An Overview of IP Flow-Based Intrusion Detection

Anna Sperotto, Gregor Schaffrath, Ramin Sadre, Cristian Morariu, Aiko Pras and Burkhard Stiller

***Abstract*—Intrusion detection is an important area of research. Traditionally, the approach taken to find attacks is to inspect the contents of every packet. However, packet inspection cannot easily be performed at high-speeds. Therefore, researchers and operators started investigating alternative approaches, such as flow-based intrusion detection. In that approach the flow of data through the network is analyzed, instead of the contents of each individual packet.**

The goal of this paper is to provide a survey of current research in the area of flow-based intrusion detection. The survey starts with a motivation why flow-based intrusion detection is needed. The concept of flows is explained, and relevant standards are identified. The paper provides a classification of attacks and defense techniques and shows how flow-based techniques can be used to detect scans, worms, Botnets and Denial of Service (DoS) attacks.

***Index Terms*—Network flows, intrusion detection, attacks, DoS, scan, worms, Botnets.**

I. Introduction

NOWADAYS hackers are continuously attacking networked systems; in fact, it would be interesting to investigate if there are still Internet users who have not been victim of an attack yet. Considering the damage caused by the attacks (billions of U.S. dollars) [1], it is important to detect attacks as soon as possible, and take, if feasible, appropriate actions to stop them. This task is particularly challenging due to the diversity in form (information gathering, password stealing, viruses, Trojan horses, Denial of Service (DoS)...) attacks exhibit.

For the detection of network attacks, special systems have been developed; these systems are called Network Intrusion Detection Systems (NIDS). In an attempt to find known attacks or unusual behavior, these systems traditionally inspect the contents (payload) of every packet [2], [3]. The problem of packet inspection, however, is that it is hard, or even impossible, to perform it at the speed of multiple Gigabits per second (Gbps) [4], [5]. For high-speed lines, it is therefore important to investigate alternatives to packet inspection. One option that currently attracts the attention of researchers and operators is *flow-based* intrusion detection. With such approach, the communication patterns within the network are analyzed, instead of the contents of individual packets. Research in this field is still relatively in its beginning, even if initial ideas to abstract from communication details and analyze *source/destination* pairs instead can already be found in papers published in the early 1990s (see for example Heberlein *et al.* [6] and Staniford-Chen *et al.* [7]). Nowadays special measurement systems are able to provide, for every pair of IP addresses and port numbers, aggregated information, such as the time data exchange has started, the time it has stopped, the amount of transferred bytes and the number of sent packets. These systems export this information in the form of Netflow [8], [9] or IPFIX [10] records to systems that analyze them. These analysis systems can then be used to detect intrusions.

In our opinion, flow-based detection can be seen as a complement of packet inspection, and should not be seen as a replacement. Both approaches can be combined into a two-stage detection process. At the first stage, flow-based approaches can be used to detect certain attacks. At the second stage, packet inspection can be used to additionally protect critical servers or selected systems, for which the first stage has discovered suspicious activities.

This paper provides a survey of current research in the area of flow-based intrusion detection. This means that we consider only contributions in network intrusion detection that make explicit use of network flows as their main input. To limit our scope, we will not consider payload-based methods; readers interested in such methods can refer to existing literature [11], [12], [13], [14], [15]. Since we concentrate on network flows, our paper does not consider host-based intrusion detection systems. Last, since details of commercial products are hard to obtain, these have also been left out of this survey.

The paper is organized as follows: Section II describes the motivations that have encouraged researchers to start this research. Section III explains the concept and ideas behind flows, as well as the network infrastructure needed for flow monitoring and analysis, such as intrusion detection. Section IV provides a classification of current attack techniques, whereas Section V provides a classification of defense techniques. Section VI discusses how flow information can be used to detect intrusions; in this section, the focus is on thwarting Denial of Service (DoS), scans, worms and Botnets. Finally, Section VII presents some conclusions and discusses the strengths and weaknesses of current flow-based approaches.

II. Motivation

The Internet is a complex system in constant evolution. Nevertheless, it is possible to make some observations with respect to security.

A first observation is that the number of attacks continues to grow. The Cert Coordination Center [16], one of the

Manuscript received 6 August 2008; revised 2 March 2009 and 2 June 2009.

Anna Sperotto, Ramin Sadre, and Aiko Pras are with the University of Twente, Centre for Telematics and Information Technology, Faculty of Electrical Engineering, Mathematics and Computer Science, P.O. Box 217, 7500 AE Enschede, The Netherlands, (e-mail: {a.sperotto,r.sadre,a.pras}@utwente.nl).

Chistian Morariu and Burkhard Stiller are with the Department of Informatics, University of Zurich, CH-8050, Zurich, Switzerland (e-mail: {morariu,stiller}@ifi.uzh.ch).

Gregor Schaffrath is with An-Institut Deutsche Telekom Laboratories, Technical University Berlin, 10587 Berlin, Germany (e-mail: grsch@net.t-labs.tu-berlin.de).

Digital Object Identifier 10.1109/SURV.2010.032210.00054

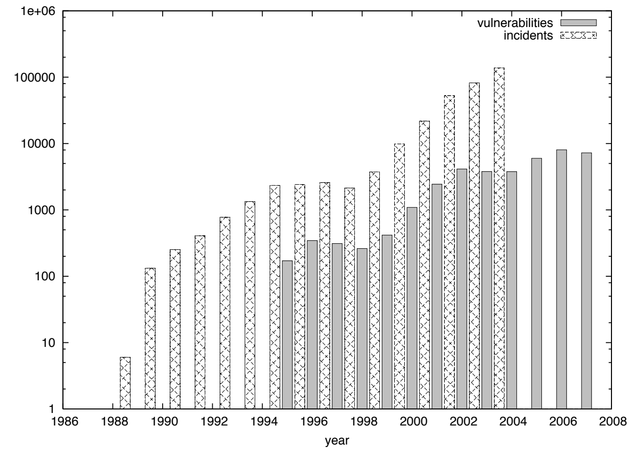


Fig. 1. Trends in incidents and vulnerabilities (logarithmic scale).

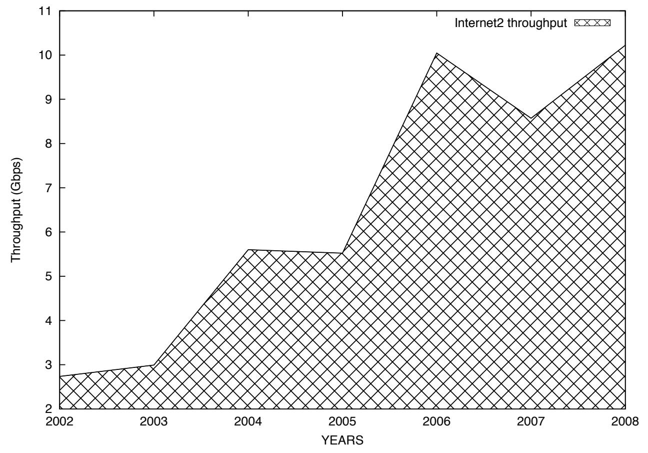


Fig. 2. Network throughput (Gbps) for the network Abilene[17].

most well-known risks, security threats and incidents response centers, offers summaries of the yearly security situation of the Internet. The Cert/CC maintains a database of vulnerabilities, with the aim to categorize them according to their severity level and damaging impact on the systems. Vendors, system administrators and users are encouraged to submit vulnerabilities. In a similar way, in the past, Cert/CC asked the Internet community for collaboration in order to report the incidents the users were subject to. Cert/CC defines an incident as *the act of violating an explicit or implied security policy* [16]. This definition, according to the Cert/CC, covers attempts to gain access to (information on) a system, Denial of Service, disruptions, unauthorized uses and changes to hardware and software.

