European Sixth Framework Network of Excellence FP6-2004-IST-026854-NoE

Deliverable D7.4

Large Scale Management Final Report

The EMANICS Consortium

Caisse des Dépôts et Consignations, CDC, France
Institut National de Recherche en Informatique et Automatique, INRIA, France
University of Twente, UT, The Netherlands
Imperial College, IC, UK
Jacobs University Bremen, JUB, Germany
KTH Royal Institute of Technology, KTH, Sweden
Oslo University College, HIO, Norway
Universitat Politecnica de Catalunya, UPC, Spain
University of Federal Armed Forces Munich, CETIM, Germany
Poznan Supercomputing and Networking Center, PSNC, Poland
University of Zürich, UniZH, Switzerland
Ludwig-Maximilian University Munich, LMU, Germany
University of Surrey, UniS, UK
University of Pitesti, UniP, Romania

For more information on this document or the EMANICS Project, please contact:

Dr. Olivier Festor
Technopole de Nancy-Brabois - Campus scientifique
615, rue de Jardin Botanique - B.P. 101
F-54600 Villers Les Nancy Cedex
France
Phone: +33 383 59 30 66
Fax: +33 383 41 30 79
E-mail: <olivier.festor@loria.fr>

Document Control

Title:	Large Scale Management Final Report
Type:	Public
Editor(s):	Ramin Sadre
E-mail:	sadrer@cs.utwente.nl
Author(s):	WP7 Partners
Doc ID:	D7.4

AMENDMENT HISTORY

Version	Date	Author	Description/Comments
0.1	2007-07-03	H. Tran, J. Schönwälder	Initial version of a LaTeX template
0.2	2008-11-25	R. Sadre	Initial version for integration
0.3	2008-11-26	R. Sadre	section SNAID/anomaly characterization
0.4	2008-12-04	R. Szuman	section SNAID/state-of-the-art
0.5	2008-12-15	P. Ming	section BioScale
0.6	2008-12-17	B. Stiller	section SNAID/integration flow-packet
0.7	2008-12-30	R. Sadre	Final integration

Contents

1 Executive Summary

Large scale management concerns all the five functional areas of network management. Work-package 7 addresses three of these areas: security, fault and configuration, the other two being addressed in work-packages 8 and 9. This deliverable reports the main scientific results obtained in work-package 7 of the Emanics NoE. The work in this work-package has been driven by two activities that started in April 2008 as a result of an open call for proposals, adressing two main topics: the scalable management of biometric devices and the scalability of network analysis and intrusion detection. The reported content has been also the subject of publications at scientific conferences.

2 Introduction

The activities in the work-package 7 have been driven by two main activities, that emerged from an open call for proposals in April 2008. The two selected proposals were addressing relevant topics to the scalability of network management.

The first topic was concerned with a scalable distribution system for biometric templates in large scale deployment projects. The objective of the BioScale activity was the development of a scalable management approach for biometric devices. Several tasks have been addressed and prototypes have been implemented. We report this work in section 3 of this deliverable.

The second topic focusses on the scalable network (security) monitoring. In the SNAID activity (Scalability of Network Analysis and Intrusion Detection), we addressed the issues on how traffic anomalies can be detected in large networks on flow level and how packet-based and flow-based intrusion detection can be combined. In addition, we discussed the state-of-the-art of algorithms for detection of network anomalies in sampled data and their implementations in available monitoring software. We cover this activity in the fourth section of this deliverable.

For an overall view on the activities and the involved participants, we summarize them in the following:

- BioScale: Scalable Management of Biometric devices (9 months, partners: UZH, UniBW)
- SNAID: Scalability of Network Analysis and Intrusion Detection (9 months, Partners: UT, JUB, INRIA, UZH, PSNC)

3 Scalable Management of Biometric devices (BioScale)

Biometric management software provided by manufacturers are difficult to use for medium and large scale deployments of biometric devices. One of the main challenges of such large scale deployment projects is a scalable distribution system for biometric templates. Furthermore, biometric solutions offered by manufacturers for large scale installations have usually a non standardized interface to access the functionality of their biometric devices, which makes integration in large scale projects difficult. The objective of the BioScale activity is the development of a scalable management approach for biometric devices.

3.1 Background: Biometric devices

There are several different devices available in the marked such as fingerprint scanner, face scanner, vein scanner, and many others. Almost all biometric devices show different capabilities, *e.g.,* devices might have an internal storage for user data and other devices might provide additional buttons to indicate different events. A common characteristic of most biometric devices is the function to convert biometric data into templates. A template contains biometric data processed, where supporting points for user verification are extracted.

3.2 BioScale requirements

The requirements of a large scale template distribution can be described as follows. The system must be failure tolerant against network, device failure, and user failure; in particular:

- All template distribution transactions have to be secure, reliable, and traceable.
- Comprehensive management interface, which allows to manage the access rights and disallows opposing data.
- Distributed access to the user management interface

3.3 BioScale approach

The core task of the BioScale approach is to design and implement a scalable distribution system for large scale deployments of biometric devices. The BioScale approach bases on the software BioXes, which is a middleware solution to control heterogeneous biometric devices of different manufacturers. BioXes registers and manages biometric identities and devices, and enables biometric access control and interoperability of multimodal, multi-vendor systems across heterogeneous environments. The core functionality of BioXes is a distribution plan, according to which templates are distributed in configurable intervals to the single devices. The JBioDC interface offers a common interface to connect to any device. The JBioDC interface is a high level management API for biometric devices.

3.4 BioScale Approach

In a first step the current implementation of the distributor of the BioXes application has to be measured. Based on the acquired measurement results a scalable template distribution strategy for biometric devices in a large scale deployment has to be defined. Because a test scenario of this size is resource intensive, a dummy biometric needs to be used which acts as a real life biometric device. In a next step, these dummy devices are used in the test scenario and the results have to be analyzed. Based on analysis of the results a strategy for the template distribution in a large scale environment has to be defined. A further step is to create a comprehensive user interface, that keeps track of the distribution and is easy to manage. The layout of the access rights management respectively the distribution plan has to be designed and a prototype has to be implemented. The prototype then is used for user tests in order to analyze comprehensibility of the interface. The feedback of these user tests are used in the final implementation of the user interface. The last step is to make the access to this interface distributed. For that reason a web implementation has to be implemented.

3.5 Tasks and Progress Report for BioScale

Based on the strategy, tasks are deduced and described in the following. Each task is described, progress is shown, and next steps are identified.

3.5.1 Task 1: Define configuration for BioScale

The basic configuration of a large scale template distribution is defined as follows: Over 1,000 biometric devices (of different manufacturers and/or biometric characteristics) and over 10,000 users with templates with 10 different locations. The connectivity is given through network (TCP/IP).

- Progress: The configuration of a large-scale setting is defined.
- Next steps: Not further steps are required in this task.

3.5.2 Task 2: Prioritized distributor

The first thing which has been made is to change the current BioXes distributor into a prioritized distributor. User based measurements showed that operation of BioXes is slow if distribution is in progress. This has to be done because up to now all requests from BioXes to biometric devices were handled with the same priority. Thus, the distributor implements two priorities: a low priority for the distributor and high priority for user interaction.

- Progress: The prioritization of the distributor has been implemented and tested.
- Next steps: Not further steps are required in this task.

3.5.3 Task 3: Set up the test scenario and perform the scalability test

A biometric dummy device has to be implemented. With the dummy devices the test scenario has to be set up and tests have to be performed. The dummy device should fulfill the following functionality:

- connected and disconnected device (over TCP/IP)
- setTemplate(Template template): Adds a template to a device database
- getTemplates(int uuid): retrieves a list of template of a user which are stored on a device
- getTemplate(int uuid, int tid): Get a template of a user identified by the user and template id
- deleteTemplate(int uuid, int tid): Removes a template identified by the user and template id
- deleteTemplates(int uuid): Removes all templates from a user
- getUserIDs():Returns a list of user IDs of all templates on a device
- hasTemplates(int uuid): Check if any template of a certain user is stored on the device
- hasTemplate(int uuid, int tid): Check if a template is on a device identified by user and template id

- Progress: This task has started partially. The dummy device has been implemented and tested.
- Next steps: Set up the test scenario and perform tests.

3.5.4 Task 4: Analyze measurement results

The results from the test have to be analyzed. Based on the results of the analysis a strategy for the template distribution in a large scale environment has to be defined.

- Progress: This task has not started yet.
- Next steps: Analyze the measurement results.

3.5.5 Task 5: Comprehensible user interface and visualization

Different layouts of the access management screen have to be designed and a first prototype has to be implemented. The implemented prototype has to carry out in a field test with inexperienced users to evaluate the comprehensibility of the newly designed screen. With the collected feedback from the tests the layouts has to be adjusted and a final implementation of the user screen has to be done.

- Progress: A first prototype has been implemented and first field test have been carried out. Figure 1 shows the new user interface.
- Next steps: Adapt the feedback of the field tests into the screen and rerun a field test.

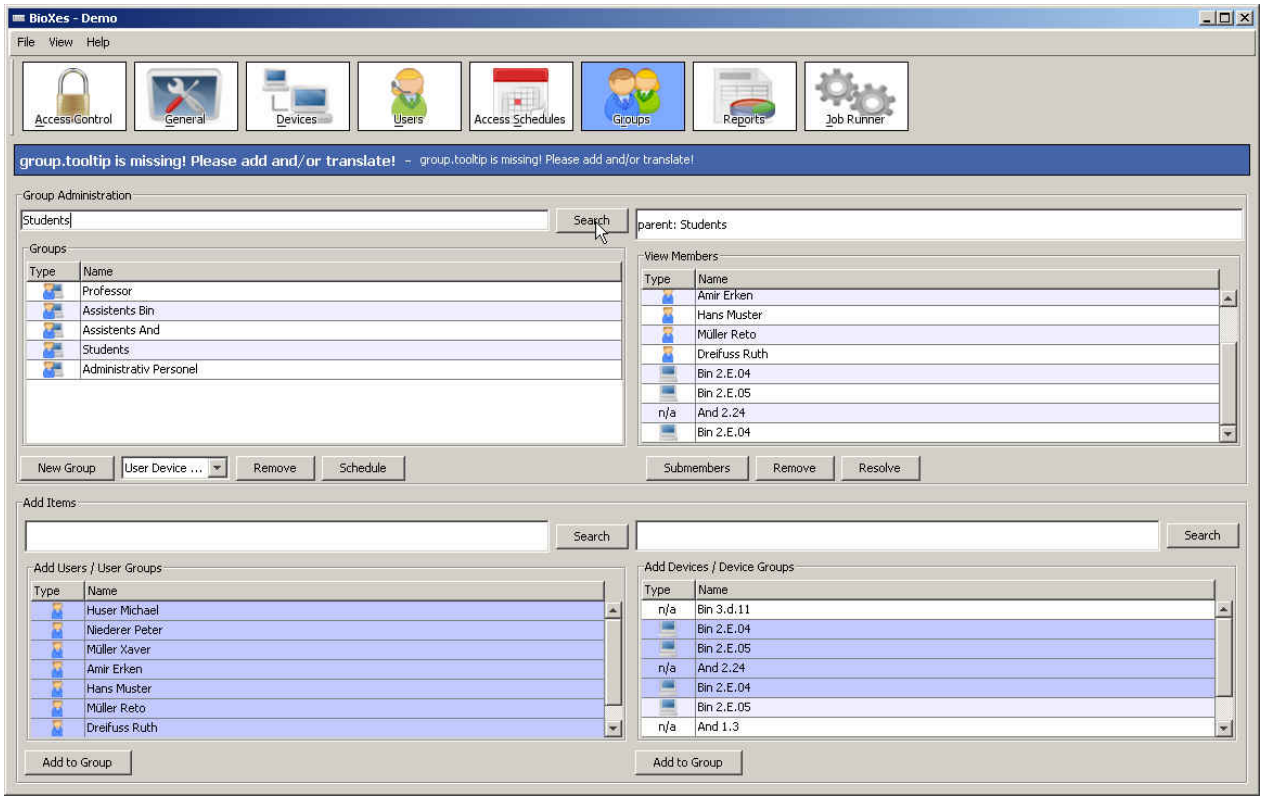

Figure 1: New user interface for managing the distribution plan

3.5.6 Task 6: Distributed user interface

In this task a first prototype of the management screens have to be implemented using the GWT technology. The screens access the existing functionality of BioXes and the user profiles can be managed over a web browser.

- Progress: A first prototype of the user management screen has been implemented. Figure 2 shows the new distributed user interface.

- Next steps: Finalize the user management screen and adopt the technologies to the other screens.

