# Signature Detection in Sampled Packets

Gerhard Münz, Nico Weber, Georg Carle
Computer Networks and Internet
Wilhelm Schickard Institute for Computer Science
University of Tuebingen, Germany

***Abstract*—Deep packet inspection and payload analysis is required for various purposes such as the detection and identification of attacks as well as service and application-level analysis of packet streams. However, network-wide deployment of full-fledged network analyzers and intrusion detection systems is a very costly solution, especially in large networks and at high link speeds. On the other hand, modern routers, switches and monitoring probes are equipped with the capability to capture and export selected packet data to a remote collector. We developed and implemented a traffic analysis system which is able to apply online pattern matching to the received packet data, e.g. in order to detect known attack signatures. As bandwidth and computational resources are limited, it is necessary to restrict the amount of packet data that is captured and exported. Therefore, we analyzed rule sets of the popular Snort intrusion detection systems and determined which parts of a packet are relevant for signature detection and which parts can be removed without impairing the detection quality.**

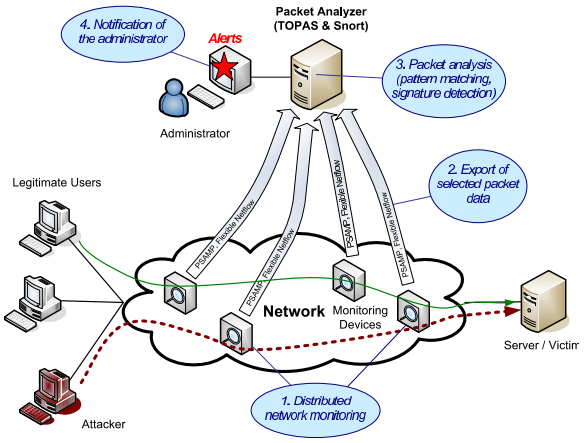


Fig. 1. Architecture and Example Scenario

## I. Introduction

Searching packet headers and payload for known patterns (that are called signatures) is a common technique to detect packets with malicious or harmful content, such as infectious worm code or exploits that can provide illegitimate access to system resources on the target host. The signatures describe characteristic header fields and strings in the payload. For traffic analysis, packet inspection and payload analysis becomes increasingly important, too, as even for basic classification of traffic into different service classes or application classes, a classification by port numbers leads to inaccurate results [1], [2]. This is due to the fact that many applications, such as peer-to-peer applications or Skype, use dynamic port allocation or reuse port numbers assigned to standard services (e.g. HTTP) to circumvent conventional port-based classification and firewall policies.

Traditionally, packet inspection is performed by network analyzers and intrusion detection systems. However, network-wide deployment of such systems at multiple observation points is costly. Furthermore, pattern matching is a computationally complex task, which makes it difficult to analyze packets at the high rates that occur in high-speed networks. Optimized parallel search algorithms implemented in hardware enable string matching at several gigabits per second [3]. However, these algorithms are of limited utility for signature analysis as signatures frequently contain regular expressions. Moreover, using specific hardware components increases the costs of network analyzers and intrusion detection systems significantly.

In this paper, we describe a cost-effective alternative based on packet monitoring capabilities of modern routers, switches, and monitoring probes. The PSAMP framework and protocol [4], [5] standardized by the IETF as well as Cisco's Flexible Netflow technology [6] allow capturing and exporting header and payload data of individual packets using the IPFIX and Netflow.v9 protocols [7], [8] respectively. As shown in Figure 1, we collect the exported packet data at a packet analyzer which performs online pattern matching and raises an alert if an attack signature is found. For proof of concept, the proposed architecture has been implemented by integrating the popular open-source network intrusion detection system *Snort* [9] as detection module into our real-time traffic analysis framework TOPAS (Traffic flOw and Packet Analysis System) [10]. A brief introduction to Flexible Netflow and PSAMP as well as some more details about the architecture and implementation are given in Section II.

