Nevertheless, during a DDoS attack, nearly every packet represents a separate flow and, therefore, the speedup is questionable in this case. Additionally, payload information is lost using such techniques which is necessarily required for most detection techniques. Therefore, methodology based approaches are required.
Blacklists / whitelists – Blacklists in packet filtering systems, i.e. firewalls, represent a functionality having two advantages. First, the packets are prevented from reaching the systems under attack and, secondly, these packets no longer reach the monitoring system and, therefore, the data rate from the monitoring systems to the attack detection systems is reduced. Additionally, whitelists can be used at the monitoring probes to not to reduce the amount of monitored data but to reduce the amount of data transmitted to the analyzers. In summary, blacklists represent hosts involved in an attack and whitelists represent legitimate traffic.
Feedback – What finally can be done is a methodological approach that is based on a parameterization that must be adapted to the current situation in the network. The attack detection system can communicate its current load to the monitoring probes. These can adapt sampling algorithms based on the number of packets received from the network, the number of packets reported to the detection systems, and the current load of the analyzers. In order to show the potentials of this feedback, we executed a set of simulations showing the reduction of packet data that is to be received and processed by the attack detection systems.

4.3. Simulation Results

The model shown in the last subsection was implemented in a simulation environment using AnyLogic. For simulating the input packet data, we used a packet trace taken in front of out workgroup server. In Fig 6, an overview is provided comparing the input and output rates of the monitor. The monitor is executing a sampling algorithm that selects 50% of the packets (count-based).Obviously, the output packet rate is about one half of the input packet rate and the output byte rate represents the reduction due to keeping only parts of the packet (IP header including transport protocol information).

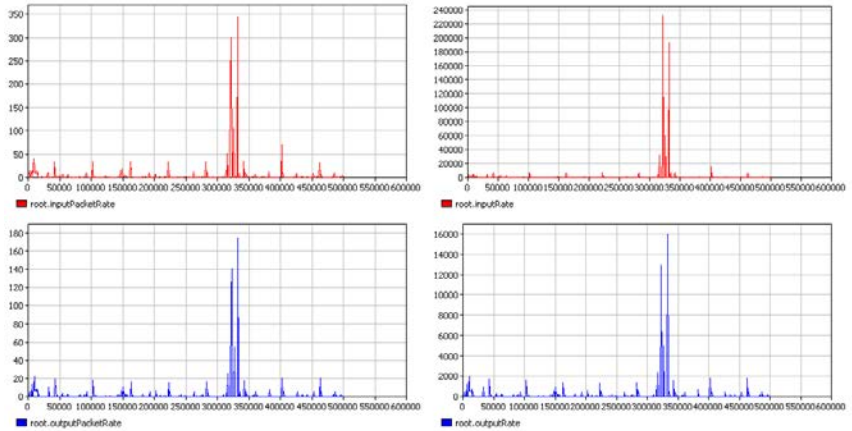

Fig 6. Comparison between input packet rate / input byte rate at a monitoring system and the corresponding output rate using an count-based sampling algorithm

In the following, we used an implemented blacklist and a whitelist representing the detected attack and legitimate flows. The parameters to look for are:

- Timout – A timeout value associated to each entry in the blacklist and whitelist. This defined how long an entry as determined by the attack detection system will be valid in the firewall system and the monitoring probe, respectively.
- Detection ratio – We assume a constant detection ratio resulting in new blacklist / whitelist entries. This is a presumption that does not correctly

correspond with the behavior in a real network. Nevertheless, it reflects the behavior of the global system pretty well due to the proper configuration of the blacklist / whitelist.

In the following simulation results, we always display the output packet rate as issued by the monitoring probe and the input rate as a reference, because this reflects the corresponding amount of data that is to be processed by the detection system.