Since 1995, Cert/CC published each year the number of catalogued vulnerabilities. In fact, the reporting of incidents started already in 1988, but ended in 2003. The reason to stop can easily be understood from Figure 1: the growth of reported incidents is nearly exponential, while the number of catalogued vulnerabilities shows a slower growth factor. The Cert/CC itself [16] gives the following explanation:

> "Given the widespread use of automated attack tools, attacks against Internet-connected systems have become so commonplace that counts of the number of incidents reported provide little information with regard to assessing the scope and impact of attacks. Therefore, we stopped providing this statistic at the end of 2003."

A second observation is that Internet traffic, as well as line speed, continues to grow. Nowadays an access speed of 1-10Gbps is not unusual. A university network, for example, reaches traffic averages in the order of hundreds of Mbps, with high activity peaks in the order of Gbps. On backbone networks, the throughput will even be higher. Internet2 [17], for example, publishes weekly reports of the Abilene traffic. Figure 2 shows the growths in the period 2002-2008.

It is clear that Network Intrusion Detection Systems should be able to handle the growing number of attacks, the growth in Internet traffic as well as the increase in line speed. Researchers assess the current, payload-based, NIDS processing capability to lie between 100Mbps and 200Mbps [4], [5]. Well known systems like Snort [2] and Bro [3], exhibit high resource consumption when confronted with the overwhelming amount of data found in today's high-speed networks [18]. In addition, the spread of encrypted protocols poses a new challenge to payload-based systems. An example is the work of Taleb *et al.* [19], [20], where the authors propose an intrusion detection systems based on per-packet inspection that rely only on header information in order to identify misuses in encrypted protocols.

Given these problems, flow based approaches seem to be a promising candidate for Intrusion Detection research.

Flows are created by specialized accounting modules usually placed in network routers. The same modules are responsible of exporting the flows to external collectors (see Section III). Flow-based Intrusion Detection Systems will analyze these flows and detect attacks. Compared to traditional NIDS, flow-based NIDS have to handle considerable lower amount of data. For example, in the case of the University of Twente network, we calculated that the ratio between packets exported by NetFlow (containing the flow records) and the packets on the network is in average equal to 0.1%. Moreover, considering the network load measured in bytes, the overhead due to Netflow is in average 0.2%. Flow based intrusion detection is therefore the logical choice for high-speed networks. However, there might exist situations in which the benefit of using flows is not so pronounced. The worst case scenario would be when a flow is created for each packet passing through the monitoring point, as a consequence of a distributed DoS attack (DDoS), for example. In this case, the number of flows would increase dramatically and extra load would be put on the monitoring and analysis systems. To mitigate this problem, or, in general, to improve the performance of routers and monitoring stations, sampling techniques or flow aggregations [21] can be applied.

Sometimes it is argued that flows do not carry enough information, compared to payload inspection, for being useful for intrusion detection. The answer to this question highly depends on the user's goals. Flows, which represent by nature aggregated information, do not carry any payload. They, therefore, do not provide the detection precision of packet-based inspection, which allows for example *pattern*

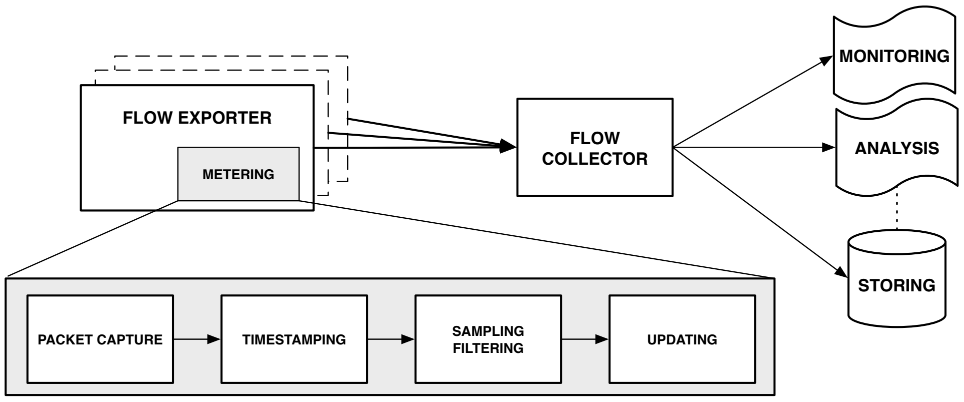


Fig. 3. IP Flow exporting and collecting architecture [22], [8].

matching in payload content. Flows are limited to information regarding network interactions. With this information, it is still possible, however, to identify communication patterns between hosts, when communication takes place and which amounts of packets and bytes have been moved. For many attacks, this information is sufficient. In any case, it is important to underline that flow-based intrusion detection is not supposed to substitute the packet-based one, but rather complements the approach by allowing early detection in environments in which payload-based inspection is not feasible. As described by Schaffrath *et al.* [23], in an ideal world payload-based solutions would always outperform flow-based ones in accuracy. In high-speed networks, however, the processing capabilities of the NIDS may be too limited to allow payload-based approaches.

III. IP Flows

In the last decade, flows have become quite popular in IP networks. Nowadays all major vendors equip their routers with flow accounting capabilities. Traffic information is collected and stored in flow records that provide an overview of network usage at different levels of granularity.

A. Flow definition

In literature, several definitions of an IP flow can be found [8], [24], [9]. This article follows the definition of *IP flow* as it was described by the IPFIX (IP Flow Information Export) working group within IETF [10], [22]:

> "A flow is defined as a set of IP packets passing an observation point in the network during a certain time interval. All packets belonging to a particular flow have a set of common properties."

In the IPFIX terminology, the *common properties* are called *flow keys*: they are, for example, source and destination addresses, source and destination port numbers and IP protocol:

$$(\mathtt{ip\_src}, \mathtt{ip\_dst}, \mathtt{port\_src}, \mathtt{port\_dst}, \mathtt{proto}).$$

Aggregated views on the network traffic can be obtained by choosing coarser grained flow definitions, according to the need of the network administrator, as discussed in the work of Fioreze *et al.* [24]. It is important to underline the difference between *flows* and *connections*, as used in the case of TCP. A flow can exist also in situations in which there is no TCP connection: an example of this is a UDP flow, where a set of packets has been sent from a certain source address/port to a certain destination address/port. Moreover, a flow does not have size restrictions: each communication between source and destination hosts will generate a flow, even if a single packet has been exchanged.