In our approach, processing of the performance critical pattern matching tasks is shifted from network analyzers and intrusion detection systems deployed in the network to a remote packet analyzer. Hence, the processing resources of the packet analyzer can be flexibly used to inspect packets observed by any monitoring device. This results in a better resource utilization if the rate of packets to be inspected at an observation point is varying over time, e.g. due to changing traffic conditions or analysis interests.

The export of packet data from the monitoring devices to the packet analyzer may cause high traffic volumes, exceeding the available or acceptable amount of bandwidth. In order to cope

with limited bandwidth as well as limited processing capacities at the packet analyzer, the packet analysis has to be restricted to a maximum data rate and packet rate. For this purpose, appropriate packet selection strategies have to be identified that, on the one hand, are simple enough to be implemented at the monitoring devices and, on the other hand, ensure that the exported data is sufficient to achieve the analysis goals. Section III presents possible solutions to this problem.

Usually, it is not necessary to export entire packets as the packet inspection concentrates on a limited number of header fields and parts of payload. By adapting the content of the exported packet data accordingly, the required bandwidth can be reduced further without impairing the analysis results. We conducted a statistical analysis of the Snort rule set in order to determine which header fields are most relevant for the rule set, and how many bytes of payload are typically considered for signature-based attack detection. The outcome of the rule set study is presented in Section IV.

Section V concludes this paper with some final remarks.

## II. Architecture and Implementation

In this section, we describe the architecture and implementation of our system depicted in Figure 1. Therefore, we start with a brief introduction to PSAMP and Flexible Netflow. Subsequently, we explain the integration of Snort with the real-time traffic analysis framework TOPAS.

### A. PSAMP and Flexible Netflow

The IETF PSAMP working group has been developing techniques for selecting packets captured at an observation point and exporting packet data to a remote analyzer [4]. Several filters and sampling mechanisms have been standardized [11] enabling deterministic as well as probabilistic packet selections. PSAMP makes use of the IPFIX protocol [7] to export packet records including header and/or payload information of the selected packets. As the actual record structure can be defined in a flexible way using templates, it is possible to export only those parts of a packet which are needed for later analysis.

The latest version of Cisco Netflow, called Flexible Netflow [6], already implements some of the PSAMP concepts. Although Flexible Netflow currently supports a limited number of packet selection options only and deploys Netflow.v9 [8] instead of the IPFIX protocol, we expect that PSAMP specifications will be supported by CISCO as soon as the standardization has been finished.

In Sections III and IV, we discuss which packet selection techniques are most appropriate for the purposes of signature detection, and which per-packet information should be contained in the exported packet records.

### B. Implementation

The prototype implementation of the packet analyzer builds on TOPAS and the open-source intrusion detection system Snort. TOPAS [10] was originally developed at the University of Tuebingen within the European project *Diadem Firewall* [12] for detecting DoS and DDoS attacks using flow records. TOPAS integrates a collector for receiving monitoring data exported with Netflow or IPFIX, and distributes the data to one or more detection modules which perform the actual traffic analysis and attack detection. Detection results are encoded in IDMEF (Intrusion Detection Message eXchange Format) [13] and forwarded to an event system for post-processing, such as correlation of alerts from different detection modules or triggering response mechanisms.