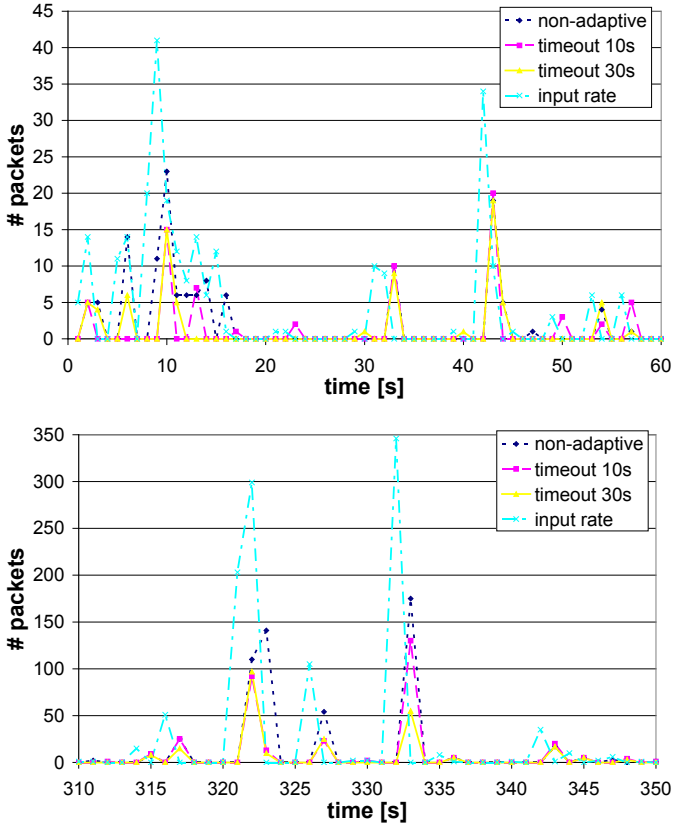

Fig 7. Simulation results: modified timeout, fixed detection ratio (10% whitelist, 1% blacklist). Shown are two magnified parts of the complete measurement

In the first simulations, we examined the effect of the timeout value associated with the single entries in the blacklist and whitelist. For analyzing this behavior, we statically configured a detection ratio of 10% for new whitelist entries and 1% for new blacklist entries. These values seem to be adequate to reflect the behavior in real networks because about ten times as mush flows can be detected as legitimate traffic that as attack traffic and most of the network packets are not directly categorizable.

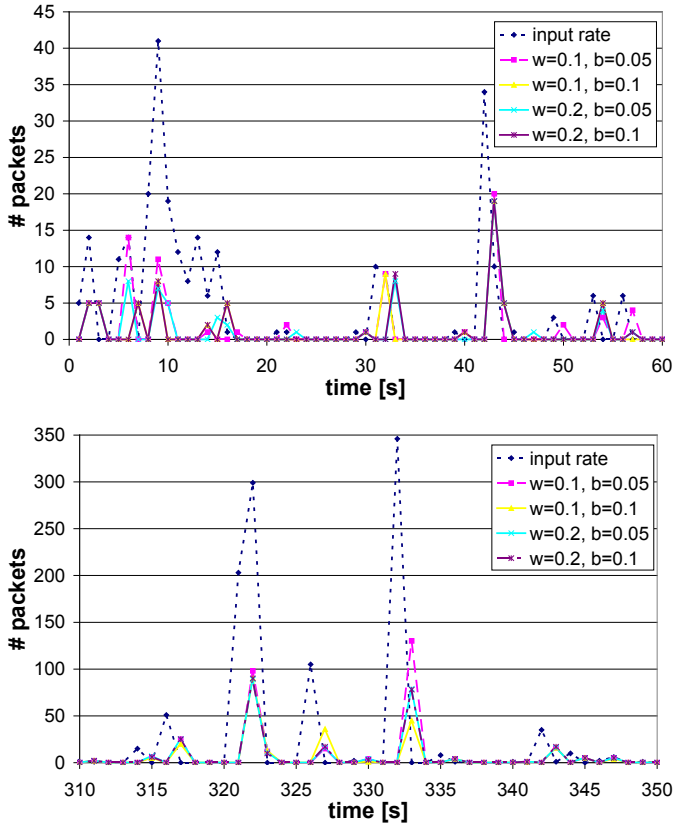

Fig 8. Simulation results: modified detection ratio, fixed timeout (10s). Shown are two magnified parts of the complete measurement