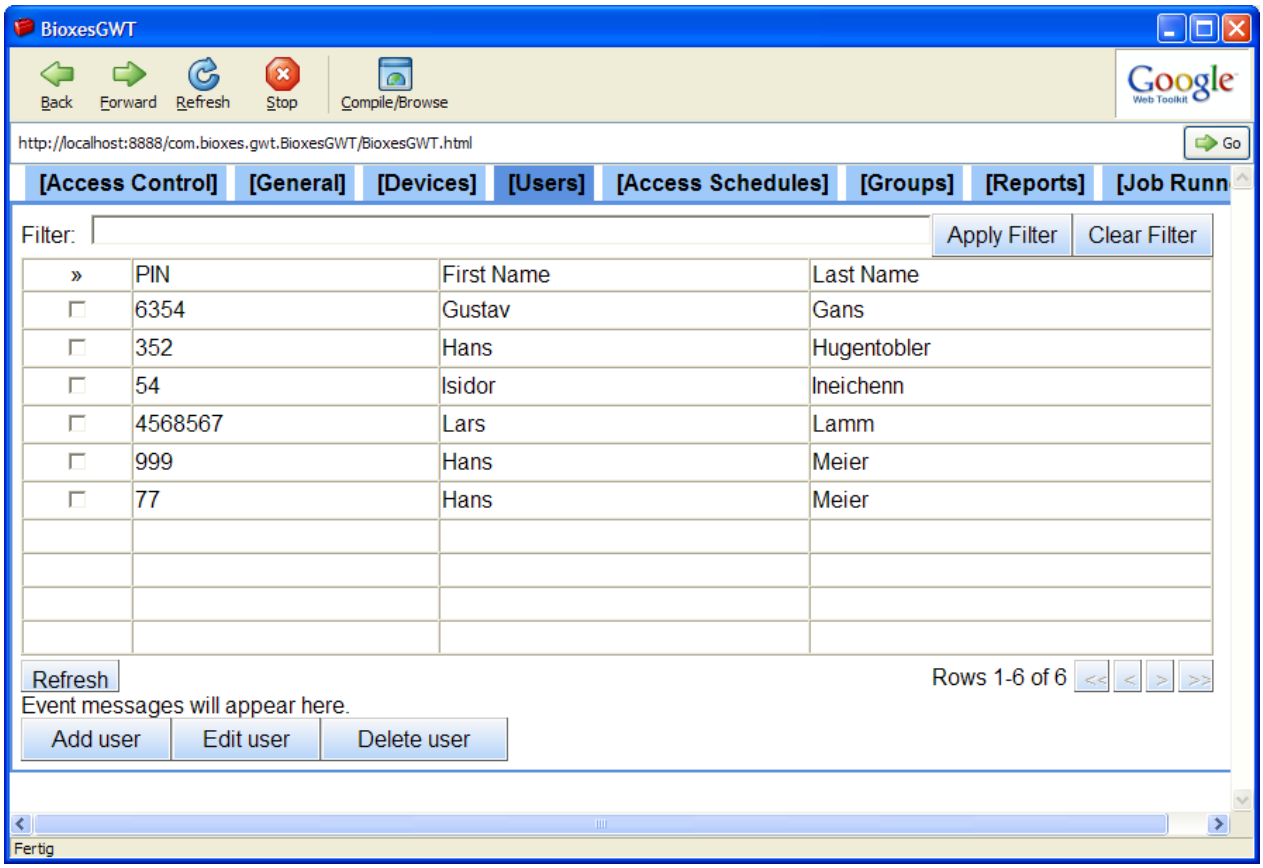

Figure 2: New distributed user interface

3.6 Further work and BioScale II

Several tasks have been addressed and prototypes have been implemented in BioScale I. However, there are still open task that will be addressed in BioScale II. In addition to the mentioned tasks, BioScale II includes unit testing, test cases, and usage scenarios.

4 Scalability of Network Analysis and Intrusion Detection (SNAID)

4.1 Anomaly characterization in flow-based traffic time series

Attacks on our network and server infrastructures are a growing source of problems for network operators and users. They may be generated by inexperienced script-kiddies, but also by professional hackers, but in any case, attacks create unwanted traffic that can affect the performance and dependability of existing services. Therefore operators employ intrusion detection systems to identify and possibly filter suspicious traffic.

The constant increase in network traffic and the fast introduction of high speed (tens of Gbps) network equipment [1] make it hard to still employ traditional packet-based intrusion detection systems. Such systems rely on deep packet payload inspection, which does not scale well. In high speed environments, approaches that rely on aggregated traffic metrics, such as *flow-based* approaches, show a better scalability and therefore seem more promising. The advantage of *flow-based* approaches is that only a fraction of the total amount of data needs to be analysed. For the University of Twente, for example, we have estimated that the amount of flow data represents less than 1% of the amount of normal packet data.

A flow is defined as an unidirectional stream of packets that share common characteristics, such as source and destination addresses, ports and protocol type. In addition, a flow includes aggregated information about the number of packets and bytes belonging to the stream, as well as its duration. Flows are often used for network monitoring, permitting to obtain a real time overview of the network status; common tools for this purpose are Nfsen [2] and Flowscan [3], while the *de facto* standard technology in this field is Cisco Netflow, particularly in its versions 5 and 9 [4, 5]. The IETF IPFIX working group [6] is currently working on a standard for IP flow exporting, based on Netflow version 9.

Large networks, when creating flows, often apply *packet sampling* in order to make the approach even more scalable. In this case, only a percentage of the total number of packets passing through the monitoring point is considered in the flows. Statistical studies have been performed about correctness and precision of sampling strategies for Internet traffic [7] and high speed environments [8]. These studies show that, despite sampling, it is still possible to offer a correct statistical overview of the network status [7]. Packet sampling in flow creation is vastly deployed [9, 10]. In particular, NetFlow relies on *systematic* sampling, where only 1 out of every n packets is considered for the accounting (1:n).

In the last years there has been an increasing interest in the application of flow-based techniques for anomaly and intrusion detection. An example is the work of [11, 12], which applies principal component analysis to traffic time series. Another example is provided by [13], which aims to detect worm spread in high speed network on a connection basis. In a similar environment, [14] addresses the problem of detecting DoS attacks and scans. In this case, the authors particularly focus on aggregated header information, as they can be exported by NetFlow (TCP flags). In addition, the presented approach is interesting because it explicitly addresses the problem of measure variation over time (with the use of value forecasting). In [15], the role of timely analysis of flow data is central. The author

proposes a general purpose platform for parallel time-based analysis of flow information for attack detection, focusing in particular on DoS attacks (SYN-flood and web server overloading). From a network monitoring point of view, time series on flows, packets, and bytes are considered to be a useful tool: they permit to have a *dynamic* and *real time* overview of the network on the basis of the stream of information coming from the exporter [9, 3].

In the following, we investigate the use of traffic time series for identifying traffic anomalies and detecting intrusions. In particular, we are interested in *whether it is necessary to consider 1) flows, 2) packets as well as 3) bytes time series, or whether it is sufficient to consider only one or two of these*. In addition, we want to know if the conclusion *also holds in the presence of sampling*. The novelty of our approach is that we rely on real case studies, performed in high-speed networks with links of 10 Gbps. Our measurements have been performed simultaneously on two different networks, the University of Twente (UT) and SURFnet, the Dutch research network, [16]. SURFnet applies 1:100 packet sampling during the flow creation.

This section is organized as follow. Section 4.1.1 presents the measurement environment in which our analysis has been conducted. Sections 4.1.2 and 4.1.4 analyze anomalies in flow traces, focusing in particular on two real examples. Finally, Section 4.1.6 presents our conclusions.

4.1.1 Measurement setup

The analysis presented in this work has been conducted on flow traces collected at the University of Twente and on the SURFnet infrastructure [16]. In particular, the analyzed traces cover a period of time of two working days, namely between Wednesday August 1st 2007, 00:00 and Thursday August 2nd 2007, 23:59. The two networks have different sizes and coverage. The UT one is a $/16$ network providing connectivity to the employees and the students on the university buildings and the campus. SURFnet has national coverage and connects via optical path the most important research institutions in the Netherlands. Since SURFnet is also the UT network service provider, the majority of the incoming and outgoing UT traffic is routed through SURFnet. UT and SURFnet traces rely on a different measurement setup. Indeed, while UT accounts all the packets passing through the measuring point, SURFnet applies a systematic sampling with ratio 1:100. In the following, the real amount of traffic is estimated scaling all the measurements by a factor of 100.

Figure 3 shows the bytes traffic time series in the considered time frame. In this and the following sections, all the time series have been created considering a time interval of 600 seconds, a good compromise between accuracy and number of samples. As expected, both networks show a clear night-day pattern, with peak of activity between 8:00 and 18:00 and with a minimum around 4:00. Around 16:00, on August 1st 2008, the amount of traffic on SURFnet drops abruptly. Since no error has been detected in our measuring setup, we suspect the down-peak to be caused by a flow creation and exporting failure in the SURFnet infrastructure, or, less likely, to a network hardware failure. Nevertheless, this event is not affecting our analysis. Table 1 also presents the average, minimum and

maximum traffic loads and the total data volume measured on the two networks during the observation period, together with the number of collected flows.

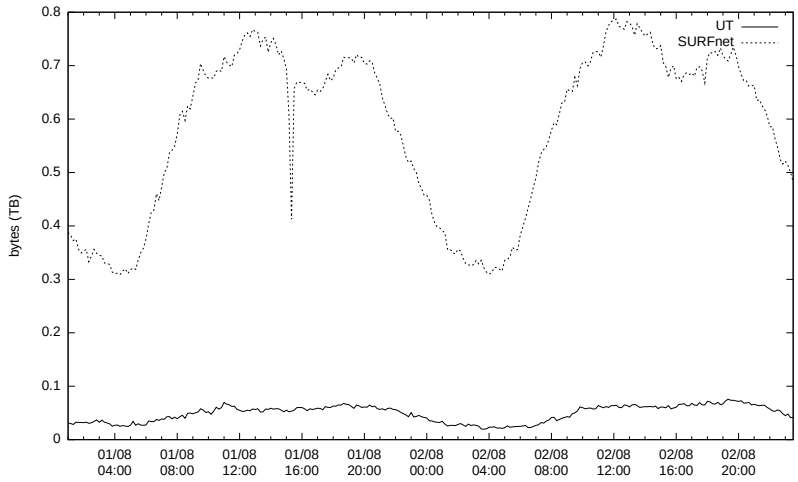

Figure 3: Bytes time series, showing UT and SURFnet (estimated values) traffic.

	Avg Load	*Max Load*	*Min Load*	*Volume*	*Flows*
UT	652Mbps	1.01Gbps	259Mbps	21.65TB	982.7M
SURFnet	7.73Gbps	10.5Gbps	4Gbps	162.3TB	523.7M

Table 1: Average, maximum and minimum traffic loads, data volume and number of flows during the period of observation on UT and SURFnet.

During our monitoring time, UT seemed to be object of repeated and diverse attacks, even if apparently without real damage. Due to space constraints, we decide to concentrate our analysis on the following examples: `ssh` and `dns` traffic traces. The choice of these two specific sub-traces is due to the fact that, quite surprisingly, the `ssh` service resulted to be one of the major attack targets, both in intensity and in number of the attacks. Similarly, by experience we noticed that `dns` tends to produce a quite regular traffic volume. This characteristic made quite easy to detect suspicious variation in traffic intensity. To properly evaluate if the observation in both networks are consistent, the `ssh` and `dns` traffic in SURFnet have been filtered in order to keep into account only the incoming-outgoing traffic from the UT network.

4.1.2 SSH traffic analysis

`Ssh` is one of the most common protocols to connect with remote machines. In general, it corresponds to the 1% of packets and the 1.2% of bytes of the total incoming-outgoing UT traffic.

Figure 4 shows the byte traffic time series in the observation time frame. In the same graph, both UT and SURFnet (estimated) traffic volume are shown. It is possible to notice that the two measurements show the same trends, and only occasionally SURFnet

strongly differs from UT. In general, the bytes trend in the observation period is quite irregular with sharp peaks and down-peaks. This situation is understandable because `ssh` can be used for both remote communications and file transfers. As a consequence, in the byte time series there is no clear evidence of attacks.

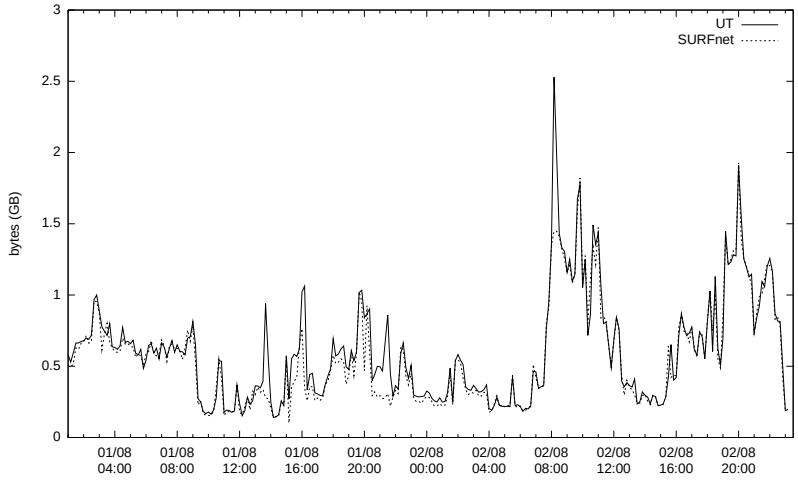

Figure 4: Bytes time series, showing UT and SURFnet (estimated values) `ssh` traffic.

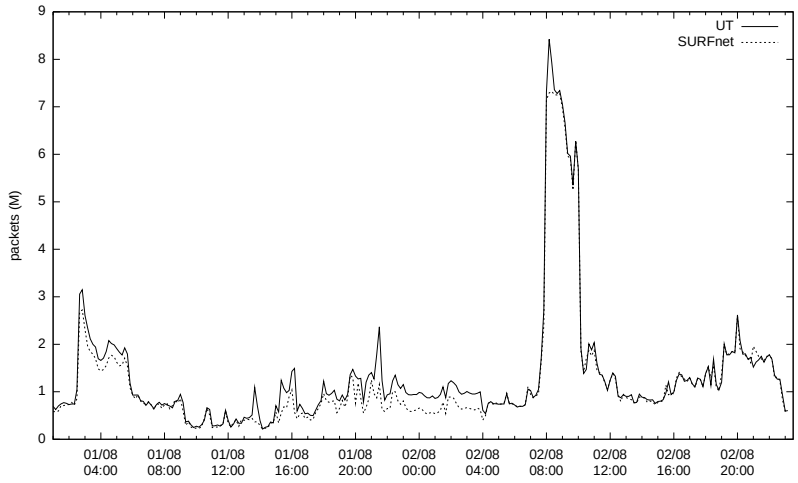

Figure 5: Packets time series, showing UT and SURFnet (estimated values) `ssh` traffic.

On the other side, looking at the packet time series (Figure 5), it is possible to notice that during the two days of observations, the UT network saw a massive increase of its `ssh` traffic. The time series is indeed characterized by sudden peaks during which the number of packets per time interval can raise of several millions. In some cases, we observe a difference of up to almost 8 millions packets. If we consider the flow time series, as in Figure 6, we can observe how the trend is also in this case characterized by peaks during which the number of flow per time interval raises form few thousand to half million. Again, the number of flows per time interval in SURFnet increases following the same behavior of the UT trace, despite the use of sampling. This phenomenon is particularly visible during the massive peaks, namely in the early morning of August 1st and in the late morning of August 2nd.