Accounting flows is a two-step process: *flow exporting*, and *flow collection*. These tasks are performed by two components: *flow exporter* and *flow collector*. Figure 3 shows this exporting/collecting process.

The *flow exporter*, also known as *observation point*, is responsible for the *metering* process, i.e., creating flow records from observed traffic. The *flow exporter* extracts the packet header from each packet seen on the monitored interface. Each packet header is marked with the *timestamp* when the header was captured. After that the header is processed by a *sampling-filtering* module, where it can be *sampled* (see Section III-C) or *filtered*. The final step is the *update* module. Each incoming packet header triggers an update to a flow entry in the *flow cache*. If there is no flow matching the packet header, a new flow entry is created. Once a flow record *expires*, it is sent to the flow collector. In case of Cisco NetFlow [8] and similarly in IPFIX [25], a flow is considered expired when:

- the flow was idle (no packets have been detected in the flow) for a longer time than a given threshold (known as *inactive timeout*). The default value for the inactive timeout for Cisco Netflow [8] is, for example, 15 seconds, but it can be changed according to the requirements of the network to be monitored.
- the flow reaches the maximum allowed lifetime. When this happens, its corresponding flow record is exported to the collector and, if necessary, a new flow record is created for that flow (*active timeout*). For Cisco Netflow, the active timeout is 30 minutes, but our experiences showed that shorter timeouts are also common. At the University of Twente, for example, an active timeout of 1 minute is used.
- the `FIN` or `RST` flags have been seen in a TCP flow.
- the flow-cache memory gets full. In this case, certain flow records are marked as expired and exported to the

collector. Least Recently Used (LRU) algorithms may be used to free the flow-cache memory, as well as heuristic algorithms.

The aim of the *flow collector* is to retrieve the flows created by the flow exporter and to store them in a form suitable for further monitoring or analysis.

B. Flow Export Protocols

A flow export protocol defines how flow records are transported between an exporter and a collector. Netflow version 5 [8], developed by Cisco, has a very simple flow export protocol that transports flow records of fixed size (48 bytes in total). A Netflow v5 flow record contains source and destination IP addresses and ports, start and end timestamps, type of service, level 3 protocol, TCP flags, next hop router, input and output SNMP interfaces, source and destination autonomous systems and network masks. Moreover, each flow carries aggregated information about the amount of packets and bytes exchanged.

Netflow version 9 and IPFIX propose flexible protocols in which flow record formats can be defined by using templates. The latter protocols allow also a larger set of parameters to be used as flow keys. An IPFIX packet is logically divided into sections known as *sets*. A message can normally consist of three kinds of sets, namely *Template sets* (format template exchange), *Data sets* (flow records) and *Options Template Sets* (necessary for the correct interpretation of a Template set). For a more detailed treatment of the IPFIX message format, see [22].

C. Sampling

IP flow accounting requires state information to be kept for each active flow. On high-speed links, there may be millions of packets per second and hundreds of thousands of active flows. If for each incoming packet, a flow lookup is performed and state information is kept for each flow, a heavy demand will be put on the CPU and memory resources of the flow exporter. In order to reduce this demand, sampling methods can be deployed. The IETF PSAMP (Packet Sampling) working group [26] is currently discussing the creation of possible standards in this area. It should be noted, however, that sampling not only lowers the demands put on the flow exporter, but also makes detection of intrusions harder. Several studies discuss the impact of sampling on intrusion detection and flow accounting. Examples are Brauckhoff *et al.* [27], Mai *et al.* [28] and Zseby *et al.* [29].

Two main categories of sampling can be identified: packet sampling and flow sampling.

- **Packet Sampling**: as explained in Izkue *et al.* [30], Wang *et al.* [31] and He *et al.* [32], sampling techniques can be divided into *systematic* and *random* ones. In *systematic* packet sampling, a packet is deterministically selected on the base of a time interval (*time-driven sampling*) or a sequence of packet arrivals (*event-driven sampling*). For example, it is possible to select a packet every t seconds, or a packet every n packets. In *random* packet sampling, on the other hand, the sampling process relies on a probability distribution function. The two main classes of random sampling are:
  - *n-in-N sampling*: The traffic is split into sequences of *N* packets. Out of these, *n* are randomly selected.
  - *probabilistic sampling*: Each packet is sampled with probability p. This sampling probability p can be fixed, or can depend on specific packet characteristics, such as for example the packet size.

  The deployment of one or more packet sampling strategies depends on which traffic characteristic the administrator is interested in. NetFlow is using an *n-in-N* sampling technique, usually in the form of *1-in-N* sampling.
- **Flow sampling**: similarly to random packet sampling, random flow sampling algorithms sample each flow with a random probability. *Sample and hold*, for example, is a sampling method proposed by Estan *et al.* [33] that accurately accounts for large flows. In this case, when the system detects the presence of a new packet that does not belong to any already existent flow, it creates a flow entry with probability p. If the new flow is created, all following packets belonging to the flow will be accounted, as opposed to packet sampling in which each packet independently undergoes the sampling procedure. It is easy at this point to imagine why this sampling strategy is biased towards large flows. Duffield *et al.* [34], [35] and Alon *et al.* [36] proposed *Smart Sampling* as a method to dynamically control the size of sampled data. *Smart Sampling*, both in the form of *threshold sampling* [35] and *priority sampling* [36], is based on the observation that packets and bytes in flows follow a heavy tailed distribution. A simple flow sampling strategy may omit flows that have large impact on the estimation of the total traffic of the network. To overcome this problem, Duffield *et al.* and Alon *et al.* propose sampling schemes in which the probability that a flow will be sampled depends on its size.

This section gave an overview of how flows are created. To understand how flows can be used for intrusion detection, we are now going to give a brief overview of the attacks present in our networks.

IV. Attack Classification

Several attack classifications have been described in literature [37]. These classifications usually distinguish between the following basic categories [38], [39]:

- **Physical attacks:** attacks based on damaging the computer and network hardware.
- **Buffer overflows:** attacks that gain control or crash a process on the target system by overflowing a buffer of that process.
- **Password attacks:** attacks trying to gain passwords, keys, etc. for a protected system.
- **(Distributed) Denial of Service attacks:** an attack which leads to situations in which legitimate users experience a diminished level of service or cannot access a service at all.
- **Information gathering attacks:** an attack that does not directly damage the target system, but gains information about the system, possibly to be used for further attacks

in the future. This category comprises network traffic sniffing and (port) **scans**.
- **Trojan horses:** a program disguised as a useful application, which deliberately performs unwanted actions.
- **Worms:** a program that self-propagates across a network. Self-propagation is the characteristic that differentiates worms from viruses (see below). A worm spread can be extremely fast: an example is the Sapphire/Slammer worm, which is known to have infected 90% of the vulnerable hosts in 10 minutes [40].
- **Viruses:** a virus is regarded as a worm that only replicates on the (infected) host computer. Hence, it needs user interactions to propagate to other hosts. Often, the definition also requires that a virus has to attach itself to files on the host, e.g., executable files, in order to be activated. As a consequence, the speed of spreading cannot be compared with a worm spread.