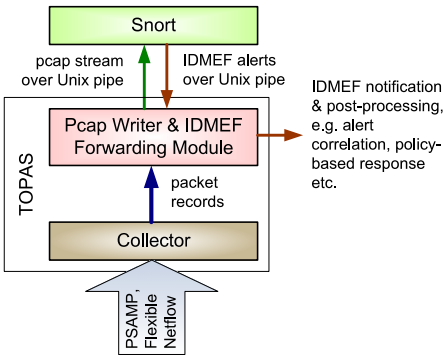


Fig. 2. TOPAS Architecture

For the purpose of signature detection in sampled packets, we integrated Snort 2.4 [14] into the TOPAS framework. As shown in Figure 2, we developed a pcap writer module that transforms packet records into frames in pcap format [15]. If a packet record does not contain a complete copy of a packet, the pcap writer module substitutes missing header fields and payload sections with padding. Via a Unix pipe, the result is passed to Snort which searches the packet for known signatures just as if it was running at the observation point. Although a tighter and faster integration of Snort is certainly possible, the pcap writer solution enables us to run Snort without any code modifications, which also facilitates keeping pace with upcoming Snort versions. In order to collect Snort alerts, we use the Snort IDMEF output plugin and a second pipe in the opposite direction. After some minor modifications in the message header, the IDMEF messages are passed on for alert notification and post-processing.

As monitoring probes, we deploy VERMONT (VERsatile MONnitoring Toolkit) [16]. VERMONT provides an interface for remote configuration based on the Netconf protocol [17] and the configuration data model proposed in [18]. This feature can be used to dynamically adapt the packet selection to the requirements of the packet analyzer as discussed in the next section.

## III. Packet Selection

As stated in the introduction, the export and analysis of packet data are subject to bandwidth and processing resource limitations. If the number of observed packets is high, we are not able to examine every packet but have to restrict the analysis to sampled packets. A good selection strategy is expected to select those packets which are relevant for the

analysis with high probability. Such a strategy ignores packets that most likely do not contain interesting information. Of course this classification into relevant and non-relevant packets depends on the analysis goal. For example, suspicious and potentially harmful packets are relevant if we intend to detect and identify worm and attack traffic. For the purpose of traffic classification, packets with characteristic payload should be chosen.

In the following subsection, we give a brief overview on standard filtering and sampling methods and discuss their appropriateness for our application. Thereafter, we describe how we can profit from configurable monitoring devices that enable the packet analyzer to dynamically adapt the selection strategy.

### A. Standard Filtering and Sampling

An overview and taxonomy of common packet filtering and sampling methods is given in [11]. A classification can be made based on the following properties:

- *Deterministic vs. nondeterministic,* depending on whether the packet selection is always identical when applied to the same sequence of packets.
- *Content-dependent vs. content-independent,* depending on whether the packet content is taken into account.

Nondeterministic content-independent methods result in a random packet selection which neither increase nor decrease the selection probability of relevant packets compared to non-relevant packets. The same applies to deterministic content-independent methods, such as *systematic count-based sampling* and *systematic time-based sampling*. However, both of these allow selecting multiple packets in a row, e.g. the first $m$ packets out of every sequence of $n > m$ packets in the case of *systematic count-based sampling*, which is an advantage if the analysis requires series of consecutive packets rather than individual packets.

Content-dependent methods, such as packet filters (deterministic) or non-uniform random sampling algorithms (nondeterministic), are the most appropriate for our purposes. They allow defining different selection probabilities depending on certain packet properties, e.g. header field values. For example, it is possible to focus on packets targeting a potentially vulnerable service on a protected host. In addition to header fields, we can take into consideration specific patterns in the packet payload, e.g. a string at a given offset in the payload. Consequently, if the packet analyzer searches for patterns or signatures containing such a string element, a very efficient packet selection can be implemented at the monitoring device. Packet selection can rely on sophisticated patterns, as long as the necessary operations are not too complex to slow down the monitoring process unacceptably.

### B. Adaptive Packet Selection

This subsection describes how a dynamic adaptation of the selection strategy can help to achieve better analysis results. Therefore, we consider packets to be part of flows, defined as unidirectional streams of packets between two communication endpoints. In many cases, the analysis goal is to classify flows and not individual packets. This is quite obvious in the case of traffic classification where flows are to be assigned to service and application classes. Similarly, worm and attack detection aims at identifying harmful flows or connections.