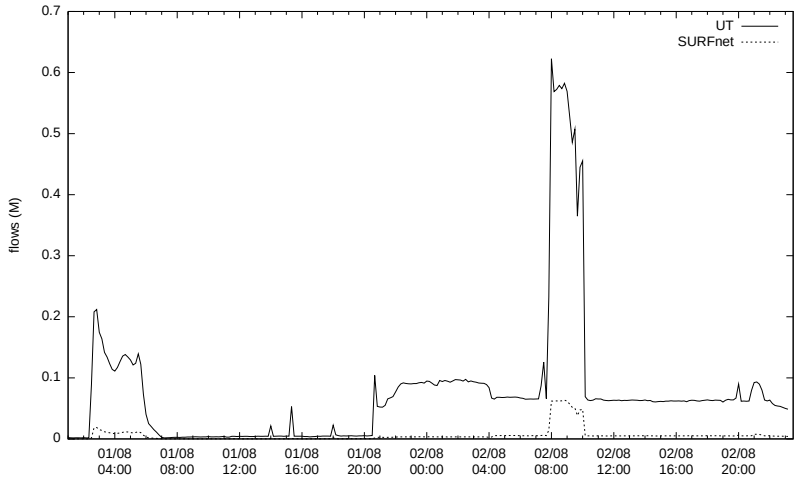

Figure 6: Flows time series, showing UT and SURFnet `ssh` traffic.

Summarizing, in the moments of major `ssh` activity, we observe a suspiciously high number of flows, matched by a very high number of packets, but with almost negligible amount of sent and received bytes. This suggest that the hosts involved are sending/receiving relatively small packets to many different hosts, scenario that suggests the possibility of a scan. A more detailed inspection of the trace shows indeed that few source hosts made the UT network object of massive `ssh` scans, during which the attackers were performing user-guessing on almost all the hosts in the UT network. It is important to underline that is the contemporaneous observation of all the three measure that permits to discriminate between normal and malicious traffic. For example, considering only peaks in the packet time series would not permit to distinguish file transfers from scanning activities. On the contrary, considering also bytes and flows would show that a file transfer has a different behavior from a scan, with peaks in the byte time series but not in the flow one. The traffic characteristics during the peaks made the `ssh` traffic trace worthy of deeper analysis. In the following, we concentrate only on the peak in the time frame from 7:50 to 10:10 on August 2nd, when the number of flows per time interval rises up to a maximum of almost 600000 flows *(`ssh`) anomalous time frame*.

4.1.3 SSH normal vs anomalous traffic

The following analysis proves indeed that the previously identified time frame is due to an attack. In order to characterize the network behavior during the anomaly, we need to compare it with a second observation time frame, that will provide us an overview of the network during a not suspicious interval. The second time windows span over a period of 2 hours, between 8:00 and 10:00 of August 1st. During this time frame, we are not observing any fast variation of the flow frequency. Since we are interested in scans and we are assuming that `ssh` scans produce variation in the flow frequency, we also assume the second time frame to be an example of *normal* network behavior (*normal time frame*).

Looking at the number of active hosts in the anomalous and normal time frames, Table 2 shows that the normal time frame is characterized by a balanced number of sources and destinations, both in UT and SURFnet. On the contrary, in the anomalous time frame,

we can observe an increased number of destinations, several times bigger than the number of sources. The number of destination hosts in the UT trace suggests that the scan covers the entire UT network (that is, as reported in Section 4.1.1, a $/16$ network), while the increased number of source hosts is an effect of the scanning activity (some of the destination hosts react to the probes). A similar trend is visible in SURFnet.

	Anomalous time frame		*Normal time frame*	
	Sources	*Destinations*	*Sources*	*Destinations*
UT	2763	65342	629	647
SURFnet	597	3020	192	192

Table 2: Number of distinct source and destination addresses during the anomalous and normal time frames in the UT and SURFnet traces.

The study of the top active sources w.r.t. the number of originated flows shows that the anomalous time frame is dominated by the presence of three major senders, that caused the attack. Table 3 shows how the traffic, expressed in flows, packets and bytes, is distributed with respect to the sources during the anomalous time frame. Together, the three most active sources are responsible of the 98 - 99% of the total amount of flows in both UT and SURFnet. All the three hosts were scanning the UT network. As already suspected during the time series analysis, also the packet repartition is unbalanced towards the major senders (responsible of $\sim 70\%$ of the packets in both UT and SURFnet). Finally, it is important to notice that the scan *does not* deeply affect the bytes distribution: the 75% and the 69% of the bytes volumes respectively in UT and SURFnet is still due to normal traffic.

	Flow Percentage		*Packets Percentage*		*Bytes Percentage*	
	UT	*SURFnet*	*UT*	*SURFnet*	*UT*	*SURFnet*
SSH TOP 1	82.6%	89.5%	65.7%	71.2%	22.3%	28.1%
SSH TOP 2	13.5%	9.2%	6.7%	7.3%	2.3%	2.8%
SSH TOP 3	2.1%	0.3%	0.4%	0.3%	0.1%	0.1%
SSH OTHERS	1.8%	1%	27.2%	21.2%	75.3%	69.0%

Table 3: Percentage of flows, packets and bytes for the attackers and the not suspicious hosts during the `ssh` anomalous time frame.

In order to give a visual representation of the network behavior during the anomalous and normal time frame, the scatter-plot in Figure 7 is presented. A time interval is characterized by a number of packets, bytes and flows. Let's suppose to assign to each measure an axis in a 3D space and plot each time interval as a point in this space. Figure 7 shows a representation of the anomalous and normal time frame. In the case of the anomalous time frame, also the projections on the planes are plotted. The graph permits to see that points belonging to the normal time frame tend to group together in a part of the space characterized by relatively small number of packets and bytes. Moreover, the time intervals in this group show a very low number of flows. On the contrary, the spatial disposition of the anomalous time frame describe a totally different behavior. Also in this case, the time intervals during the anomaly tend to be spatially close. This is an indication of the fact that they share common features. In addition, as emphasized by the projections, points in

this group present high values of the coordinates x (packets) and z (flows), while only few cases show a massive byte volume (y axis). Most importantly, the two groups are spatially distant to each other, confirming that anomalous and normal time intervals show clearly detectable differences.

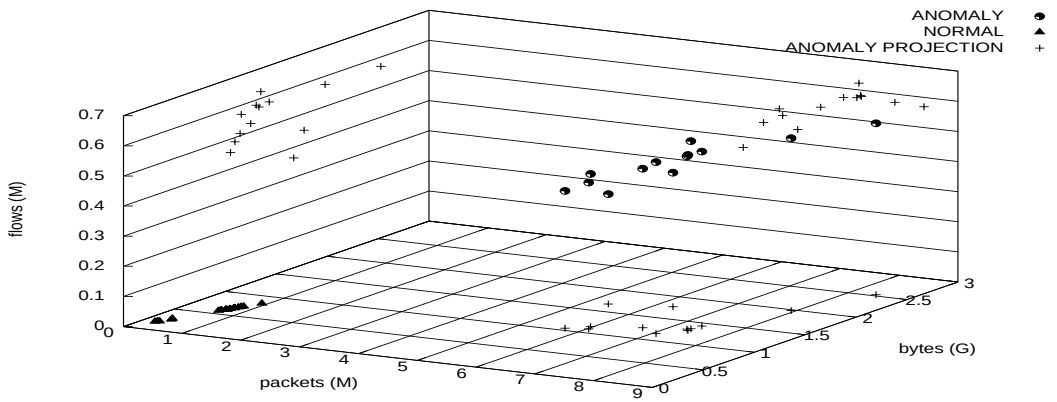

Figure 7: `ssh` anomalous and normal time frame space disposition (UT trace).

4.1.4 DNS traffic analysis

`Dns` is the second trace we analyze. Commonly, `dns` is responsible of the less than 1% of the incoming-outgoing data volume at the UT network.

In the previous section, `ssh` traffic seems to suggest that the flow frequency analysis can easily enlighten the presence of anomalies. Unfortunately, this hypothesis does not hold for `dns` traffic. As it possible to see in Figure 8, the number of flow per time interval is almost constant during the entire observation period and nothing would suggest the presence of an anomaly.

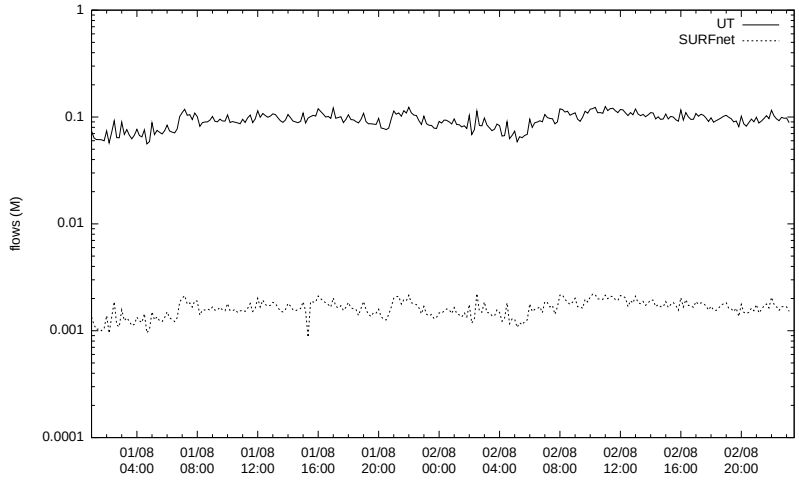

Figure 8: Flows time series, showing UT and SURFnet `dns` traffic (in logarithmic scale).

The situation appears to be different if we are interested not in the flow time series but in the packet and byte ones. Figures 9 and 10, indeed, show that in the time windows between 1:40 and 7:00 am on August 1st, the UT network saw a massive increase in the volume of `dns` traffic, both in packets and in bytes. In particular, both measures raise abruptly from few thousands to millions (between 10 to 28 millions in a time interval). The SURFnet trace shows the same behavior, even in presence of sampling.

The just described anomaly is unnoticeable if only the flow time series is taken into account. This observation is particularly relevant because it witness how flow frequency variation is not expressive enough to characterize anomalies. By definition, DNS traffic produces quite small UDP packets during the query process and it relies on TCP only in case of databases updates. Since the analysis of the protocol repartition during the anomaly shows that the 99.7% of the flows are UDP and they are responsible of the 99.9% of the bytes volume, we can exclude that the anomalies is caused by a database update. Under this consideration, we proceed for a more detailed analysis of the anomaly.

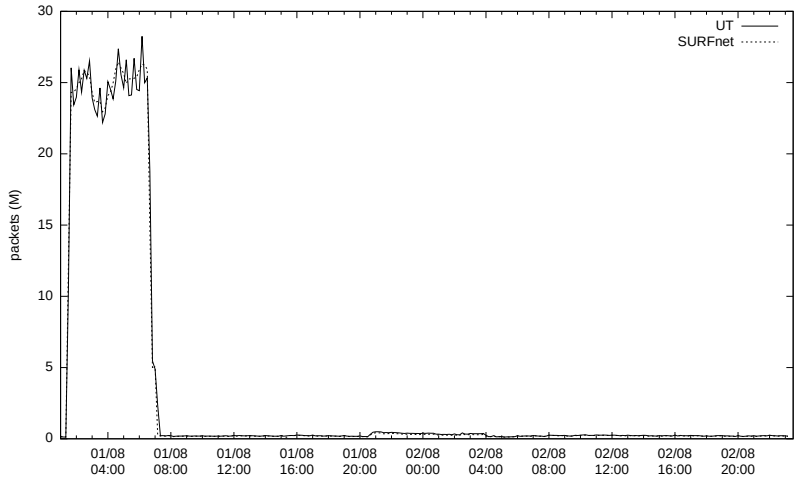

Figure 9: Packets time series, showing UT and SURFnet (estimated values) `dns` traffic.

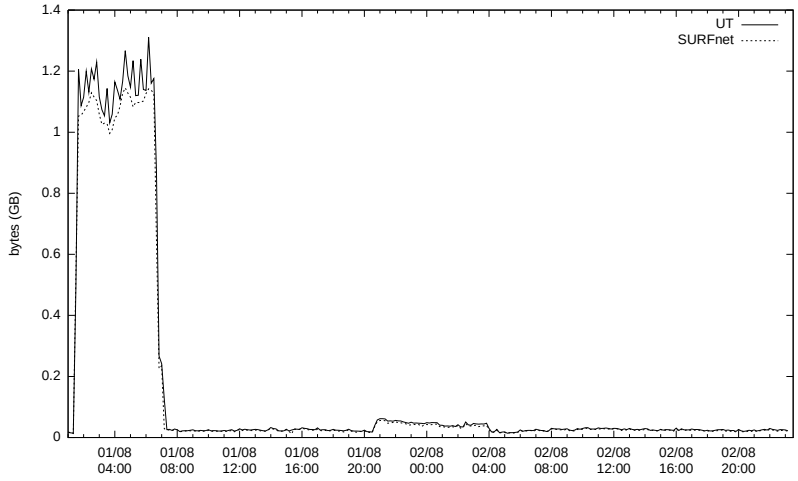

Figure 10: Bytes time series, showing UT and SURFnet (estimated values) `dns` traffic.

4.1.5 DNS normal vs anomalous traffic

As already for `ssh`, a not-anomalous interval has been chosen for sake of comparison. The *`dns` normal time frame* spans between 12:00 and 17:00 of August 1st. The large amount of bytes sent depicts a different scenario compared to the one presented in Section 4.1.2: the sharp variation in the byte and packets time series, together with the use of a large percentage of UDP packets suggests indeed the possibility of a DoS against a few number of destination hosts. The study of the anomalous time frame w.r.t the volume of byte sent clearly show the prevalence of three sources. Far away from the scenario of the `ssh` anomaly, the three sources are creating in average less that 300 flows each, being in this way responsible of only the 0.003% of the total UT flows. On the other side, each one of the major sources generates a packets volume almost 50 times bigger than all the other sources together. The proportion in the case of bytes is 20. SURFnet shows the same proportions. More generically, as it is possible to see in Table 4 the top senders host are responsible of more than 99% of the packets and the 98% of bytes in both UT and SURFnet traces. A deeper analysis of the traces shows that the three major sources share a single destination, towards which 33GB of data have been sent during the entire anomalous time frame (with packets of constantly exactly 46B in size). This configuration support the thesis that the destination host has been victim of a Distributed DoS targeting the `dns` service.