In addition, Hansman *et al.* [39] summaries under the category "Network attacks" various other attacks, such as spoofing, session hijacking and parameter tampering.

The previous categories should not be regarded as mutual exclusive classes of attacks. For example, buffer overflows and port scans can be regarded as separate categories of attacks, but also as specific techniques used by worms and DoS attacks. Rather, these categories describe general "concepts" of attacks that have been frequently observed in practice. Note that not all taxonomies provide a classification like the one given above. For example, Howard [41] focuses on a process-driven taxonomy, based on the objective of the attacker, the used tools, etc.

Nowadays, an additional threat has evolved pertaining Botnets. **Botnets** are groups of computers "infected with malicious program(s) that cause them to operate against the owners' intentions and without their knowledge", as defined in Lee *et al.* [42]. Botnets are remotely controlled by one or more *bot-masters*. Moreover, Botnets are the perfect infrastructure for setting up and supporting any kind of distributed attack, such as, for example, DoS attacks and SPAM campaigns. Infected hosts unknowingly become part of Botnets, and take part in malicious activities [43], [44]. The threats posed by Botnets are such that we decided to include them in our attack classification.

Flow-based intrusion detection, since it relies only on header information, can address only a subset of the attacks presented above. In particular, the research community currently provides approaches to detect the following classes of attacks:
- Denial of Service;
- Scans;
- Worms;
- Botnets.

Approaches to detect these attacks will be further discussed in Section VI.

V. Detection classification

According to Halme *et al.* [45], an Intrusion Detection System is an *anti-intrusion approach* that aims to *discriminate intrusion attempts and intrusion preparation from normal system usage*. Since the first papers on intrusion detection appeared in the Eighties of the previous century, several taxonomies of intrusion detection techniques were proposed. Our study identifies two main contributions to the field, the work of Debar *et al.* [11], [12] and that of Axelsson [13].

Debar *et al.* [11], [12] were among the first to propose an intrusion detection system taxonomy. Their classification focuses on the following elements:
- **Detection Method**: if a system bases the detection on a definition of *normal* behavior of the target system, it is called *behavior-based*. If it matches the input data against a definition of an attack, it is known as *knowledge-based*. In literature, the community usually refers to these classes with the names of *anomaly-based* and *misuse-based* solutions [46], [13], [14], [47], [48].
- **Behavior on detection**: a system can be proactive and act against the intruder (*active system*) or can generate alerts that will be later processed by a different system or a human operator (*passive system*).
- **Audit source location**: the data processed in order to detect intrusion can be *host* or *application logs*, *network packets* or *alerts* generated by other detection systems.
- **Detection Paradigm**: the IDS can detect the current status of the target system (secure or insecure) or can alert on a state transition (from secure to insecure).
- **Usage frequency**: the system can perform its task in real-time (*continuous monitoring*) or post-mortem (*periodic analysis*)

Axelsson [13] bases his taxonomy on the one proposed in Debar *et al.* [11], [12], but extends and completes it. In particular, beside the previously described characteristics, a system is described also on the basis of the following:
- **Locus of data-processing**: a system can be *centralized* or *distributed*, irrespectively of the origin of the data.
- **Locus of data-collection**: the data collection can be *centralised* or *distributed*.
- **Security**: the intrusion detection system can be itself target of security threats.
- **Degree of inter-operability**: a system can be built to work in *conjunction* with other systems (exchanging data) or *stand-alone*.

In his work, later followed by Almgren *et al.* [49], Axelsson focuses on detection methods, once again divided in two classes: anomaly-based and misuse-based. In that work, an *anomaly-based* system can be described as:
- **Self-learning**: the system is able to automatically build a model of the normal behavior of the system, or:
- **Programmed**: the definition of normality has to be provided by the system developer.

A *misuse-based* system, on the other hand, presents a unique subclass, *programmed*: the system is provided with a knowledge-base of attacks, against which it matches the inputs.

Figure 4 shows the detection capabilities of legal and illegal activities, for misuse (knowledge-based) and anomaly (behavior-based) systems, respectively. A misuse-based model is supposed to describe only illegal activities. In some cases, however, if the system is not *accurate* enough, legal activities

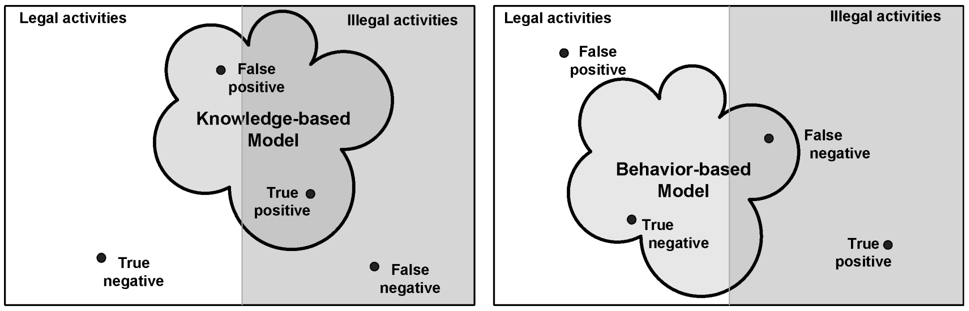


Fig. 4. Detection capabilities of different intrusion detection models [46].

can be flagged as intrusions; such events are called *false positives*. At the same time, if the model is not *complete*, it will not be able to report all malicious activities; unflagged illegal activities are known as *false negatives*. An anomaly-based model, on the other hand, is supposed to describe only legal activities (*normality*). Also in this case, incompleteness and inaccuracy can lead to false positive and false negatives.

Finally, in [13], Axelsson introduces a third class of systems in which both anomaly-based inspired characteristics and misused-based ones coexist. In his work, such systems are known as *compound*.

In their taxonomies, Debar *et al.* [11], [12] and Axelsson [13] consider a wider spectrum of categories, including some that are outside the scope of this paper. For example, they distinguish between *host-based* and *network-based* detection approaches. The former analyses the status of a single host, monitoring internal functionality (e.g., CPU usage, system call traces, log-in attempts). The latter bases its analysis on network information and can monitor an entire network. In this paper, we are only interested in this second kind of systems, since we are analyzing network data only. In particular, we are interested in flow-based approaches; since such approaches are relatively new, the taxonomies found in literature do not explicitly consider them. Despite this, since flows represent network-based aggregated data, we still consider the taxonomies useful for our purpose.

VI. Flow-based solutions

This section presents the state of the art solutions for each category of attack that can be detected using flows (see Section IV). Moreover, it classifies each contribution according to the taxonomies presented in Section V.

A. Denial of Service

Detection of Denial of Service is often addressed in flow based intrusion detection. These attacks, by their nature, can produce variations in the traffic volume that are usually still visible at flow scale.