For traffic analysis, some flows are more interesting than others, e.g. flows with rare or unknown sources or destinations, flows directed to vulnerable ports, or flows originating from untrusted hosts etc.. Also, traffic anomalies are usually worth being examined, e.g. if unusually high numbers of packets or flows are directed to a single destination. The adaptation of packet selection and data export parameters requires dynamically configurable monitoring devices. Another prerequisite is an interface for remote configuration which enables the packet analyzer to quickly change the packet selection strategy according to its current analysis goals and interests. A possible solution provides the configuration data model for IPFIX and PSAMP [18] in combination with the Netconf protocol [17] as implemented for Vermont [19].

While the autonomous reconfiguration of monitoring devices is subject of ongoing work, a possible application scenario can be described as follows. The packet analyzer provides the monitoring devices with a white list describing filtering rules for flows where no packet inspection is required. With these rules, most of the normal traffic is omitted from packet sampling. For any other flow, packet data is to be exported until the packet analyzer sends a stop signal to the monitoring devices. Alternatively, a maximum number of exported packets per flow could be configured that is usually sufficient to classify a flow. Depending on the analysis results, the packet analyzer may adapt the configuration of the monitoring devices, e.g. by extending the white list. A protection mechanism is necessary to cope with situations where many new flows not matching the white list risk to overwhelm the packet analyzer. One possibility is to discard packets using probabilistic packet sampling. If the analysis requires consecutive packets of a flow, we can apply hash-based filtering instead using flow key header fields as input to the hash function [11]. As a result, either all or no packets of each flow are selected.

## IV. Analysis of Snort Rules

A means to further reduce the amount of monitoring data is to export only those packet header fields and payload sections which are actually needed for packet inspection. In order to know which parts of packet are relevant for signature detection, we performed a statistical study on the Snort rule set and found out that the majority of Snort signatures refer to a few packet header fields and include additional patterns to be searched for in the packet payload. The following subsections summarize the results of the study for the whole Snort rule set (labeled *All* in the tables and diagrams) as well as for four disjoint rule categories:

- *Backdoor:* Various rules for backdoors, exploits, and suspicious shell commands (rule files: `backdoor`, `exploit`, `shellcode`, `tftp`).

TABLE I
PROTOCOLS, DESTINATION IP ADDRESSES, AND DESTINATION PORTS IN SNORT RULES

| Rule Set | All | Web | Backdoor | Service | DoS |
|---|---|---|---|---|---|
| TCP | 6494 (92.49%) | 1594 (100%) | 636 (85.83%) | 657 (96.48%) | 15 (34.09%) |
| UDP | 356 (5.07%) | 0 (0%) | 80 (10.80%) | 24 (3.52%) | 13 (29.55%) |
| ICMP | 132 (1.88%) | 0 (0%) | 1 (0.13%) | 0 (0%) | 15 (34.09%) |
| IP | 39 (0.56%) | 0 (0%) | 24 (3.24%) | 0 (0%) | 1 (2.27%) |
| Specific server(s) | 1408 (20.05%) | 958 (60.10%) | 15 (2.02%) | 414 (60.79%) | 1 (2,27%) |
| Home network | 4450 (63.38%) | 625 (39.21%) | 382 (51.55%) | 242 (35.54%) | 36 (81.82%) |
| Ext. network | 1130 (16.09%) | 11 (0.69%) | 331 (44.67%) | 23 (3.38%) | 7 (15.91%) |
| Specific address(es) | 10 (0.14%) | 0 (0%) | 0 (0%) | 2 (0.29%) | 0 (0 %) |
| Any host | 23 (0.33%) | 0 (0%) | 13 (1.75%) | 0 (0%) | 0 (0%) |
| Specific port(s) | 5679 (80.89%) | 1038 (65.12%) | 371 (50,07%) | 656 (96.33%) | 25 (56,82%) |
| Any port | 1342 (19.11%) | 556 (34.88%) | 370 (49.93%) | 25 (3.67%) | 19 (43.18%) |