	Flow Percentage		*Packets Percentage*		*Bytes Percentage*	
	UT	*SURFnet*	*UT*	*SURFnet*	*UT*	*SURFnet*
DNS TOP 1	0.01%	0.14%	35.3%	35.3%	34.9%	34.8%
DNS TOP 2	0.01%	0.15%	32.6%	32.6%	32.3%	32.5%
DNS TOP 3	0.01%	0.14%	31.4%	31.4%	31%	31%
DNS OTHERS	99.97%	99.56%	0.7%	0.7%	1.8%	1.7%

Table 4: Percentage of flows, packets and bytes for the attackers and the not suspicious hosts during the `dns` anomalous time frame.

As previously in Section 4.1.2, a 3D representation of the anomalous and normal time frames is presented in Figure 11. Also in this case, the spatial disposition of the points in the two groups confirms the diversity between anomalous and normal time intervals. Points in the normal time frame show a relative variability in the number of flows, but almost no changes in the number of packets and bytes. On the contrary, the points in the anomalous group are characterized by large x and y coordinates (packets and bytes). Only two time intervals during the anomalies are distant from the majority: they show indeed a relatively small number of packets and bytes. Nevertheless, the xy-projection of the anomaly confirms that this points are in any case anomalies. All the points in the anomalous time frame, with no exception of the two just described, belong to the same straight line. This is a consequence of the fact that the attackers were flooding the victim with fixed size packets. As final observation, in the graph it is possible to see that the number of flows during the anomalous and normal time frames do not differ enough to detect the ongoing attack, confirming the observation about the flow time series.

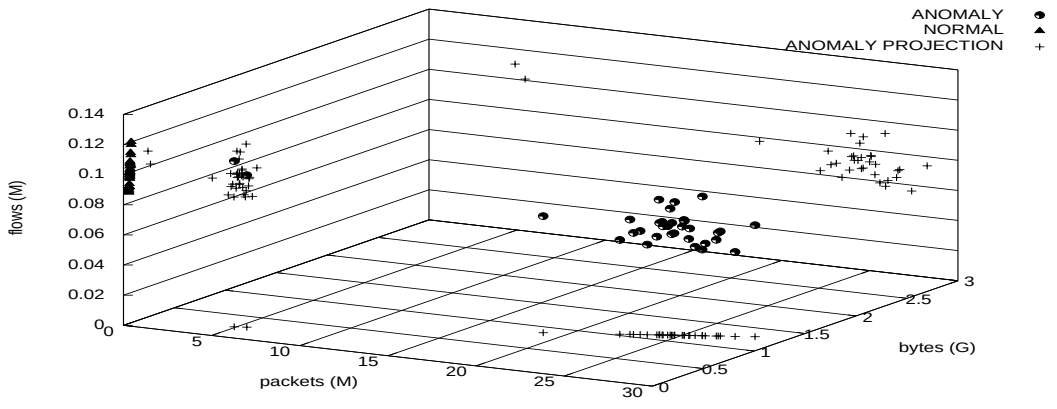

Figure 11: `dns` anomalous and normal time frame space disposition (UT trace).

4.1.6 Conclusions

An important contribution of this work is that our conclusions are based on extensive measurements on real, high speed networks, with line speeds of 10 Gbps. Our analysis confirm previous findings, that indicate that flows contain sufficient information to detect network intrusions. In particular, our study investigated whether flow, packet and byte time series are all needed to identify intrusions, or whether it is sufficient to consider only one or two of these metrics. Detailed analysis of two anomalies brought us to the conclusion that, to correctly identify suspicious traffic in general, all three metrics should be taken into consideration.

Our analysis also showed that, for certain classes of attacks, the choice to monitor only a single metric may still be sufficient. This is for example the case in our flow time series for `ssh` traffic. On the other hand, such choice entails the risk of hiding other attacks. This is, for example, the case for the `dns` DoS attack, which does not appear in the flow time series. Therefore it is important to observe flow, packet as well as byte time series variation, to properly characterize anomalies.

Our study showed that this conclusion also holds in the presence of sampling. Sections 4.1.2 and 4.1.4 showed that the sampled traces closely approximate the non-sampled traces, which means that accurate anomaly detection is even possible in case of sampling. This observation suggest that the development of scalable, but still accurate intrusion detection solutions is possible.

4.2 State-of-the-art of algorithms for detection of network anomalies in sampled data and their implementations in available monitoring software

A network behavior analysis system examines network traffic or statistics on network traffic to identify unusual traffic patterns, such as distributed denial of service (DDoS) attacks,

certain forms of malware (e.g. worms, backdoors), and policy violations (e.g., a client system providing network services to other systems).

Detection tools usually do not monitor the networks directly, but instead rely on network flow information provided by routers and other networking devices. Flow refers to a particular communication session occurring between hosts. The detection algorithms try to find abnormal behaviors from this flow-based statistics. The most popular standards for flow data formats are NetFlow [17] and sFlow [18]. Typical flow data particularly relevant to intrusion detection and prevention includes the following:

1. Source and destination IP addresses
2. Source and destination TCP or UDP ports or ICMP types and codes
3. Number of packets and number of bytes transmitted in the session
4. Timestamps for the start and end of the session.

4.2.1 Traffic anomalies detection methods

The following are details of the major existing detection technologies which are based on flow data analysis. Summary of different techniques is presented in Table 5.

Entropy-based detection Entropy-based anomaly detection [19] is based on the fact that suspicious traffic is more uniform and structured than normal network traffic and more random in some other aspects. Entropy tells how random a data-set is and can be measured by transforming the sequence of values into binary form and then compressing it. The size of the resulting compressed set corresponds to the entropy contents of the sequence.

In suspicious traffic hosts try to connect to many other hosts thus a source IP address is seen in many flows (while the target address is more random in flows) and since they are relatively few hosts this source address contains less entropy per address than normal network traffic. On the port level scans usually address a specific target port which means that this port is showing frequently in the traffic flow resulting in decreasing entropy contents of the destination port. For entropy calculations the aforementioned compression algorithms like bzip2, gzip or LZO are fed with data field of interest like source IP address found in flow records from the router. The compressed size of the resulting set is an entropy estimation.

The effect of significant entropy change is used to detect network anomalies especially large-scale attacks which are easily detected with this approach then slow or small-scale worms. When observing compression ratio i.e. the number of compressed bits per uncompressed bits of the suspicious traffic the change is clearly visible when some network attack occurs. The detection is based on the fact that during such attack source IP addresses become more uniform (small number of hosts generate more flows) and hence contain less entropy which decreases the compressibility. At the same time destination

Method	**Algorithm**	**Short characteristic**	**Behavior with sampled data**
Entropy based	Entropy estimation of compressed size of data from flows	Best for large-scale attacks	Well
Holt-Winters	Exponential smoothing of traffic data trends and seasonal effects	Detects anomalies which could not be easily identified by visual data (graphs) inspection. Could be used for planning.	Well
Wavelet analysis	Based on different frequency components, a deviation algorithm is presented to identify anomalies by setting a threshold for the signal composed	Best for high-frequency attacks	Well
Sketch based	A dimensionality reduction probabilistic technique building compact summaries	Massive data streams at ISP level. Analysis at low computation and memory costs in combination with HW and EWMA.	
CUSUM	Change point detection rises alarm when accumulated sum cresses threshold	Very useful for SYN DoS attacks	
ARIMA	Forecasting that captures the linear dependency of the future values on the past values	Gives low false positive and false negatives and presents high robustness	Well
Bayesian	Graphical model for a domain containing uncertainty	Computationally efficient and enables what is anomalous to be defined explicitly and in a principled fashion. Decreases false alarms.	

Table 5: Summary of network anomalies detection methods

IP address becomes more random which leads to increasing compressibility of this traffic characteristic.

Experiments show that when sampled flow data is considered this method doesn't change the basic behavior. Its impact is visible in the change of absolute value and amount of change but still possible to detect.

Holt-Winters forecasting and Exponentially Weighted Moving Average Holt-Winters [20] has been probably the most popular algorithm for anomaly detection. The detection is based on the assumption that observed time series in network traffic pattern can be decomposed into three components: a baseline, a linear trend and a seasonal effect with each component evolving over time. Network anomalies are indicated by a deviation from this statistical traffic pattern when forecasted and measured values from sampled data differ.

The model using Holt-Winters algorithm is usually combined with Exponentially Weighted Moving Average (EWMA) aka exponential smoothing [21] so it predicts the values of a time series one step into the future and measures the deviation between this value and the current data from flows resulting in an information whether observed traffic parameters are too different from predictions. Exponential smoothing is an algorithm used in this method predicts the next value of a time series given the current value and the current prediction (weighted average of all past observations in the time series). Smoothing places greater weight on the most recent data while forecasting with older observations decaying exponentially as the observations moves into the past.

With Holt-Winters algorithm suspicious change in monitored traffic pattern is detected when the current value of the time series e.g. amount of traffic in a specific flow falls out the confidence band or in slightly other approach the number of violations within a moving window of a fixed number of observations exceeds a specific threshold.

As the method is based on historical data it needs some initialization with past data. The advantage is that it removes diurnal and week start false positives also gives long term forecasts – can use for scheduling, bandwidth planning, etc. One disadvantage is that it requires a few adaptation parameters setting of the algorithm.

Wavelet analysis Statistics-based algorithms strongly rely on change detection for finding anomalies in traffic flows. Wavelet techniques [22], [23], [24] treat the flow based data as a generic signal (e.g. byte counts, packet counts) and provide means for isolating characteristics of the signal considering both time and frequency. Wavelet processing is made of two complementary steps: decomposition and reconstruction process. The first step extracts from the original flow-based signal a hierarchy of component signals, each of which maintains time as its independent variable. In particular it results in so called low-frequency representation which is a single dataset that extracts the general slow-varying trends of the original signal (like traffic patter change over several days) and in high-frequency representation which consists of short-term signal variations. The two representations are obtained through grouping corresponding wavelet coefficients into two intervals and signals are subsequently synthesized from them. The second step of wavelet

processing is reconstruction which performs the inverse of decomposition and recaptures the original signal.

Two general approaches of wavelet-based algorithms exist for traffic anomaly detection. One examines the various representations of the decomposition step and tries to get information about the original signal. Another approach tries to assemble the new signal from decomposition results through altering some of the values of the derived signals and the applying reconstruction process. This results in suppressing not important values from the anomaly detection point of view like day or night variation in the traffic. Therefore this part of wavelet analysis is tricky in order to determine which wavelets provide best resolution of signals in data flow to properly detect anomalies. Choosing a wavelet transform that fits best the given application is also a challenge.

Wavelet-based algorithms implement anomalies identification with the use of a deviation algorithm by setting a threshold for the signal composed from the wavelet coefficients at aforementioned different frequency levels.

The evaluation results show that some forms of DDoS attacks and port scans are detected within high-frequency representations due to their inherent anomalous alterations generated in traffic patterns. On the other hand low-frequency anomalies do not generate such patterns making it difficult to recognize. Thus these algorithms are particularly useful for characterizing data with sharp spikes and discontinuities. Experiments also show that NetFlow sampled data does not have much impact although it brings small amount of "jitter" in this type of signal.

Sketch-based change detection Both EWMA and wavelet algorithms are often used on top of data summaries built by sketch method. This is a dimensionality reduction probabilistic technique [25], [26], using projections or aggregation of high dimensional object into a smaller set of dimensions. It helps to build compact summaries of traffic measurements and enables computationally efficient methods for identifying anomalies in the traffic data.

Basically a sketch is a random aggregation of sampled IP flows characterized by 5-tuple (source and destination IP, source and destination port, protocol), time interval and number of packets in the flow. The use of hash function which maps tuple values to numbers 1..s and then aggregating all IP flows which hash to the same value results in s sketch flows. Usually sketch size is between 32 and 1024. When considering random aggregation of IP flows it gives an ability to find anomalies which are visible under some aggregations while not under other ones. This technique also leads to high detection rate as if a traffic anomaly happens only within a small set of aggregations but not within most of them it probably means false alarm.

Sketch-based change detection is then combined with forecasting methods which use forecasting techniques like Holt-Winters or Exponentially Weighted Moving Average to produce forecasts based on past observations. Both sketches can be then compared to determine significant changes indicating network anomalies.

ARIMA modeling Auto Regressive Integrated Moving Average (ARIMA) technique is also called Box-Jenkins methodology which originates from its authors' names. It's an

method of forecasting modeling used as a component in detecting network traffic anomalies. ARIMA [27], [28] captures the linear dependency of the future values on the past values. This algorithm is able to model a wide spectrum of time series behavior.

An ARIMA model includes three order parameters: the autoregressive parameter, the number of differencing passes and the moving average parameter. Many commonly used smoothing algorithms are special instances of ARIMA models. For example, the EWMA, is equivalent to ARIMA (corresponding parameters are 0,1,1), also Holt-Winters, is equivalent to ARIMA (corresponding parameters are 0,2,2).

Some studies shows that comparing to Holts-Winter approach, in ARIMA approach, the trends and seasonal effects are reserved whereas in HW these effects are not very visible. It also shows uniformly high fidelity (low false positive and false negatives) and high robustness (to routing changes and missing or corrupted data) [29].

CUSUM The CUSUM (cumulative sum) algorithm [30] is based in change point detection theory. Detection algorithm tries to detect changes in statistics of the traffic flow, based on measurements of the statistics in consecutive intervals of the same duration. In simple words an alarm is signaled when the accumulated volume of measurements that are above some traffic threshold, exceeds some aggregate volume threshold. CUSUM is sometimes joined with adaptive threshold algorithm (to calculate the threshold) where an alarm is signaled when the measurements exceed a threshold, for a number of consecutive intervals [31].