It is important to underline, nevertheless, that in case of flow-based detection we are implicitly addressing the problem of *brute force* DoS attacks, i.e., a type of DoS that relies on resource exhaustion or network overloading. Unfortunately, it is almost impossible to directly detect *semantic DoS* attacks, i.e., attacks in which the service interruption is caused by the payload contents. For example, let us consider the (nowadays out-of-date) Ping of Death attack. In such attack, the attacker sends malformed or otherwise malicious ping packets, which causes the victim system to crash. Since this attack does generate a single ICMP flow, the attack would most likely go undetected. There would be, indeed, no change in flow frequency and intensity. A different case of semantic attack would be one that changes the distribution of flows. An example is the DoS effect related to the scanning phase of the Sapphire/Slammer worm [40]: while spreading, the worm provokes the crash of Microsoft SQL Server hosted on the target machine. Nevertheless, at flow basis, the detection of this attack would be most probably related to the scanning phase, and not to the DoS itself.

An overview of how often DoS attacks appear in practice is given in Moore *et al.* [50]. They estimate that, based on an analysis conducted over multiple one-week traces for a period of three years, on average the number of different victim IPs on the entire Internet is 24.5/hours. Even though Moore *et al.* do not specify if they investigated brute force or semantic attacks, the statistics clearly show that DoS attacks detection is, still in these days, a problem that requires experts' attentions.

There are two main examples of anomaly-based DoS detection in high-speed networks, using flow information only. The work of Li *et al.* [51] and Gao *et al.* [52], in the first place, approach the problem using aggregate flow measures collected in appropriate data structures, named *sketches*. A sketch is originally a one-dimensional hash table suitable for fast storing of information: it mainly counts occurrences of an event. In their papers, the authors work with 2D sketches, a more powerful extension of the original ones, in which, for each dimension, a set of flow-derived fields is hashed. Sketches permit to statistically characterize how the traffic varies over time, simply by tracking the presence of a flow in a specified time frame. An anomaly-based engine triggers alarms based on a forecast value of the measure the system is supposed to monitor: a sharp variation from the mean is flagged as an anomaly. A simple example of the use of sketches in DoS attacks is the detection of SYN Flooding attacks [53], as described in Gao *et al.* [52]. In this case, the sketch is supposed to store, for each time frame and each tuple `(dest_IP, dest_port)`, the difference between the number of SYN packets and the number of SYN/ACKs. If the stored value for the current time deviates from the expected one, a DoS SYN Flooding attack is going on. The sketch-based approach could potentially be deployed also without the use of flows, relying in this case on header inspection. Nevertheless, in this case the data reduction gain provided by flows would be most probably lost. Gao *et al.* developed a prototype that receives exported flows from a netflow-enable router in real time.

A similar approach is proposed by Zhao *et al.* [54]. In this case, a data-streaming algorithm is used to filter part of the traffic, and identify IPs that show an abnormal number of connections. The authors consider both the case in which a host is the source of an abnormal number of outgoing connections (*large fan-out*), as well as the case in which a host is the destination of an unusual number of connection attempts (*large fan-in*). The first case matches the definition of a scanning host, while the second is used for detecting DoS victims. The method is based on 2D hash tables, clearly resembling the work of [51] and [52]. In their paper, Zhao *et al.* also apply a flow *sampling* algorithm (see Section III), to reduce the amount of data to be processed and significantly raise processing speed. At the same time, since sampling further reduces the available information, the authors developed statistical formulas to accurately estimate the *fan-in/fan-out* of the considered hosts.

A more detailed approach is presented by Kim *et al.* [55]: in this paper many different DoS attacks are described in terms of traffic patterns, based on flow characteristics. In particular, the authors focus on the number of flows and packets, the flow and packet sizes, total bandwidth used as well as average flow size and number of packets per flow. An example of attack pattern is the one produced by a SYN Flooding attack: a large flow count, yet small packet counts, as well as small flow and packet sizes and no constraints on the bandwidth and the total amount of packets. The pattern is significantly different from the one generated by an ICMP or UDP flooding attack, in which we have large bandwidth consumption and the transfer of a large number of packets. Kim *et al.* clearly identify the metrics they are interested in and formalize them into *detection functions* that give the likelihood of a traffic pattern representing an attack.

In the context of DoS monitoring and detection, it is important to cite also the work of Münz *et al.* [56], which propose a general platform for DoS detection. The system, known as TOPAS (Traffic flOw and Packet Analysis System), acts as a flow collector for multiple sources and locations, offering preprocessing capabilities in order to obtain an information format suitable for further processing. On this platform, many different detection modules can run in real-time according to the necessities of the network administrator. Examples of modules are a *SYN flood detection module*, a *traceback module* (to allow identification of the entry point of spoofed packets in the attacked network) and a *Web Server overloading module* (focusing on DoS attacks using HTTP requests). The work has been developed within the context of the European Diadem Firewall project, which specifically focuses on DoS and DDoS detection [57].

Attention must also be given to the work of Lakhina *et al.* [58], [59], [60], [61]. The analysis is conducted on flow aggregation, namely on origin-destination flows between Points of Presence (PoP) on the Abilene [17] and Sprint-Europe [62] networks. On this small set of pairs (only n^2, where n is the number of PoPs), it is possible, through principal component analysis, to decompose the traffic flowing through the backbone in time related traffic trends (*eigenflows*). There are three types of eigenflows: deterministic eigenflows that show a periodical trend (day-night pattern), spike eigenflows that show isolated values that strongly deviate from the average and noise eigenflows that appears to be roughly Gaussian. The spike components reveal the presence of a traffic anomaly. The proposed method is general enough to capture various kinds of anomalies, due to failures or attacks, and is appropriate for almost all the attack classes we are interested in (DoS, scans and worms).

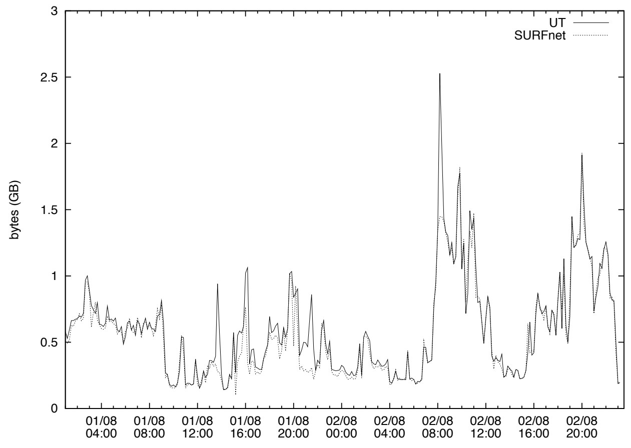
B. Scans

Scans are usually characterized by small packets that probe the target systems. Keeping this characteristic in mind, it is easy to imagine that scans can easily create a large number of different flows. There are three categories of scans: (i) a host scanning a specific port on many destination hosts (*horizontal scan*); (ii) a host scanning several ports on a single destination host (*vertical scan*); (iii) a combination of both (*block scan*). Irrespectively of the kind of scan, the result will be a variation of the flow traffic in the network. At the same time, scans are less likely to have impact on the total traffic volume, as shown in Sperotto *et al.* [63].