TABLE II
FREQUENCIES OF OTHER RULE ATTRIBUTES

| Protocol | IP | TCP | UDP | ICMP |
|---|---|---|---|---|
| Number of rules | 39 | 6494 | 356 | 132 |
| `ip_proto` | 9 (23.08%) | 0 (0%) | 0 (0%) | 0 (0%) |
| `ipopts` | 3 (7.69%) | 0 (0%) | 0 (0%) | 0 (0%) |
| `fragbits` | 2 (5.13%) | 0 (0%) | 0 (0%) | 0 (0%) |
| `ttl` | 1 (2.56%) | 1 (0.02%) | 0 0%) | 0 (0%) |
| `dsize` | 0 (0%) | 15 (0.23%) | 4 (1.12%) | 7 (5.30%) |
| `flags` | 0 (0%) | 20 (0.31%) | 0 (0%) | 0 (0%) |
| `itype` | 0 (0%) | 0 (0%) | 0 (0%) | 131 (99.24%) |
| `icode` | 0 (0%) | 0 (0%) | 0 (0%) | 81 (61.36%) |
| `icmp_id/_seq` | 0 (0%) | 0 (0%) | 0 (0%) | 17 (12.88%) |
| `isdataat` | 0 (0%) | 128 (1.97%) | 14 (3.93%) | 0 (0%) |
| `offset` | 0 (0%) | 1497 (23.05%) | 128 (35.96%) | 0 (0%) |
| `content` | 23 (58.97%) | 5518 (84.97%) | 349 (98.03%) | 39 (29.55%) |
| `uricontent` | 0 (0%) | 1396 (21.50%) | 0 (0%) | 0 (0 %) |
| `pcre` | 0 (0%) | 4262 (65.63%) | 122 (34.27%) | 0 (0%) |
| `byte_jump/_test` | 4 (10.26%) | 0 (0%) | 137 (38.48%) | 0 (0%) |

- *Web:* Web server and client rules (`web-*`).
- *Service:* Other service-specific rules (`dns`, `finger`, `ftp`, `imap`, `mysql`, `nntp`, `oracle`, `pop2`, `pop3`, `rservices`, `smtp`, `snmp`, `sql`, `x11`).
- *DoS:* DoS and DDoS rules (`dos`, `ddos`).

The presented results are based on the official Snort 2.4 rules released for registered users in October 2006 (2006-10-04). In addition, we analyzed an older release (2005-07-27) in order to determine possible trends in the evolution of signatures. The two releases mainly differ in the number of rules in the Backdoor category, which had tripled within 14 months, and the number of Web rules, which increased by more than 40 percent.

### A. Header Fields and Rule Attributes

Each Snort rule starts with a header specifying signature values for the transport protocol as well as the source and destination addresses and ports. A well designed rule header restricts the application of the rule using information about the network topology, e.g. the address range of the protected home network or addresses of specific servers. This results in reduced processing overhead per packet and a speed-up of the packet analysis. In addition, the number of false positive alerts can be decreased, e.g. by applying service-specific rules to packets directed to corresponding servers only.

Table I shows the distribution of the protocol attributes and how the destination IP address and port number are restricted to predefined values. As can be seen, most rules refer to TCP traffic; other protocols mainly occur in the Backdoor and DoS category. Obviously, destination address and port are very frequently considered in the signatures. However, the evaluation of the destination address requires knowledge about which hosts are servers for specific services, and which hosts are clients only, information that might not be available to