CUSUM is widely useful for detecting SYN flooding attacks where time series analysis is based on overall aggregation of TCP SYN packets. It uses the difference between the number of SYN packets in a time interval and the estimated number for the same interval as a Gaussian random variable. A SYN flooding attack is then detected using the cumulative sum based on the likelihood of the momentary value being caused by a change in the mean traffic rate. This method offers quite fixed false alarm rate and it was shown that CUSUM detects small but persistent changes with higher probability because little effects accumulate over time.

And again CUSUM (and it's variation Multi-channel CUSUM) is used in frameworks detecting network anomalies together with supporting techniques like variations of sketches which aggregate multiple data streams [32]. The Multi-channel CUSUM is statistical self learning algorithm for initializing required parameters (e.g. mean and variance) in order to build an initial normal profile, in an adaptive manner with various network load and traffic patterns. M-CUSUM belongs to anomaly based intrusion detection class, which detect a change in traffic parameters, through using the assumption that most anomalies induce a change in distributions of the above parameters.

It has to be mentioned that CUSUM raises only alarm when attack is detected and provides no information about victim server or attack type.

Bayesian techniques In order to identify whether an event should be classified as anomalous and to distinguish between anomalous but legitimate behavior the threshold-based decision algorithms are sometimes replaced with Bayesian techniques [33], [34]

which is used to model a domain containing uncertainty (like an intrusion detection system). Bayesian network decision process is used here to classify input events, which utilizes available additional information into detection decision. This approach is observed to bring a significant decrease in the number of false alarms.

In short in this type of algorithm there is a hypothesis that a given data belongs to a particular class (data flow is normal or anomalous). Then a probability for the hypothesis is calculated given the evidence gathered from so called information variables (measurable properties of input events, possibly some other model outputs). If at some stage there are additional training data, then each training example can incrementally increase/decrease the probability that this hypothesis is correct. More specifically, they are graphical models which use directed acyclic graphs to compactly represent joint probability distributions. Each random variable represents a node on a graph; directed edges indicate that a dependence exists between a pair of nodes.

Naive Bayes model is also used which is a simplified Bayesian algorithm.

4.2.2 Implementation in available monitoring software

The following are details of the popular applications of flow based anomaly detection algorithms.

We could not find any publicly available software tool for entropy based worm and anomaly detection in IP networks. Also the authors of the [19] were using for measurements their own prototype implementation in C language for Linux, what could confirm that there is a lack of publicly available above algorithm implementations.

NFSEN NFSen/nfdump [35] NetFlow engine is an example of software tool which experimentally implements a plug-in of the Holt-Winters forecasting algorithm for aberrant behavior detection using RRDtool [36]. Generally, NFSen (NetFlow Sensor) is a graphical web based front end for the nfdump NetFlow tools which collect and process NetFlow data on the command line. NFSen allows users to:

1. Display netflow data: Flows, Packets and Bytes using RRD (Round Robin Database).
2. Easily navigate through the netflow data.
3. Process the netflow data within the specified time span.
4. Create history as well as continuous profiles.
5. Set alerts, based on various conditions.
6. Write own plugins to process netflow data on a regular interval.

NFSen is distributed under the BSD license. The NFSen-hw version can run standalone or can cooperate with existing NfSen instance. It requires at least one week data before prediction and abnormal behavior detection, because one week period is selected by default.

DDosVax DDosVax [37] is a set of methods conceived by the Faculty of Electrical Engineering of the Eidgenossische Technische Hochschule Zuerich in order to quickly detect Denial of Service traffic. DDoSVax is short for "In search of a vaccine (Vax) against Distributed Denial of Service (DDoS) attacks", which was the initial motivation for starting the project.

DDoSVaX research is still ongoing at ETH Zrich, but most of the work under consideration for our purposes was carried out in 2003-2005. It is important to note that it was based on a different threat model. While our anomaly detection tries to identify potentially subtle deviations from the regular, DDoSVax intends to develop capabilities to detect huge work outbreaks in an as early stage as possible. Common to both goals is that detection needs to take place in semi-realtime during a phase when obvious signals for the presence of Findings of the Advanced Anomaly Detection Pilot.

RRDTool Round Robin Database (RRD) [38] is a system to store and display time-series data. It stores the data in a very compact way that will not expand over time, and it presents graphs by processing the data at different temporal resolutions. The RRDtool software is available as a standalone application or as a PERL library. It provides the functionality required by a real-time environment:

1. conversion of updates at irregular intervals into regular intervals via linear interpolation
2. conversion of counter data to rates
3. aggregation over user-defined time intervals via multiple aggregation functions

Aberrant behavior detection algorithm [20] was integrated within RRDtool [39], as opposed to implemented as a separate program and is decomposed into three pieces, each building on its predecessor:

1. An algorithm for predicting the values of a time series one time step into the future.
2. A measure of deviation between the predicted values and the observed values.
3. A mechanism to decide if and when an observed value or sequence of observed values is 'too deviant' from the predicted value(s).

The algorithm that adaptively predicts future observations in a time series is the Holt-Winters Time Series Forecasting Algorithm. Aberrant behavior (a potential 'failure') is reported whenever the number of times the observed value violates the confidence bands meets or exceeds a specified threshold within a specified temporal window (i.e. 5 violations during the past 45 minutes with a value observed every 5 minutes).

The round robin database (RRD) is organized into sequential sections, round robin archives (RRA). Within each RRA(array) is a section for each of the data sources stored in this RRD. Each RRA is defined by a consolidation function which maps primary data points (PDP) to consolidated data points (CDP). The aberrant behavior algorithm needs at least two

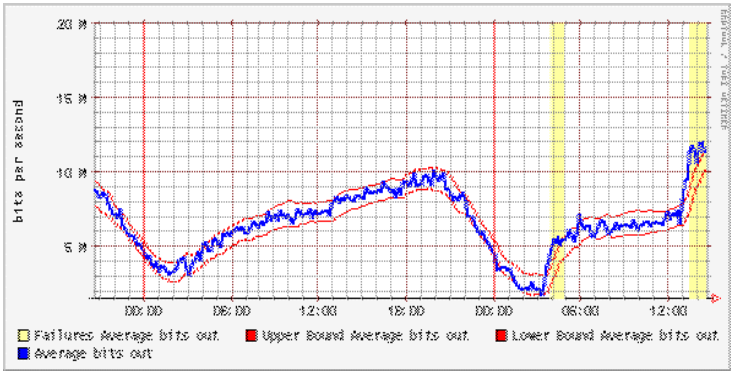

Figure 12: An example of daily graph generated on Wed, May 31, 2000 for the router interface

arrays, one to store the forecast values corresponding to each PDP, and a second to store the predicted deviation corresponding to each PDP. As implemented, the seasonal coefficients and deviations that are used to calculate the forecast and predicted deviations are stored in a second pair of RRAs. These arrays have length equal to the seasonal period and are updated once for each PDP. Failures are tracked by a fifth RRA, which determines violations and failures on each call to RRD update.

The aberrant behavior detection algorithm requires nothing unusual for the RRDtool update command, so the collection mechanism (i.e. Cricket invoking SNMP) will run normally. For example a network technician monitoring outgoing bandwidth at the router interface can view the following daily activity graph, including confidence bands and any failures.

The thin red lines in Figure 12 are the confidence bands and the yellow bars represent failures. The graph suggests that bandwidth on this outgoing link is increasing faster than expected by the model (past history). It is up to the network technician to decide if this represents aberrant behavior of interest or to view the time series for this router interface over a longer time period first. Generally, there are many benefits of aberrant behavior detection, but it does complicate the software implementation and its impact for runtime performance has to be also considered.

Bro Bro [40] is a stand-alone system for detecting network intruders in real-time by passively monitoring a network link over which the intruder's traffic transits. Bro is conceptually divided into an "event engine" that reduces a stream of (filtered) packets to a stream of higher-level network events, and an interpreter for a specialized language that is used to express a site's security policy. From the perspective of the rest of the system, just above the network itself is libpcap module, the packet capture library used by tcpdump. Using libpcap gains significant advantages: it isolates Bro from details of the network link technology (Ethernet, FDDI, SLIP, etc.), it greatly aids in porting Bro to different Unix variants and it means that Bro can also operate on tcpdump save files, making off-line development and analysis easy. Bro's "event engine" layer, first performs several integrity checks to assure that the packet headers are well-formed. If these checks fail, then Bro generates an event indicating the problem and discards the packet. If the checks succeed, then the

event engine looks up the connection state associated with the tuple of the two IP addresses and the two TCP or UDP port numbers, creating new state if none already exists. Bro maintains a tcpdump trace file associated with the traffic it sees. After the event engine has finished processing a packet, it then checks whether the processing generated any events (kept on a FIFO queue). If so, it processes each event until the queue is empty and it checks whether any timer events have expired, and if so processes them, too.

A key facet of Bro's design is the clear distinction between the generation of events versus what to do in response to the events. This structure reflects the separation between mechanism and policy. The "policy script interpreter" executes scripts (written in the specialized Bro language) which specify event handlers. For each event passed to the interpreter, it retrieves the (semi-)compiled code for the corresponding handler, binds the values of the events to the arguments of the handler, and interprets the code. This code can execute arbitrary Bro scripting commands, including generating new events, logging real-time notifications (syslog function), recording data to disk, or modifying internal state. Finally, along with separating mechanism from policy, Bro's emphasis on asynchronous events as the link between the event engine and the policy script interpreter buys a great deal in terms of extensibility. Adding new functionality to Bro generally consists of adding a new protocol analyzer to the event engine and then writing new event handlers for the events generated by the analyzer.

The Bro event engine and script interpreter are implemented in C++ as a single-threaded design but multi-threaded or distributed design may be adopted in the future.

Bro does the additional processing for the four applications it currently knows about: Finger, FTP, Portmapper, and Telnet. Admittedly these are just a small portion of the different Internet applications used in attacks, and Bro's effectiveness will benefit greatly as more areadded in the future

Bro has operated continuously since April 1996 as an integral part of its authors site's security system. It initially included only general TCP/IP analysis; as time permitted, the additional modules discussed above has been added. It runs under Digital Unix, FreeBSD, IRIX, SunOS, and Solaris operating systems. It generates about 40 MB of connection summaries each day, and an average of 20 real-time notifications, though this figure varies greatly. The system runs on the FDDI ring and generally operates without incurring any packet drops. Bro is publicly available in source-code [41].

Integrated Measurement Analysis Platform for Internet Traffic (IMAPIT) The IMAPIT [22] environment was developed to study a signal analysis of network traffic anomalies in IP flow and SNMP data collected at the University of Wisconsin's border router. IMAPIT contains a data management system which supports and integrates IP flow, SNMP and anomaly identification data. IMAPIT also includes a robust signal analysis utility which enables the network traffic data to be decomposed into its frequency components using a number of wavelet and framelet systems. IMAPIT has two significant components: a data archive and a signal analysis platform. The data archive uses RRDTOOL which provides a flexible database and front-end for our IP flow and SNMP data. The analysis platform is a framelet signal analysis and visualization system that enables a wide range of wavelet systems to be applied to signals. Signal manipulation and data preparation in IMAPIT analysis was performed using a modified version of the freely-available LastWave software

package. In addition to wavelet decomposition, authors implemented their deviation score method for exposing signal anomalies. Both flow and SNMP time-series data can be used as input to compute the deviation score of a signal.

Evaluation of deviation scores in IMAPIT as a means for automating anomaly detection shows it to be similarly effective to sophisticated time-series techniques such as Holt-Winters Forecasting. Both techniques have a small set of tunable parameters, and can perform poorly if configured incorrectly or perform well if their parameters are configured appropriately.

Other anomaly detection system prototypes based on wavelet analysis techniques have also been developed and implemented recently, such as Waveman by Huang et al. and NetViewer by Kim et al [24].

Internet Threat Detection System Using Bayesian Estimation A system that analyzes threat of the Internet using Bayesian estimation with transition of frequencies of IP packet access events to specified IP addresses has been developed by the authors of [42]. This system detects critical state of the Internet caused by increase of wide area network attacks by analyzing time-series transition of frequencies of port scanning events to some specified IP address. The authors defined critical states of Internet as highly probable states where WAN attacks are becoming active in the Internet and may cause loss of services or damages to a network site. Their method uses differences between frequency of port scans and their trend as observations.

An Internet threat detection system proposed in [42] is to send alert messages to system administrators hopefully before their network sites have been attacked, by observing changes of port scanning and detecting threat caused by increasing activities of network attacks. This system consists of a number of sensor systems which record TCP/UDP access events making use of Linux syslog in clscan compatible data format and sends them to the log server. The log server collects clscan data from several sensor systems and provides data to the analysis and detection system on demand. The analysis and detection system detects critical status of the Internet based on the method mentioned above and requests the notification system to send alerts. The notification system alerts system administrators, via the web or mobile phone browsers, about the critical status of the Internet.

Bayesian FlowMatrix FlowMatrix is Network Anomaly Detection and Network Behavioral Analysis (NBA) System, which in fully automatic mode constantly monitors network using NetFlow records from routers and other network devices in order to identify relevant anomalous security and network events. New release of FlowMatrix, supports Network Applications Behavior Analysis. This means user can define 3 groups of applications to monitor and FlowMatrix will automatically create a baseline for each of them, just like it does for network. When the baseline is crossed a security event is triggered. This allows to catch many attacks, exploits and other security violations on more granular level giving even better visibility to network and network applications environment. After initial learning period of (7-14 days) FlowMatrix builds multidimensional behavioral models of

network and network applications and later uses them to detect relevant anomalous security and network events. FlowMatrix provides short response time of 1 minute to know about anomaly right when it begins to happen.

The FlowMatrix receives NetFlow records from routers or other network devices configured to send NetFlow to FlowMatrix. It processes NetFlow records and after learning period builds detailed multidimensional behavioral models of network. Later it compares measured parameters from incoming NetFlow records to built models and identifies relevant anomalous events which significantly deviate from what is expected by the models and logs an event. To help user identify what each logged event means FlowMatrix performs (when possible) classification of each event to corresponding class of attack or network events.