In Fig 7, the simulation results are shown. Obviously, the adaptive configuration has a large impact on the amount of output packets. Additionally, it can be seen

that a large timeout, which results also in large blacklist / whitelist tables and therefore, exhaustive search operations, does not lead to a massive reduction of the output packet rate. The reason for this behavior is the relatively short time a flow is lasting in the network.
In Fig 8, a second simulation run is shown. This time, the timeout was fixed at 10 seconds and the detection ration was modified. Interestingly, especially the whitelist has not that large impact on the amount of packets sent to the detection system as expected by the percentage of "white" packets (up to 20%). Additionally, it can be seen that the amount of data presented by the monitoring probe to the attack detection system can be adapted using an information exchange between all involved systems.
An optimal adaptation to the behavior of the network seems to be very important and will lead to an optimized global system. This optimization step can be executed independently by each participating entity in the network by two meanings: local parameterization and communication / interoperation of neighboring entities. Therefore, we have shown that it is possible to self-organize the complete monitoring and analyzing environment for network security enhancement focusing on the monitoring part.

5. Conclusions

In this paper we have shown that the monitoring data can easily be adapted to the available resources at an analyzing attack detection system. The monitoring environment already has all required facilities to ensure this kind of operations which just have to be enabled and tuned to the current network behavior. The shown feedback mechanism is part of the CATS system and is expected to be employed in large network security environments. In this paper we focused on the analysis of the possibilities of reducing / adjusting the packet rate that is to be processed at the detection system. It was shown that this amount can be easily reduced by parameterizing the algorithm parameters. In summary it can be said, that we the results of the simulation be used for a first configuration of the monitoring part of any network security environment that is based on a differentiated monitoring and analyzing part.

Acknowledgements

This work is part of collaborative research work conducted together with Prof. Georg Carle and Gerhard Münz from University of Tübingen, Germany. CATS was developed in the research project DIADEM funded by the European Commission.

References

[1] R. Bace and P. Mell, "Intrusion Detection Systems," National Institute of Standards and Technology, NIST Computer Security Special Publication SP 800-31, November 2001.
[2] J. Beale and B. Caswell, Snort 2.1 Intrusion Detection, 2nd edition ed, Syngress, 2004.
[3] B. Caswell and J. Hewlett, "Snort Users Manual," The Snort Project, Manual, May 2004. (http://www.snort.org/docs/snort_manual.pdf)
[4] R. K. C. Chang, "Defending against Flooding-Based Distributed Denial-of-Service Attacks: A Tutorial," IEEE Communications Magazine, vol. 10, pp. 42-51, October 2002.
[5] B. Claise, M. Fullmer, P. Calato, and R. Penno, "IPFIX Protocol Specifications," draft-ietf-ipfix-protocol-03.txt, February 2004.
[6] B. Claise, "Cisco Systems NetFlow Services Export Version 9," RFC 3954, October 2004.
[7] B. Claise, "Packet Sampling (PSAMP) Protocol Specifications," draft-ietf-psamp-protocol-01.txt, February 2004.
[8] B. Claise, "IPFIX Protocol Specification," Internet-Draft, draft-ietf-ipfix-protocol-08.txt, February 2005.
[9] L. Deri, "nProbe: an Open Source NetFlow Probe for Gigabit Networks," Proceedings of TERENA Networking Conference (TNC 2003), Zagreb, Croatia, May 2003.
[10] T. Dietz, F. Dressler, G. Carle, and B. Claise, "Information Model for Packet Sampling Exports," Internet-Draft, draft-ietf-psamp-info-02.txt, July 2004.
[11] F. Dressler, G. Münz, and G. Carle, "Attack Detection using Cooperating Autonomous Detection Systems (CATS)," Proceedings of 1st IFIP TC6 WG6.6 International Workshop on Autonomic Communication (WAC 2004), Berlin, Germany, October 2004.
[12] F. Dressler, C. Sommer, and G. Münz, "IPFIX Aggregation," Internet-Draft, draft-dressler-ipfix-aggregation-00.txt, January 2005.
[13] F. Dressler and G. Carle, "HISTORY - High Speed Network Monitoring and Analysis," Proceedings of 24th IEEE Conference on Computer Communications (IEEE INFOCOM 2005), Miami, FL, USA, March 2005.
[14] F. Dressler, G. Carle, J. Quittek, C. Kappler, and H. Tschofenig, "NSLP for Metering Configuration Signaling," Internet-Draft, draft-dressler-nsis-metering-nslp-01.txt, February 2005.