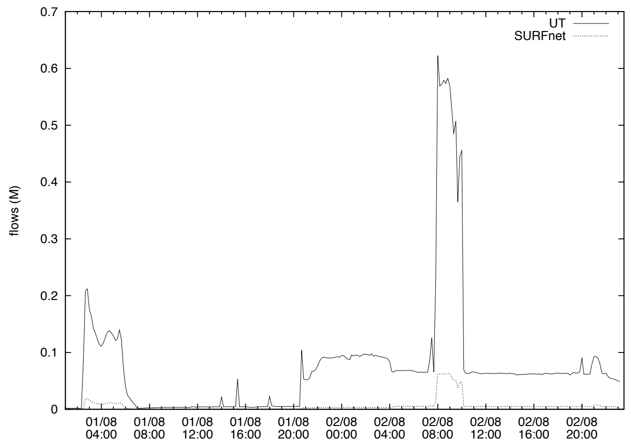
Figure 5 shows an example of SSH flows captured in 2007 at the University of Twente (UT) and SURFnet [64], the UT Internet service provider. The byte time series (Fig. 5(a)) is quite irregular, with sharp high- and down-peaks that do not clearly indicate the presence of an attack. On the other hand, the flow time series (Fig. 5(b)) shows sudden and frequent peaks, during which the number of flows can rise to several hundreds of thousands per observation bin (10 minutes, in the case of Figure 5). After a more detailed analysis, these peaks appeared to correspond to multiple SSH scanning sessions, trying to guess user names and passwords. An interesting difference between SURFnet and the UT is that SURFnet applies 1:100 packet sampling, whereas the UT does not apply packet sampling. Still Figure 5 shows that scans can even be detected in SURFnet, despite the sampling.

In literature, scans have generally been investigated by considering their most obvious characteristic: the scanning source shows an unnaturally high number of outgoing connections. The problem has been approached in this way by Zhao *et al.* [54], already cited in the previous section. Looking at host behavior from an incoming/outgoing connection perspective allows addressing DoS and scan attacks as faces of the same problem: hosts with an inadequate and unusual fan-in/out. Similarly, Kim *et al.* [55] attempt to describe a scan in terms of traffic patterns, as already explained in the case of DoS. The authors differentiate between network (horizontal) scans and host (vertical) scans.

The approach described in Wagner *et al.* [65] is not related to traffic volume anomalies. In this case, the probabilistic measure of entropy is used to disclose regularity in connection-based traffic (flows). Entropy has been introduced in Information Theory in 1948 [66] and, generally speaking, is a measure of randomness and *uncertainty* of a stochastic process. Entropy is also related to lossless data compression:



(a) Bytes



(b) Flows

Fig. 5. Byte (a) and flow (b) time series for SSH traffic at the University of Twente network and SURFnet [63].

the theoretical limit of the compression rate of a sequence of bits is exactly the entropy of the sequence. Starting from this well-known result, Wagner *et al.* created an efficient analysis procedure based on compression of sequences of network measurements. They observe that, in the case of a scanning host, the overall entropy in a specific time window is subdued to a change. In particular, the presence of many flows with the same source IPs (the scanning host) will lead to an abrupt decrease of the entropy in the distribution of the source IP addresses. At the same time, the scanning host will attempt to contact many different destination IPs on (possibly) different ports, generating an increase in these entropy measurements. The combined observation of multiple entropy variations helps in validating the presence of an attack. Other approaches are based on logistic regression [67] and distances from baseline models [68].

C. Worms

Worm behavior is usually divided into a target discovery phase (the worm explores the network in order to find vulnerable systems) and a transfer phase (the actual code transfer takes place) [71], [72]. Code Red [73] and Sapphire/Slammer [40] are examples of this mechanism. Flow-based detection systems usually focus on the target discovery phase, since the

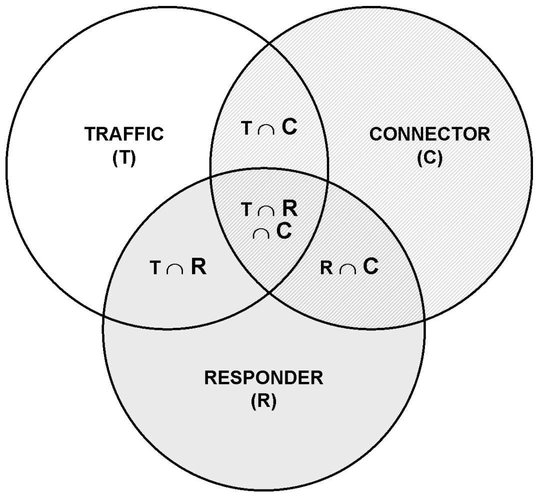


Fig. 6. Host classes and their intersections [69], [70].

transfer of malicious code cannot easily be detected without analyzing the payload. In many cases, worm detection can be similar to scan detection, and many researchers use the same approach for both threats. The approach adopted by Wagner *et al.* [65], for example, can naturally be extended to worms, as well as the ones of Zhao *et al.* [54] and Gao *et al.* [52].

Dübendorfer *et al.* [69] and Wagner *et al.* [70] attempt to characterize the host behavior on the basis of incoming and outgoing connections. The proposed algorithm assigns hosts to a set of classes. The definition of these classes is such that only suspicious hosts will belong to them. The *traffic class* groups hosts that send more traffic than what they receive. Hosts that show an unusual high number of outgoing connections are part of the *connector class*. Finally, hosts involved in many bidirectional connections belong to the *responder class*. In the proposed model, a host can belong to more than one class. Figure 6 describes the three classes and their possible intersections. The method aims to periodically check the status of the hosts of an entire network. In this way, it is able to detect worm spreads, as they cause massive changes in the cardinality of one or more classes. Moreover, Dübendorfer *et al.* [69], by properly filtering the interesting flows, manage to identify both e-mail spreading worms and scanning worms, without concerns about their scanning strategies.

A different approach is taken by Dressler *et al.* [74], that uses the correlation between flows and honeypots logs. In this case, the need for a *ground truth*, i.e., a trusted source of information for the system validation, made the authors rely on a honeypot. In this way, deploying at the same time honeypot, flow monitor and a collecting database, it is possible to carefully identify *worm flow-signatures*, that is sequence of connections and flow-related information about the scanning and transmitting behavior of a worm. According to the presented results, the approach seems to be promising.