FlowMatrix has following key features [43]:

1. Performs continuous 24x7 fully automatic behavioral analysis of network traffic to identify relevant anomaly security and network events.
2. Performs continuous 24x7 fully automatic behavioral analysis of your 3 groups of network applications traffic to identify relevant anomaly security and network events.
3. Classifies each reported anomaly event (when possible) as belonging to proper class of security or network events (DDoS, Scans, Alpha flows, network outages etc.).
4. Collects and presents relevant detailed information for each anomalous event so user can drill down to investigate each reported event to decide on proper set of actions.
5. Utilizes NetFlow records collected by network devices such as routers and switches. This eliminates need for additional expensive network probes and as result substantially lowers price for building network security monitoring solution. Currently only NetFlow versions 1, 5, 7 are supported, more being added;
6. Provides short response time — 1 minute.
7. Builds multidimensional behavioral models of your network and network applications in order to lower false positive rate.
8. Provides rule system for more interactive event identification
9. Moderate hardware requirements for small and medium size networks. As an example on Pentium 4 2.4 GHZ system with 2 GB of memory: FlowMatrix is able to handle 10000 flows per second up to 20000 flows per second is possible on more capable hardware.

Bayesian Network Classifiers in Weka Various Bayesian network classifier learning algorithms are implemented in Weka [44].

The summary of their main capabilities:

1. Structure learning of Bayesian networks using various hill climbing (K2, B, etc) and general purpose (simulated annealing, tabu search) algorithms.
2. Local score metrics implemented; Bayes, BDe, MDL, entropy, AIC.
3. Global score metrics implemented; leave one out cv, k-fold cv and cumulative cv.
4. Conditional independence based causal recovery algorithm available.
5. Parameter estimation using direct estimates and Bayesian model averaging.
6. GUI for easy inspection of Bayesian networks.
7. Part of Weka allowing systematic experiments to compare Bayes net performance with general purpose classifiers like C4.5, nearest neighbor, support vector, etc.
8. Source code available under GPL allows for integration in other open-source systems and makes it easy to extend.

More detailed description of this algorithms and their implementations can be found at [45].

4.3 Integration of Flow-based and Packet level-based Intrusion Detection

Network-based intrusion detection systems monitoring state-of-the-art high-volume network links demand for more computational resources than available from conventional computer hardware. Frequently, this problem is overcome by increasing the available resources with dedicated, specialized hardware. Instead of employing expensive infrastructure, efficiently decreasing the amount of data to be processed may lead to the same goal.

NetFlow enabled routers can be used as a source for aggregated connection data that can be leveraged for intrusion detection purposes. Using a model, possible use cases for combining conventional, packet-based and novel, flow-based intrusion detection are elaborated. With the Bro intrusion detection system as a central component, an architecture to combine both data sources is presented. In order to effectively reduce the amount of packets to be processed, a pre-filter employing flow data for the detection of peer-to-peer-based and IRC-based botnets is presented.

The performed evaluation, based on the comparison of the new, combined approach and the analysis of all packets, shows, depending on the implementation, a reduction of resource usage. Minor losses in the detection rate were observed. The prototypical implementation which was realized with existing software components shows synergy potentials for combining NetFlow and packet data for intrusion detection purposes. Further work is recommended, particularly to develop a flow-based pre-filter in a low-level language in order to further enhance resource efficiency.

4.3.1 IDS Evaluation

In this section we present evaluation techniques to establish comparability between different intrusion detection systems.

Literature on standardized evaluation methodologies for intrusion detection systems is very rare. There are no commonly applied methods for performing benchmarks so that different approaches and systems become *universally* comparable. The most notable effort in this area were the tests at the Air Force Research Laboratory, funded by DARPA in 1998 and 1999 [46]. The data published, however, unfortunately is no longer suitable for testing systems nowadays as attacks have changed and developed in the time passed. As this problem is out of scope with respect to this work, we have decided to go a pragmatic way to provide at least local comparability between the approaches (pure flow-based, pure packet-level based and combined).

[47] presents some of the commonly agreed characteristics and metrics as well as generic methodologies used to measure the performance of intrusion detection systems.

The measurable IDS characteristics presented are as follows:

- **Coverage**: This fundamental measurement determines what attacks can be detected by the IDS. While for misuse based systems this can be determined by counting the number of signatures for individual attacks when run under ideal conditions (e.g. in a lab environment), measuring anomaly based systems is a more difficult task. An open issue is the classification of attacks discussed in various papers on taxonomies and the granularity of what is counted to be an attack. The Common Vulnerabilities and Exposures (CVE) [48] is a commonly accepted database for known vulnerabilities but does not cover other security threats, not necessarily exploiting vulnerabilities but the presence of networked systems in general (e.g. worms).

- **Probability of false alarms (false positive rate)**: This measures the rate of false positives during a specified time frame, i.e. the number of alarms produced by the IDS for non-malicious traffic. This is a frequent problem with anomaly based systems when user behavior changes without malicious background. Signature based systems are also affected by false positives if weak signatures are used that do not only match traffic containing the attack, but also benign traffic.

- **Probability of detection (detection rate)**: The detection rate specifies the number of correctly detected attacks in relation to the total number of attacks in the traffic during a specified time frame. Included in this measurement are evasion techniques used by attackers for “hiding” their attacks from the IDS.

- **Ability to handle high bandwidth traffic**: This measurement shows the ability of how well and up to what extent the IDS is able to handle high-volume traffic. Two parameters are important in determining this value: transmitted bits per second (bandwidth) and transmitted packets per second.

- **Resistance to attacks directed at the IDS**: This measurement shows how resistant the IDS is against direct attacks to the IDS itself (i.e. not the monitored hosts). A possible scenario may be an attacker that tries to disrupt the correct functioning

of the IDS by sending large amounts of bogus traffic to the network, causing an overload situation exhausting all IDS resources so that it will not be able to detect the actual attacks inserted in between the bogus traffic.

- **Ability to correlate events**: This measures how well an IDS is able to correlate single attack events from different information sources (IDS, firewalls, routers, system logs, etc.) for the purpose of the detection of distributed attacks that may otherwise remain undetected.
- **Ability to detect never before seen attacks**: This measurement shows how well an IDS is able to detect attacks it is has not encountered before and that it does not contain explicitly programmed signatures for. This ability is seen mostly in IDSs based on the anomaly detection paradigm because “normal usage” is used as the basis of detection (i.e. anomalies are deviations from the defined normal behavior). It measures the detectability of attacks the system was not explicitly programmed to detect.
- **Ability to identify an attack**: This measures whether the IDS is able to explicitly name the detected attack or assign it to a category.
- **Ability to determine attack success**: This measures whether the IDS is able to differentiate between successful and unsuccessful attacks. It is important that the network operators are able to respond quickly to successful attacks and therefore also essential for the analysis of attack correlation.
- **Capacity verification for NIDS**: Because NIDS may operate on all layers of the OSI-model, it is important to measure the ability of a NIDS to capture, process and perform at the same level of accuracy given a certain network load as when it operates on a quiescent network (i.e. in a lab environment without benign background traffic). In order to have comparable capacity indicators, [49] proposed standardized metrics and processes for testing IDS.

  The capacity of an IDS is strongly linked to measuring resource consumption of the IDS processes. These so called “Fixed Resource Limitations” are based on the following resources:

  – Memory size
  – Memory bus bandwidth
  – Memory latency
  – Bus bandwidth for the network interfaces
  – Persistent storage bandwidth
  – CPU speed and bandwidth

  If one or more of these resources reach their limits, the system will not able to process all data and may therefore miss attacks.

  Apart from these network oriented metrics, on also wants to be able to monitor the resource consumption of the tested IDS in order to make valid assumptions on how much traffic can be monitored until its resources are exhausted, therefore measuring the capacity of the IDS. Therefore, it is necessary to measure the following values:

  - CPU consumption
  - Memory (RAM) usage
  - Number of CPU cycles per packet/flow
  - Memory usage per “state”
  - Other measurements

Background traffic is an important factor when testing IDSs. It is usually assumed that it is free of attacks, its sole purpose is to keep the IDS busy in order to make valid evaluation statements. Additionally, it is especially important for anomaly based IDS, as these systems must be trained (i.e. the statistical models) to a ground truth of which deviations are detected. An IDS that is only fed with the traffic containing the attacks may perform very well, as it can dedicate all of its resources to the detect those attacks which is not the case in the real world. IDS testing approaches can therefore be classified by the type of background traffic used [47]:

- **No background traffic**: This approach can be used to determine the detection rate of specific attacks in a lab environment. No statement can be made about false positives. Using this approach makes the implicit assumption that the IDS is able to detect an attack regardless of the background activity and performs equally well under all load scenarios.
- **Real background traffic**: In this very effective approach, attacks are injected into a stream of real background activity. Measurement of the detection rate is realistic because the background traffic is real and contains all common anomalies and subtleties. This is also the greatest drawback, as the background traffic itself may already contain attacks that falsify the result. It is also not possible to repeat the test under the very same circumstances because replaying recorded traffic often comes with timing issues. Other drawbacks are privacy concerns to distribute the complete test results and the laborious manual false positive rate testing.
- **Sanitized background traffic**: Sanitized traffic is cleansed of packet headers to overcome the political and privacy problems of real background traffic. The main difficulty is to remove all sensitive data while preserving realistic test data. Chances are that the injected attacks will not realistically interact with the sanitized background traffic.
- **Simulated background traffic**: As opposed to the previous scenarios, the complete traffic is generated artificially in a testbed environment. The main difficulty is the complexity and hence the costs of setting up such a testbed so that it provides realistic input data. Standard load testing traffic generators are often inappropriate because they fail to reproduce the variations of real traffic. The advantage of this approach is the absence of privacy problems and guaranteed attack free background traffic. The tests are also repeatable.

4.3.2 Flow-based vs. packet-based intrusion detection

In the following line-up in table 6, a comparison of key aspects of flow-based and packet-based IDS are shown.

In summary, if shortcomings of both intrusion detection approaches are analyzed, it is discovered that a tradeoff exists between (a) the flow-based IDS with only limited, aggregated data available for detecting intrusions and (b) the packet-based side, where for additional data a higher resource consumption is required. A potential combination of both approaches will be able to detect at least the same amount of attacks, while consuming less resources. If a system is running at maximum capacity, i.e. reaching one of the Fixed Resource Limitations, a possible additional leverage may arise from an increased detection rate caused by the reduced resource consumption.

Flow-based intrusion detection	Packet-based intrusion detection
NetFlow records contain aggregated data only up to network layer (OSI layer 3) and depending on the probe configuration also up to specific transport (OSI layer 4) information (TCP flags). With NetFlow v9 templates it is possible to include small parts of the payload in the records. The use cases for intrusion detection are therefore limited.	Packets naturally contain all headers up to application layer (OSI layer 7) and the complete payload. Packet-based IDS are therefore considered more flexible in the application of intrusion detection patterns.
Because of limited data available in flow-based IDS, defining accurate detection rules is not possible in all cases. This may result in a reduced alert confidence and higher number of false positives [50].	As the complete data is available, rules can be defined accurately on any part on the traffic resulting in less false positives and a higher alert confidence.
In order to process NetFlow data, one must have an additional component of a (hardware or software) probe. On the IDS side, the data must first be decoded. The complexity of flow-based IDS is therefore generally higher.	Simple packet-based IDS can be implemented comparatively easily without having to necessarily decode any protocol first.
The generation of flow records introduces a delay between the moment the first packet of a connection is established and the time when the record reaches the IDS. Depending on the configuration, records may only be emitted after the connection has been closed or has timed out.	In the prevalent case of mirrored switch ports, the complete data is available to the IDS in realtime.
Flows may have a insufficient time resolution for some intrusion detection purposes (e.g. how many bytes were transmitted at what time).	Packet-based IDS have the maximum possible time resolution available.

Encrypted payload does not affect the operability of flow-based IDS.	Signature matching is impossible for most cases of encrypted payload, degrading the detection performance of the IDS.
NetFlow data is aggregated data. Therefore there is less data to be processed in the IDS. Resource usage is generally lower.	In packet-based IDSs all traffic is forwarded to the IDS and then filtered. Therefore the resource usage is higher.
Pre-filtering, aggregating flows and therefore the handling of states is offloaded to the NetFlow probe device (e.g. Cisco Routers). The probe must therefore be dimensioned adequately in order to reliably process amount of traffic flowing without degrading network performance. Because in many networks NetFlow is already activated for other purposes (e.g. network monitoring), these processing efforts can be considered as sunk costs (i.e. resources that are consumed in either case).	Filtering, aggregation and state handling is maintained completely on the IDS machine, either by the libraries (pcap) or the IDS itself. Pre-filtering is often implemented by using complex and expensive hardware accelerated devices such as FPGAs.
Flow-based IDS have an overall lower amount of data to process, also in the analysis part, because parts of the processing is outsourced to the probe device. Resource consumption is therefore overall lower.	Packet-based IDS, in most cases when no hardware pre-filter is used, must process every packet received, possibly generating a huge workload on the system. Resource consumption is therefore generally higher.
There are less privacy issues with NetFlow data, as much of the potentially confidential contents of connections never leave the transmission network.	Packet-based IDS receive the full payload data of every packet that may contain confidential data.

Table 6: Flow-based / Packet-based IDS comparison

4.3.3 Combination Approach

In order to derive suitable use-cases and scenarios that leverage the use of a combined flow- and packet-based IDS on an analytical basis, a theoretical, graphical set model was constructed. The sets classify attack types in terms of detectability with the two approaches. It is further divided by “reliably detectable” and “suspicious”. Signatures in the “reliably detectable” class can show a high detection and low false positive rate with the respective approach. Signatures in the “suspicious” class can only be used to mark traffic as suspicious which needs further analysis in order to make a viable declaration of it being attack traffic.