[15] N. Duffield, "A Framework for Packet Selection and Reporting," draft-ietf-psamp-framework-05.txt, December 2003.
[16] N. Duffield, "A Framework for Packet Selection and Reporting," Internet-Draft, draft-ietf-psamp-framework-10.txt, January 2005.
[17] M. Handley, "Internet Denial of Service Considerations," draft-iab-dos-00.txt, January 2004.
[18] R. Kemmerer and G. Vigna, "Intrusion Detection: A Brief History and Overview," IEEE Computer, pp. 27-30, April 2002.
[19] R. B. Lee, "Taxonomies of Distributed Denial of Service Networks, Attacks, Tools, and Countermeasures," Princeton University, Technical Report, 2004.
[20] T.-H. Lee, W.-K. Wu, and T.-Y. W. Huang, "Scalable Packet Digesting Schemes for IP Traceback," Proceedings of IEEE International Conference on Communications, Paris, France, June 2004.
[21] J. Mirkovic and P. Reiher, "A Taxonomy of DDoS Attack and DDoS Defense Mechanisms," ACM SIGCOMM Computer Communication Review, vol. 34, pp. 39-53, April 2004.
[22] V. Paxson, "Bro: A System for Detecting Network Intruders in Real-Time," Comuter Networks, vol. 31, pp. 2435-2463, 1999-12-14 1999.
[23] J. Quittek, T. Zseby, B. Claise, and S. Zander, "Requirements for IP Flow Information Export (IPFIX)," RFC 3917, October 2004.
[24] J. Quittek, S. Bryant, and J. Meyer, "Information Model for IP Flow Information Export," Internet-Draft, draft-ietf-ipfix-info-06.txt, February 2005.
[25] T. Zseby, M. Molina, N. Duffield, S. Niccolini, and F. Raspall, "Sampling and Filtering Techniques for IP Packet Selection," draft-ietf-psamp-sample-tech-06.txt, February 2005.

Biography

Dr. Dressler is an Assistant Professor leading the Autonomic Networking Group at the Department of Computer Sciences at the University of Erlangen-Nuremberg, Germany. He received his M.Sc. in Computer Science and his Ph.D. in 1998 and 2003, respectively.

In 2003, Dr. Dressler joined the Networking Group at the Wilhelm-Schickard-Institute for Computer Science at the University of Tuebingen. In 2004, he joined the Computer Networks and Communication systems Group at the Department of Computer Sciences at the University of Erlangen-Nuremberg.

Dr. Dressler is a member of ACM, IEEE, ACM SIGCOMM, and GI (Gesellschaft für Informatik). His primary research interests are focused on the research area of self-organizing autonomous systems. Currently, his group is working in projects on sensor networks and robotics, bio-inspired network technology, autonomic networking, network security, and network monitoring. He is author or co-author of more than 40 publications in these areas.