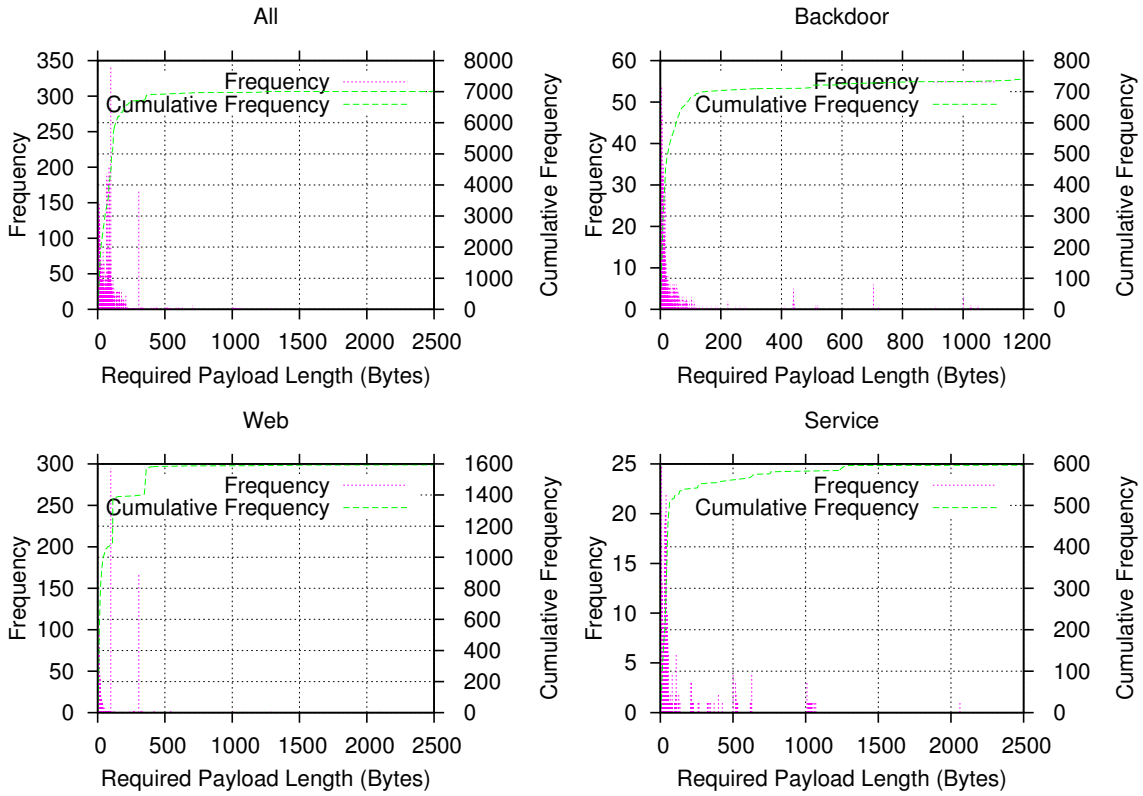


Fig. 3. Payload by Rule Set

network operators. On the other hand, the destination port can be evaluated without any further knowledge as long as services are running on well-known port numbers.

Source address and port of a packet play a less important role. The source port is set to a specific value in 13.7% of all Snort rules, mainly in the Web and Backdoor categories. The usage of the source address is negligible; only 0.5% of the rules specify a specific address or server, while most of them look for *any host*, *external network*, or *home network*.

A possibility to further eliminate false positives is to apply TCP rules to packets of established TCP connections only. In Snort, the TCP connection state is determined by a preprocessor plugin which sets the so-called `flow` attribute of a packet. This attribute is evaluated by nearly all TCP rules, and most of the time the rules refer to established connections. Consequently, it would be beneficial if the monitoring devices were able to determine the connection state and exported only packets of established connections; the `flow` attribute could then be ignored, and the preprocessor plugin deactivated.

Table II lists the frequency of other attributes used in Snort rules. Attributes concerning additional header fields are quite rare (`ip_proto`, `ipopts`, `fragbits`, `ttl`, `dsize`, `flags`). ICMP rules commonly evaluate the type (`itype`) and code (`icode`) fields. Other frequently used attributes refer to the packet payload (`isdataat`, `offset`, `content`, etc.). In the following subsection, we use the pattern length distribution in the Snort signatures to determine a reasonable number of bytes of payload that should be exported by the monitoring devices.