In order to achieve a good detection performance, it is necessary that traffic marked as suspicious with one approach contains at least all traffic that indeed contains attacks. This means keeping the false negative rate low is crucial when designing signatures for

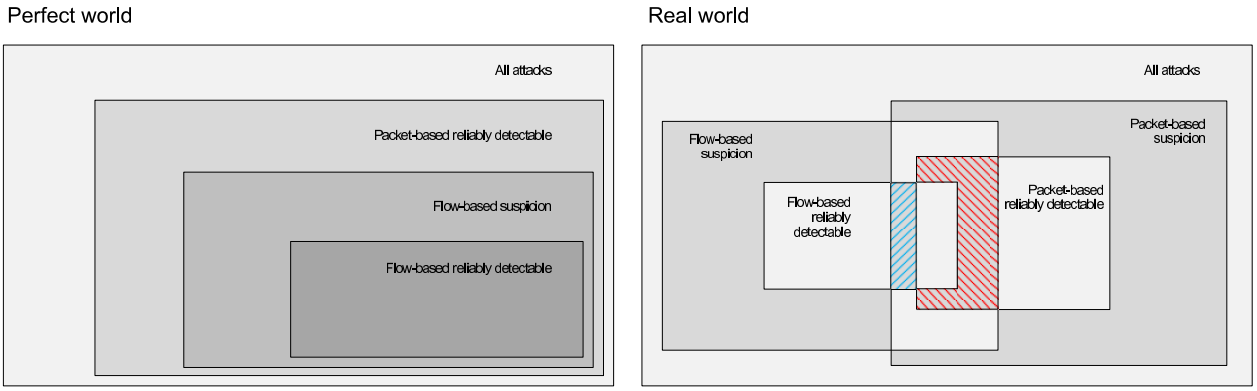

Figure 13: Set theory model of concepts

marking suspicious traffic while the false positive role is not of high importance. Marking suspicious traffic can therefore be seen as a pre-filtering operation, i.e. selecting traffic for further analysis. This is similar to the the pre-filtering approach using intelligent hardware taps using FPGAs. However, looking at the resource model, in contrast to FPGAs, we do not change the hardware layer but implement the (more flexible) pre-filtering on the analysis layer.

The graphical set model is drawn for two cases named the “perfect world” and the “real world” case. The perfect world, in this case, is a hypothetical construction of the situation when unlimited computing power and time is available to process network traffic. In this case, there are no resource constraints and the complete traffic can be analyzed with an unlimited amount of signatures and/or models to detect attacks. In the real world case, we introduce the resource constraints of limited computing power and time and show how this situation differs from the perfect world.

In both cases, the outmost set is the set containing all attacks, regardless of whether they can or cannot be detected with an IDS. Attacks that are only in contained in this set may be very new attacks for which no signatures exist yet. Naturally, the goal of intrusion detection efforts is to keep the number of attacks only belonging to this set as small as possible.

- **Perfect world**: In the perfect world model, the “All attacks” set contains the “packet-based reliably detectable” set, i.e. attacks that can be reliably detected with packet-based IDS according to the definition above. This set contains the “flow-based suspicious” set that itself contains the “flow-based reliably detectable” set. It is important to note that all sets are completely self-contained in others and that the “packet-based suspicious” set does not exist at all. The reason for this is simply because with unlimited computing power and time we can analyze all traffic with all signatures and/or models on packet-level. The flow-based sets are contained in the packet-based set because they provide no advantage in terms of information available to the IDS, as flow records are nothing else than summarized data of packets, i.e. all information available in the flow records is also available in the packets.
- **Real world**: While this statement also holds true in the real world, the primary problem introduced is that it is too expensive to analyze every packet of the monitored

traffic. Using summarized information from flow records can therefore be useful to decrease resource consumption, possibly at the price of a lower detection rate. In the real world scenario we have two suspicion sets for the two approaches that each contain their respective “reliably detectable” set. According to the above definitions, all attacks that can be reliably detected can also just be used to raise a suspicion while the opposite must not hold true. All sets partially overlap.

4.3.4 Proof of Concept

A proof of concept was chosen to be implemented in order to evaluate the elaborated setup with respect to feasibility and performance.

The released development version of Bro at the time of writing was 1.3.2 and does not yet support NetFlow. The newest work version was therefore checked out of the subversion repository. Furthermore, problems with respect to Broccoli queries were encountered with the publicly available Timemachine version 20061220. Therefore, the current project state from the subversion repository was requested from the authors.

Bros documentation was found to be incomplete and often out-of-date, thereby forcing its users to learn the implemented concepts from other, already existing policy scripts, reading through the Bro mailing list and finding correct solutions by trial-and-error. Commands and constructs used in policy scripts are partially explained in the Bro-IDS Wiki [18]. Debugging of policies is possible by launching Bro with the -d parameter that offers debugging options similar to the GNU debugger, such as setting breakpoints or displaying variable contents.

The proof of concept implementation was set up in the CSG Testbed of University of Zurich. In order for Bro to successfully compile in this environment, a number of libraries had be installed using the packet manager. As the purpose of the proof of concept was to show the feasibility of the hybrid intrusion detection mechanisms, a straightforward scenario was chosen: All flows representing web server access, which usually carries HTTP traffic, signified by a filter for a destination port value of port 80/TCP, were to be identified and fed to the Bro in packet mode.

4.3.5 Evaluation Sites

When it comes to evaluating the performance of intrusion detection systems, many problems arise. One of the first questions that needs to be addressed is the environment in which to test the IDS. Basically, there are two choices: a real environment (i.e. traffic from a production network) or a simulated environment (i.e. an experimental setup in a lab network). Both options have advantages and drawbacks. If one wants to make a comparative evaluation in order to compare how well an IDS performs in relation to another, logically it must be fed with the same input data. But if the same data has to be provided to more than one system, there are often infrastructural problems. The compared IDSs must be run simultaneously and receive the traffic from the network in real time. This is often difficult as it is connected with high hardware expenses. Instead, one could pull a traffic trace from the network tap and replay the traffic to each of the IDSs one after the other [51].

But then another problem arises, namely whether the monitored traffic actually carries the attacks that the IDSs should be tested against. Measuring false positives on real traffic is feasible by going manually through the logged payload and checking manually if the attack really existed. On the other hand, determining the false negative rate is a nearly unfeasible task, as the whole payload would have to be taken apart by hand looking for attack signs. This leads to using synthetic traffic traces. The problem with these are that they are factually not publicly available. There are only few resources: The traces from the DARPA project [46] back in 1999 are too old as they do not reflect the recent changes in Internet traffic (from HTTP to P2P) and new attacks in general (worms and botnets), the traces from the DEFCON 8 and 10 hacker conferences are more up-to-date but it is unclear what attacks they contain. Another more recent project is openpacket.org, not holding any usable traces yet. The main reason for the lack of availability of real traffic packet traces are privacy considerations. Such traces may contain private data that is not intended for the public. While replacing well-known fields in headers, for example IP addresses with such of the RFC 1918 private address space[1], is feasible, replacing all privacy relevant data in the almost arbitrary payload is an impossible task with such simple methods. For this reason, it is generally significantly easier to obtain archives of NetFlow records for evaluating pure NetFlow based IDSs, as these records contain only well-formed data.

Prepared packet traces of synthetic traffic are difficult to find. The only other alternative is to create own packet traces in a lab environment. This again, can pose significant effort to the tester, as attack paths of worms and botnets are distributed models and therefore need more than one host in order to perform realistically enough. A possible solution could be the use of multiple parallel running virtual machines on the same hardware to simulate a few hosts that the attack can propagate through. However, if we now feed those synthetically generated traces into the IDS, it will presumably perform rather well, considered that its resources are not exhausted by benign traffic that makes up most of the traffic in real networks. IDSs are therefore often evaluated by attaching them to a real network and defining that traffic to be attack free — a potentially misleading assumption — in order to put a realistic load on the system. The synthetic traffic carrying the to be detected attacks is then injected into the real traffic.

Because evaluation of intrusion detection systems is a very complex subject and is not the primary focus of this work, we decided to use a pragmatic approach: Evaluation is conducted using real, live traffic at several different network locations carrying high-speed traffic. Meaningful results are achieved by comparative measurements between different approaches. In the primary test, our combined NetFlow/Packet based approach is compared to using a pure, conventional packet based approach. Later, the different pre-filters are tested against each other by using a known positive. Finally, we quantify the false positive rate by manual inspection.

As the goal of all evaluation purposes was to detect infected hosts in the UniZH network, the evaluation was performed using only the internal network interface eth1. In order to characterize the link in terms of intrusion detection relevant performance parameters, hourly samples of 10 million packets were taken with tcpdump on 24.01.2008. The recorded samples were then processed with the tcpdstat software to determine throughput in megabit per second, packet frequency (number of packets per second) and the packet

[1] Because these addresses do not identify a globally unique host, privacy issues are avoided.

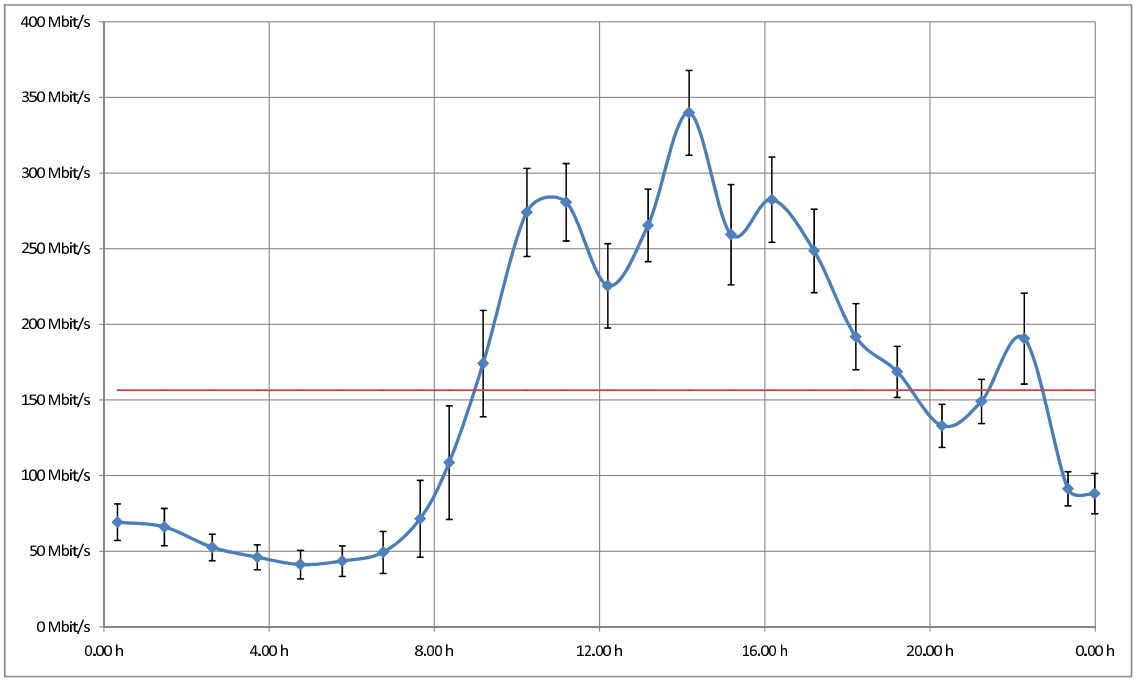

Figure 14: Throughput measurement at University of Zurich Internet Uplink, 24.01.2008

size distribution.

14 shows the hourly throughput measurement in megabits per second. The average (red line) lies at 156.5 Mbit/s, the peak at 14:09h at almost 340 Mbit/s. The standard deviation shown by the vertical indicators at the measurement points, averages at 20.75 Mbit/s. Deviations during a single measurement can occur because the sample measurement is terminated by the number of packets, hence throughput can change during the measurement itself.

15 shows the packets frequency measurement in packets per second. The curve largely correlates with the throughput measurement. The average lies at 27670 packets/s (red line), the peak at 14:09h at 57460 packets/s.

16 shows the packet size distribution in bytes of the combined sample measurements throughout the day. 43% of the packets are smaller than 128 bytes, 45% are larger than 1023 bytes, 12% have a size between 128 and 1023 bytes. The MTU for most hosts is observed at 1434 bytes with 29% of all packets having this size and only 0.58% with a size larger.

Figure 14 shows the hourly throughput measurement in megabits per second. The average (red line) lies at 156.5 Mbit/s, the peak at 14:09h at almost 340 Mbit/s. The standard deviation shown by the vertical indicators at the measurement points, averages at 20.75 Mbit/s. Deviations during a single measurement can occur because the sample measurement is terminated by the number of packets, hence throughput can change during the measurement itself. Figure 15 shows the packets frequency measurement in packets per second. The curve largely correlates with the throughput measurement. The average lies at 27670 packets/s (red line), the peak at 14:09h at 57460 packets/s.