Finally, Collins *et al.* [75] propose a solution to the problem of *hit-list worms* detection. A hit-list worm is a worm that bases its scanning strategy on the sequential probing of a predefined list of hosts that are supposed to be always online. This technique is used because worms usually have a slow initial spreading phase, and the use of a hit-list consistently increase the initial infection speed. Since hit-lists are commonly used

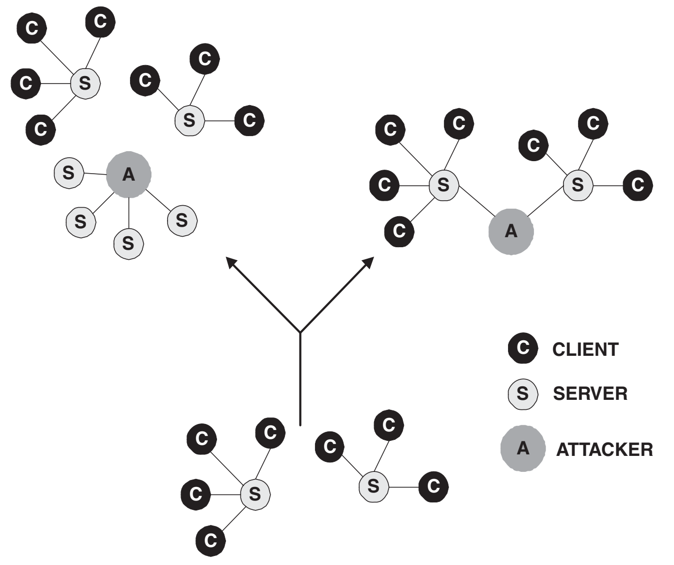


Fig. 7. Example of graph based hit-list worm spreading analysis [75].

to start infections, detecting them as soon as possible may be quite useful. Collins *et al.* employ a graph-based algorithm that slices the network according to a monitored protocol (like HTTP, FTP, SMTP, or Oracle). They argue that the number of hosts normally involved in the use of a certain protocol, i.e., the number of vertexes in the graph, is in average regular over time. Also the pattern of communication between hosts, i.e., the cardinality of the connected components in the graph (connected subgraphs with a maximal number of vertexes), has the same property. This regularity is disturbed only when a new host starts to scan the network following a hit-list: in this case, the authors observe a larger number of vertexes in the graph (the scanned hosts) and a drastically enlarged cardinality of the connected components. The scanning host, indeed, will communicate with servers (in its hit-list) that in normal conditions do not have any connection. Figure 7 shows cases of hit-list infections that modify the number of vertexes in the graph and the cardinality of the largest connected component. In the example, two servers, depicted at the bottom of the figure, communicate with, respectively, four and three clients during a normal observation period. The two disjoint sets of hosts will form two connected components in the *protocol graph*. During a malicious observation period, i.e., while a hit-list worm is spreading, two situations can be observed. In the first one, on the upper left of the figure, the attacker contacts servers that do not normally appear in the monitored traffic: as consequence, the cardinality of the vertex set in the protocol graph will increase, meeting the first detection condition in [75]. In the second case, on the upper right corner, an attacker will contact both servers, since they are on its hit-list. As a consequence, the two connected components described in the example will be reduced to one, meeting the second detection condition.

D. Botnets

As explained in Section IV, Botnets consist of infected hosts (bots) controlled by a central entity, known as master (or bot-master). As these networks tend to be spread over multiple administrative zones, complete identification of bots is a difficult problem. Since bots are no longer harmful once the master is isolated, a straightforward mitigation approach is to identify the master. Nevertheless, as Zhu *et al.* [79] pointed out in their survey on Botnet research, the defense against botnets is not yet efficient and the research in this field is still in its infancy.

As a fact, many Botnets used to rely on IRC channels, which can be identified at flow level, as described in the work of Karasaridis *et al.* [76]. The authors propose a model of IRC traffic that does not rely on specific port numbers. Karasaridis *et al.* address two main points. First, they propose a multistage procedure for detecting Botnets controllers. Starting from reports of malicious activity obtained from diverse sources (e.g., scan logs, spam logs, and viruses), the authors identify groups of flows involved in suspicious communications (*candidate controller conversations*). These conversations may happen between a host and a candidate server (*controller*) that use either an IRC port (e.g., 6667, 6668 or 7000) or that hides the control traffic using a different protocol. In the second case, the candidate conversation is checked against the flow model. The second aim of Karasaridis *et al.* is, once the controllers have been identified, to group the suspected bots into behavioral groups, i.e., clusters of bots that show the same activity pattern. For this purpose, they suggested a hierarchical clustering procedure that groups the hosts based on their port activities. In [76], the authors also explain why Botnet detection slightly differs from scan or DoS detection. For scans and DoS, current research aims at *real time* identification, with alerts that permit the network administrator to intervene as soon as possible. In the case of Botnets, only long time observations can lead to the identification of the bots and controller.

In a similar way, the work of Livadas *et al.* [77] and Strayer *et al.* [44] approach the problem by modeling the TCP flows of IRC chats. The authors present the first results of a study pertaining to the use of machine learning techniques for Botnet traffic identification. In particular, they structure their approach in order to answer two research questions: is it possible to distinguish between i) IRC and non-IRC traffic; ii) botnet IRC traffic and normal IRC traffic. In the paper, the effectiveness of machine learning methods, such as Naive Bayes classifiers, Bayesian networks and classification trees, is tested. The input is an enriched version of flows (including additional information, such as variance of the bytes per packet in the flow, or the number of packets for which the PUSH flag is set). The work shows that automatic identification of Botnet IRC traffic seems possible.

A different approach is proposed by Gu *et al.* [78]. They developed a Botnet detector, *BotMiner*, which is independent of Botnet Command and Control (C&C) protocols and structures. Gu *et al.* developed a detection framework that aims to characterize a Botnet according to the following definition:

> A *coordinated group* of *malware* instances that are *controlled* via C&C channels.

BotMiner sniffs the traffic at the observation point and conducts two parallel analyses. On one side, it relies on flows for detecting groups of hosts with similar communication patterns. On the other side, it inspects packet payloads (via

TABLE I
CATEGORIZATION OF THE PROPOSED SOLUTIONS ACCORDING TO THE TAXONOMY.

System	Detection Method	Behaviour on detection	Usage Frequency	Data processing	Data collection
Li *et al.* [51] Gao *et al.* [52]	anomaly	active	real-time	centralised	distributed
Zhao *et al.* [54]	not spec	not spec	real-time	centralized	distributed
Kim *et al.* [55]	misuse	passive	real-time	centralized	distributed
Münz *et al.* [56]	compound	passive	real-time	centralized	distributed
Lakhina *et al.* [58], [59], [60], [61]	anomaly	passive	real-time	centralized	centralized
Wagner *et al.* [65]	anomaly	passive	real-time/batch	centralized	centralized
Gates *et al.* [67]	misuse	passive	batch	centralized	centralized
Stoecklin *et al.* [68]	anomaly	passive	batch	centralized	centralized
Dübendorfer *et al.* [69] [70]	compound	passive	real-time	centralized	centralized
Collins *et al.* [75]	anomaly	passive	real-time	centralized	centralized
Dressler *et al.* [74]	misuse	passive	real-time	centralized	centralized
Karasaridis *et al.* [76]	misuse	passive	real-time	centralized	centralized
Livadas *et al.* [44] [77]	not spec	passive	batch	centralized	centralized
Gu *et al.* [78]	anomaly	passive	real-time	centralized	centralized

Snort) in order to detect anomalous activities. These activities are then clustered together in order to detect groups of hosts that have similar malicious behavior. In both steps, unsupervised clustering techniques have been used. As the authors describe, both step are necessary in order to properly identify possible bots, and a *cross correlation* phase is performed in order to merge the results of the previous analyses and extract meaningful groups of malicious host that form a Botnet. The approach, which has already been implemented in a working prototype, shows good detection results. Moreover, it clearly shows that the problem of Botnet detection is more complex than the general problem of attack detection. A misbehaving host, indeed, is not sufficient to indicate the presence of a Botnet. A more sophisticated intra-host communication analysis is needed to characterize the group nature of Botnets.