### B. Required Payload Length

Most Snort rules search for a given pattern in the packet payload. Since the payload of a single packet can be very long, we would like to set an upper limit for the number of exported bytes. Therefore, we used the pattern length and offset of a Snort rule to calculate the required minimum payload length. The calculated values are lower estimates based on the assumption that patterns are located at the earliest possible location in the payload unless a higher payload length is specified by a `dsize` or `isdataat` attribute. Figure 3 depicts the resulting frequency distributions.

The diagrams show that about 145 bytes of payload are necessary to satisfy 90 percent of all rules. For the older rule set release (2005-07-27), 90 percent were already reached at 100 bytes. This increase is mainly caused by newly added signatures with very long patterns in the Web category (rule file: `web-clients`) describing malicious contents, e.g. Microsoft ActiveX objects, that are downloaded from a web server. The trend towards more and more rules for such server-originated "attacks" is also reflected in the high frequency of *any* destination port in Table I which increased by a factor of ten compared to the older rule set.

### C. Conclusions

Based on our observations, we conclude that the most relevant header fields evaluated by Snort rules are the transport protocol and the source and destination port numbers. Instead of port numbers, ICMP rules require ICMP type and code values. The destination IP address can be useful for signature detection if the packet analyzer disposes of knowledge about potentially vulnerable servers in the network. The destination address is also needed to report the target of a detected attack. In addition to these header fields, most rules include a pattern in the first 145 bytes of payload. If monitoring devices exported these packet contents, 804 of 7021 Snort rules (11.4%) would not work as they evaluate other or further packet information. This number can be further decreased at the cost of enlarging the exported payload section, or exporting rarely used header fields like TCP flags. Furthermore, source addresses, although not required by the signatures, might be of interest to identify the origin of attack packets.

### D. Verification with DARPA Packet Traces

In order to verify our conclusions, we compared the detection results using a packet trace of the DARPA Intrusion Detection Evaluation 2000[1] [20] consisting of 347,987 packets. In a first run, we configured Vermont to export a PSAMP record for every observed packet, containing a copy of the entire packet and a timestamp with the observation time. This resulted in 37.6 megabytes of monitoring data, and Snort running on TOPAS detected 47 harmful packets. In a second run, we exported PSAMP records containing destination IP address, protocol identifier, 166 bytes of IP payload (transport header and payload), and the observation time. IP payload was used since the export of transport layer payload has not yet been standardized by PSAMP. The amount of monitoring data was reduced to 12.5 megabytes (66% less then before). As expected, Snort was still able to detected most of the harmful packets (41 or 87.2%). The six missed alarms were triggered by two Snort rules requiring longer payload sections.

## V. Conclusion

We presented an architecture and implementation that enables pattern matching in packet data exported by PSAMP and Flexible Netflow monitoring devices. We evaluated to which extend standard sampling and filtering methods are appropriate to select those packets with high probability that are relevant for signature detection, and we discussed the advantages of dynamically adapting the packet selection to the current analysis goals. With a statistical study of Snort rules, we gained knowledge about the parts of packet that are relevant for signature detection and that should therefore be exported by the monitoring devices.

As PSAMP is in the final phase of standardization, and as Flexible Netflow is already available on the market, we expect that the export of packet data becomes a commonplace extension for devices already supporting flow information export today, e.g. routers, switches, and monitoring probes. Based on this assumption, our approach represents a cost-effective solution to perform signature detection in high-speed networks where the distributed deployment of conventional network analyzers and intrusion detection systems would be too expensive. Our current research activities aim at validating the presented packet selection and export strategies under real traffic conditions. Furthermore, we intend to combine flow and packet-based attack detection within the TOPAS framework in order to improve the detection results of individual methods.

[1] DDoS scenario *LLDOS 2.0.2*, inside traffic

## References


[1] T. Karagiannis, K. Papagiannaki, and M. Faloutsos, "BLINC: Multilevel Traffic Classification in the Dark," in *Proc. of Conference of the Special Interest Group on Data Communication (SIGCOMM'05)*, Philadelphia, Pennsylvania, USA, Aug. 2005.