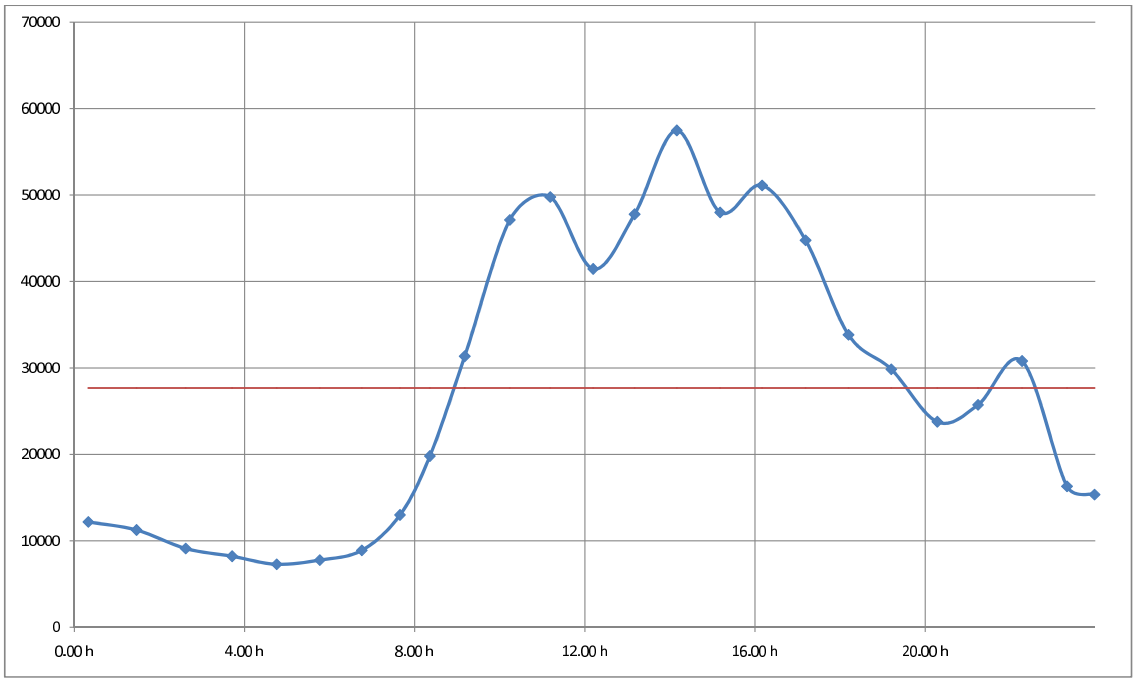

Figure 15: Packet frequency measurement at University of Zurich Internet Uplink, 24.01.2008

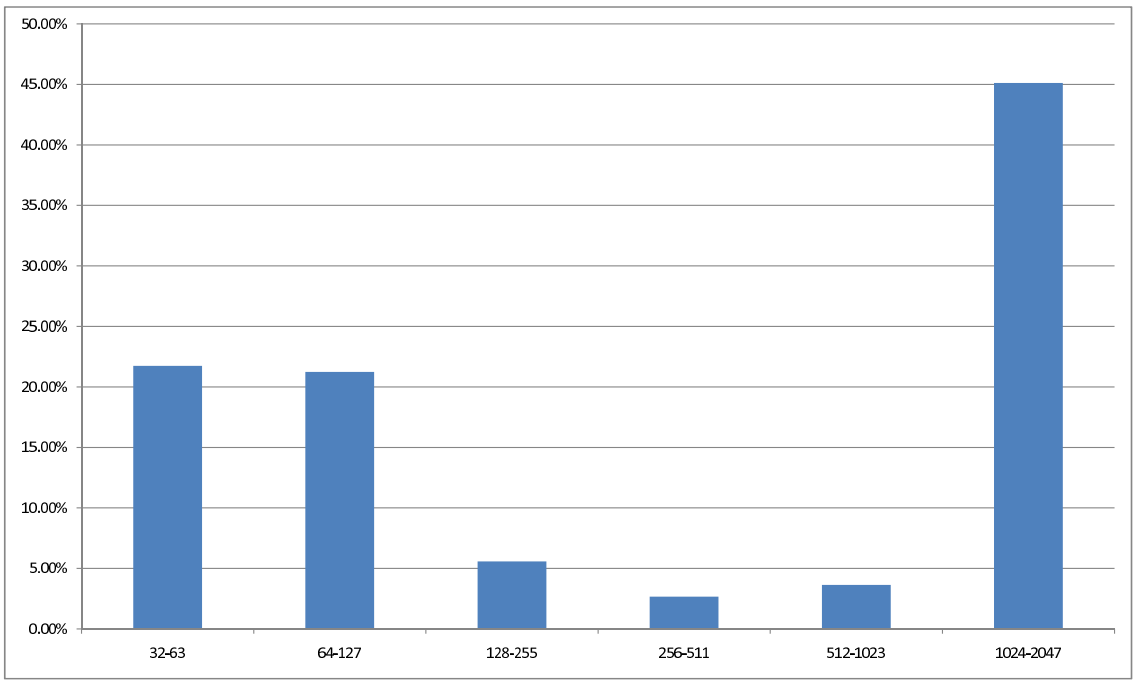

Figure 16: Packet size distribution measurement (in bytes) at University of Zurich Internet Uplink, 24.01.2008

4.3.6 Summary and Conclusions

In this section a method to combine flow-based and packet-based intrusion detection mechanisms was described. This approach – fully described in all levels of details in the thesis of F. Hensel termed ”Flow-based and Packet level-based Intrusion Detection as Complementary Concepts” – decreases resource usage, improves scalability, and enhances alert confidence.

Essentially, it was stated that there exists a tradeoff between limited data availability of flow-based approaches and higher resource consumption of packet-based approaches. The combination model to combine both approaches leverages their advantages and overcomes their shortcomings. Using a proof of concept, the feasibility of the combination approach was tested.

The evaluation was run in an implementation with real traffic, at two high-volume sites, presented above in terms of the UniZH-network. Due to technical issues, the outcome can only state that a combination of flow- and packet-based intrusion detection has the potential to optimize resource consumption for intrusion detection in high-volume environments. As for all intrusion detection systems, careful adjustment to the networks characteristics is necessary. Unfortunately, because of limited resources, the work was not able to do conclude with an overall conclusive statement. However, the trend to combine and gain at the same time has become visible.

5 Conclusions

We report in this deliverable on the activities in the work-package 7 performed in the time frame April 2008 to December 2008. These activities have been driven by two main topics, that emerged from an open call for proposals. The first topic is concerned with a scalable distribution system for biometric templates in large scale deployment projects. The objective of the BioScale activity was the development of a scalable management approach for biometric devices. Several tasks have been addressed and prototypes have been implemented. The second topic is addressing the scalable network (security) monitoring. In the SNAID activity (Scalability of Network Analysis and Intrusion Detection), we addressed the issues on how traffic anomalies can be detected in large networks on flow level and how packet-based and flow-based intrusion detection can be combined. In addition, we discussed the state-of-the-art of algorithms for detection of network anomalies in sampled data and their implementations in available monitoring software. The reported content has been also the subject of publications at scientific conferences.

6 Acknowledgement

This deliverable was made possible due to the large and open help of the WP7 Partners of the EMANICS NoE. Many thanks to all of them.

References

[1] Internet2 NetFlow: Weekly Reports. netflow.internet2.edu/weekly, April 2008.

[2] P. Haag. Nfsen: Netflow sensor. nfsen.sourceforge.net, April 2008.

[3] D. Plonka. Flowscan. www.caida.org/tools/utilities/flowscan/, April 2008.

[4] B. Claise. Cisco Systems NetFlow Services Export Version 9. Request for Comments: 3954, October 2004. IETF.

[5] Cisco IOS NetFlow Configuration Guide. www.cisco.com, April 2008.

[6] IP Flow Information Export Working Group. www.ietf.org/html.charters/ipfix-charter.html, April 2008.

[7] G. He and J. C. Hou. An in-depth, analytical study of sampling techniques for self-similar internet traffic. In *ICDCS '05: Proc. of the 25th IEEE International Conference on Distributed Computing Systems*, pages 404–413. IEEE Computer Society, 2005.

[8] E. Izkue and E. Magaña. Sampling time-dependent parameters in high-speed network monitoring. In *PM2HW2N '06: Proc. of the ACM international workshop on Performance monitoring, measurement, and evaluation of heterogeneous wireless and wired networks*, pages 13–17. ACM, 2006.

[9] Cisco IOS NetFlow. www.cisco.com/go/netflow, April 2008.

[10] sFlow. www.sflow.org, April 2008.

[11] A. Lakhina, M. Crovella, and C. Diot. Characterization of network-wide anomalies in traffic flows. In *IMC '04: Proc. of the 4th ACM SIGCOMM conference on Internet measurement*, pages 201–206. ACM, 2004.

[12] A. Lakhina, M. Crovella, and C. Diot. Diagnosing network-wide traffic anomalies. In *SIGCOMM '04: Proc. of the Conference on Applications, technologies, architectures, and protocols for computer comm.*, pages 219–230. ACM, 2004.

[13] T. Dubendorfer and B. Plattner. Host behaviour based early detection of worm outbreaks in internet backbones. In *WETICE '05: Proc. of the 14th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprise*, pages 166–171, Washington, DC, USA, 2005. IEEE Computer Society.

[14] Y. Gao, Z. Li, and Y. Chen. A dos resilient flow-level intrusion detection approach for high-speed networks. *ICDCS 2006: 26th IEEE International Conference on Distributed Computing Systems*, pages 39–39, 2006.

[15] G.Munz and G.Carle. Real-time analysis of flow data for network attack detection. *IM '07: 10th IFIP/IEEE International Symposium on Integrated Network Management, 2007.*, pages 100–108, 2007.

[16] SURFnet. www.surfnet.nl, April 2008.

[17] Netflow information. www.cisco.com/go/netflow.

[18] Inmon corporation's sflow: A method for monitoring traffic in switched and routed networks rfc3176. www.isi.edu/in-notes/rfc3176.txt.

[19] A. Wagner and B. Plattner. Entropy based worm and anomaly detection in fast ip networks. In *14th IEEE International Workshops on Enabling Technologies: Infrastructures for Collaborative Enterprises (WET ICE 2005), Linkping, Sweden*, June 2005.

[20] J.D. Brutlag. Aberrant behavior detection in time series for network monitoring. In *Proceedings of the 14th Systems Administration Conference (LISA 2000), New Orleans, Louisiana, USA*, December 2000.

[21] P.J. Brockwell and R. Davis. *Introduction to Time Series and Forecasting*. Springer, New York, 1996.

[22] P. Barford, J. Kline, D. Plonka, and A. Ron. A signal analysis of network traffic anomalies. In *Proceedings of the 2nd ACM SIGCOMM Workshop on Internet measurement, pp. 71-82, Marseille, France*, 2002.

[23] S. Kim, A. L.N. Reddy, and M. Vannucci. *Detecting Traffic Anomalies Using Discrete Wavelet Transform*. LNCS, vol. 3090/2004, pp. 951-961, Springer Berlin, 2004.

[24] W. Lu and A.A. Ghorbani. Network anomaly detection based on wavelet analysis. *EURASIP Journal on Advances in Signal Processing*, 2009.

[25] B. Krishnamurthy, S. Sen, Y. Zhang, and Y. Chen. Sketch-based change detection: Methods, evaluation and applications. In *ACM SIGCOMM Internet measurement Conference, Miami, USA*, October 2003.

[26] X. Li, F. Bian, M. Crovella, C. Diot, R. Govindan, G. Iannaccone, and A. Lakhina. Detection and identification of network anomalies using sketch subspaces. In *IMC'06, Riode de Janeiro, Brazil*, October 2006.

[27] G. E. P. Box and G. M. Jenking. Time series analysis, forecasting and control. In *holden-Day*, 1976.

[28] B. Krishnamurthy, S. Sen, Y. Zhang, and Y. Chen. Sketch-based change detection: Methods, evaluation, and applications. In *Proceedings Internet Measurement Conference (IMC'03), pp. 234-247, ACM Press.*, 2003.

[29] Z. Ge, A. Greenberg, M. Roughan, and Y. Zhang. Network anomography. In *Proceedings of Internet Measurement Conference (IMC '05), pp. 317–330*, 2005.

[30] H. Wang, D. Zhang, and K. G. Shin. Detecting syn flooding attacks. In *Proceedings of IEEE INFOCOM*, 2002.

[31] IST-SCAMPI project. D2.3 enhanced scampi implementation and applications. www.ist-scampi.org/publications/deliverables/D2.3.pdf.

[32] O. Salem, S. Vaton, and A. Gravey. A novel approach for anomaly detection over high-speed networks. In *Proceedings of European Conference on Computer Network Defense (EC2ND'07), Heraklion, Greece*, October 2007.

[33] M. Panda and M. Patra. Network intrusion detection using nave bayes. *IJCSNS International Journal of Computer Science and Network Security, vol. 7 no. 12 , pp. 258-263*, 2007.

[34] C. Kruegel, D. Mutz, W. Robertson, and F. Valeur. Bayesian event classification for intrusion detection. In *Proceedings of 19th Annual Computer Security Applications Conference, Las Vegas*, 2003.

[35] Nfsen homepage. nfsen.sourceforge.net.

[36] J. Mohcsi and G. Kiss. Anomaly detection for nfsen/nfdump netflow engine with holt-winters algorithm. bakacsin.ki.iif.hu/˜kissg/project/nfsen-hw/JRA2-meeting-at-Espoo_slides.pdf.

[37] Ddosvax homepage. www.tik.ee.ethz.ch/˜ddosvax/project/.

[38] Rrdtool homepage. oss.oetiker.ch/rrdtool/.

[39] Rrdtool implementation of aberrant behavior detection homepage. cricket.sourceforge.net/aberrant/rrd_hw.htm.

[40] V. Paxson. Bro: A system for detecting network intruders in real-time. *Computer Networks, vol. 31, no. 23-24, pp. 2435–2463*, 1999.

[41] Bro intrusion detection system homepage. bro-ids.org.

[42] M. Ishiguro, H. Suzuki, I. Murase, and H. Ohno. Internet threat detection system using bayesian estimation. In *FIRST 16th Annual Conference, Budapest*, June 2004.

[43] Bayesian flowmatrix. www.akmalabs.com/news.php.

[44] I.H. Witten and E. Frank. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, 2nd Edition, San Francisco, 2005.

[45] Remco R. Bouckaert. Bayesian network classifiers in weka for version 3-5-7. www.cs.waikato.ac.nz/˜remco/weka_bn/.

[46] R. Lippmann, J.W. Haines, D.J. Fried, J. Korba, and K. Das. The 1999 DARPA off-line intrusion detection evaluation. *Computer Networks*, 34(4):579–595, 2000.

[47] P. Mell, V. Hu, R. Lippmann, et al. An overview of issues in testing intrusion detection systems. *National Institute of Standards and Technology ITL, July*, 2003.

[48] Common Vulnerabilities and Exposures (CVE). `http://cve.mitre.org/`.

[49] M. Hall and K. Wiley. Capacity Verification for High Speed Network Intrusion Detection Systems. *Recent Advances in Intrusion Detection: 5th Internatonal Symposium, RAID 2002, Zurich, Switzerland, October 16-18, 2002: Proceedings*, 2002.

[50] Daniela Brauckhoff, Martin May, Bernhard Plattner. Flow-Level Anomaly Detection - Blessing or Curse? *IEEE INFOCOM 2007, Student Workshop, Anchorage, Alaska, USA*, 2007.

[51] A. Turner. Tcpreplay: Pcap editing and replay tools. *online, http://tcpreplay.sourceforge.net*, 2006.