[2] A. W. Moore and K. Papagiannaki, "Toward the Accurate Identification of Network Applications," in *Proc. of Passive and Active Measurement Workshop (PAM 2005)*, Boston, MA, Mar. 2005.

[3] S. Fide and S. Jenks, "A Survey of String Matching Approaches in Hardware," Department of Electrical Engineering and Computer Science, University of California, Irvine, Tech. Rep. TR SPDS 06-01, Mar. 2006.

[4] N. Duffield, D. Chiou, B. Claise, A. Greenberg, M. Grossglauser, P. Marimuthu, J. Rexford, and G. Sadasivan, "A Framework for Packet Selection and Reporting," Internet-Draft, work in progress, draft-ietf-psamp-framework-12, Jun. 2007.

[5] B. Claise, J. Quittek, and A. Johnson, "Packet Sampling (PSAMP) Protocol Specifications," Internet-Draft, work in progress, draft-ietf-psamp-protocol-08, Jun. 2007.

[6] Cisco Systems, "Introduction to Cisco IOS Flexible NetFlow," White Paper, Jun. 2006. [Online]. Available: http://www.cisco.com

[7] B. Claise, S. Bryant, S. Leinen, T. Dietz, and B. H. Trammell, "IPFIX Protocol Specifications," Internet-Draft, work in progress, draft-ietf-ipfix-protocol-26, Sep. 2007.

[8] B. Claise, G. Sadasivan, V. Valluri, and M. Djernaes, "Cisco Systems NetFlow Services Export Version 9," RFC 3954 (Informational), Oct. 2004.

[9] M. Roesch, "Snort: Lightweight Intrusion Detection for Networks," in *Proc. of 13th USENIX Conference on System Administration*. USENIX Association, Nov. 1999.

[10] G. Münz and G. Carle, "Real-time Analysis of Flow Data for Network Attack Detection," in *Proc. of IFIP/IEEE Symposium on Integrated Management (IM 2007)*, Munich, Germany, May 2007.

[11] T. Zseby, M. Molina, N. Duffield, S. Niccolini, and F. Raspall, "Sampling and Filtering Techniques for IP Packet Selection," Internet-Draft, work in progress, draft-ietf-psamp-sample-tech-10, Jun. 2007.

[12] Diadem Firewall Homepage, http://www.diadem-firewall.org/, 2006.

[13] H. Debar, D. Curry, and B. Feinstein, "The Intrusion Detection Message Exchange Format (IDMEF)," RFC 4765 (Experimental), Mar. 2007.

[14] Snort Homepage, http://www.snort.org/, 2006.

[15] Libpcap Homepage, http://www.tcpdump.org, 2007.

[16] R. T. Lampert, C. Sommer, G. Münz, and F. Dressler, "Vermont - A Versatile Monitoring Toolkit for IPFIX and PSAMP," in *Proc. of IEEE/IST Workshop on Monitoring, Attack Detection and Mitigation (MonAM 2006)*, Tuebingen, Germany, Sep. 2006.

[17] R. Enns, A. Bierman, K. Crozier, T. Goddard, E. Lear, P. Shafer, S. Waldbusser, and M. Wasserman, "NETCONF Configuration Protocol," RFC 4741 (Standards Track), Dec. 2006.

[18] G. Münz and B. Claise, "Configuration Data Model for IPFIX and PSAMP," Internet-Draft, work in progress, draft-muenz-ipfix-configuration-02.txt, Jun. 2007.

[19] G. Münz, A. Antony, F. Dressler, and G. Carle, "Using Netconf for Configuring Monitoring Probes," in *Proc. of IEEE/IFIP Network Operations & Management Symposium (NOMS 2006), Poster Session*, Vancouver, Canada, Apr. 2006.

[20] Website of DARPA Intrusion Detection Evaluation, http://www.ll.mit.edu/IST/ideval/, 2007.