Even though Gu *et al.* [78] and Karasaridis *et al.* [76] present better results than Livadas *et al.* [77] and Strayer *et al.* [44], all the contributions clearly show that the problem of Botnet detection still remains unsolved. This is mainly due to the subtle and highly dynamic evolution of the Botnets themselves. Since the research on Botnet identification is still in its beginning phase, a strong research effort is needed to develop effective detection procedures. In this regard, flow-based approaches play an important role.

E. Solutions classification

The intrusion detection taxonomies presented in Section V allowed us to categorize the state of the art in the field of flow-based intrusion detection. However, in our specific case, not all the categories in the taxonomies are relevant to our problem. For example, network information is the only *audit data* we are interested in, so this category has been omitted from our study. The *detection paradigm* (state/transition-based) is applicable mainly to host-based solutions, and for this reason has also been discarded. We also have not considered Axelsson's *security* class. Only one of the contributions, indeed, explicitly addresses the problem of attack resilience [52]. Finally, it is important to notice that, at the moment, the main research concern is still on developing flow-based detection engines, and less effort is put on problems like *interoperability* of different instances of the IDS, or between the IDS and other network components (firewalls, routers...). In our survey, only a few contributions specifically address this subject, such as Li *et al.* [51] and Gao *et al.* [52].

Table I, which presents our classification, gives some insight in the current research trends in flow-based intrusion detection. As it has been for payload-based solution, also in this case, the *anomaly-/misuse-based* classes play an important role: we can see contribution in both fields. Moreover, some researchers, such as Münz *et al.* [56], Dübendorfer *et al.* [69] and Wagner *et al.* [70], developed *compound* methods. This is due to the interest in joining the strengths of both anomaly and misuse-based approaches, as well as to the increasing interest in multi-purpose platforms that offer a shared base for different detection modules. The work of Gu *et al.* [78], on the other hand, is classified as anomaly-based. It indeed uses Snort only as a complementary source of data, while the entire detection engine is based on anomaly techniques. On some occasions [44], [54], [77], the detection approach is unclear or not specified. This happens when these works address more general problems than detection, and attack identification serves only as a possible application. For example, Zhao *et al.* [54] are interested in *super-sources/destinations* as a more general problem of *scan/DoS detection*. On the other hand, Strayer *et al.* [44] and Livadas *et al.* [77] do not specify if they aim at modeling either the normal behavior of IRC conversations or the anomalous one pertaining to the command-and-control streams. Therefore, it has not been possible to classify the contribution regarding this category.

By considering the behavior on detection, the focus seems to be on passive solutions, which complete their task with the rising of an alert to the network administrator. This also means that the majority of the solutions heavily rely on human intervention for attack mitigation and blocking. At the same time, nevertheless, our classification shows that there is a clear preference for *real-time solutions*, which clearly signals the need for fast responses in flow-based security.

The majority of the contributions rely on centralized *data processing*. Flows are a powerful approach to data reduction, as already pointed out in Section II. In many cases, a stand-alone machine will have enough processing power to deal with the flow-data stream. On the other hand, flows are particularly suitable to be exported towards remote collection points, making it extremely easy to develop a system based on

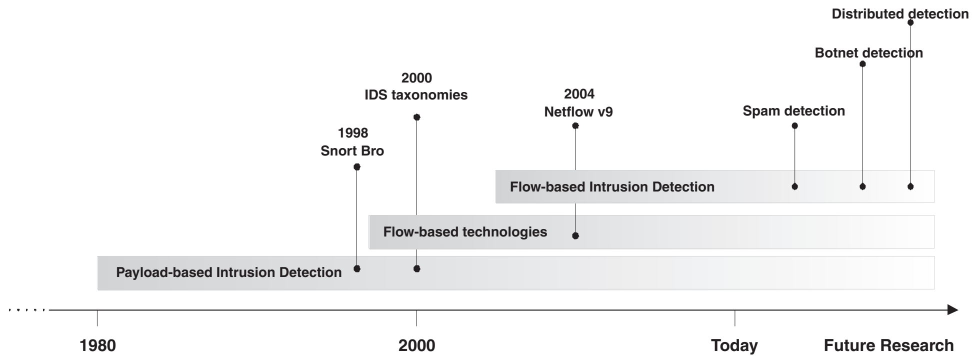


Fig. 8. Time line of evolution of intrusion detection and flow-based technologies.

distributed *data collection* points. The majority of the solutions that we studied assume a single (centralized) collection point for the ease of analysis, but the authors do not explicitly exclude the possibility of distributed collection.

VII. Conclusion

This paper presented a survey of the state of the art of flow-based intrusion detection, focusing on the period 2002–2008. During this time, flow-based techniques attracted the interest of researchers, especially for analysis of high-speed networks. The recent spread of 1-10Gbps technologies, and the day by day increasing network usage and load, have clearly pointed out that *scalability* is a growing problem. In this context, flow based solutions to monitor and, moreover, to detect intrusions help to solve the problem. They achieve, indeed, *data and processing time reduction*, opening the way to high-speed detection on large infrastructures.

This paper also showed, however, that in some cases the complete absence of payload should still be perceived as the main drawback of flow-based approaches. For example, the use of flow-based techniques makes it very difficult to detect so-called *semantic attacks* (see Section VI-A); attacks for which the disruptive power is in the payload, and which do not create visible flow variations (bytes, number of packets or number of flows). Nevertheless, as mentioned in Section II, flow-based intrusion detection is not meant to substitute payload-based solutions, but to complement them in situations where technological constraints make payload-based techniques infeasible.

Figure 8 shows, in a schematic time line, the evolution of payload-based intrusion detection, flow-based technologies and flow-based intrusion detection. Payload-based solutions constituted the first effort in developing network-based intrusion detection. Nevertheless, they are, still today, a meaningful approach to security. Figure 8 also shows the rise of flow-based technologies (see Section III). Once flow-based monitoring became an established technology, we can see how flows became also a source of data for intrusion detection (see also Section VI). The paper showed that the major efforts in flow-based detection concentrate on DoS, scan and worm detection, while Botnet detection appears to be a more recent research field.

Figure 8 also identifies some open issues that should be addressed in future research. First, the emphasis will be on improving the *local detection* of threats. An example would be to extend the current research to the detection of unsolicited e-mail. For this, as far as we know, only sporadic flow-based contributions have been proposed [80]. Detection of SPAM sources would address one of the most serious issues in our networks, since it has been estimated that the percentage of SPAM in the first half of 2008 has been 75-85% [81]. Moreover, Ramachandran *et al.* [82] estimated that $\sim$80% of the spam messages are sent by Botnets. In our opinion, Botnet detection is the second major research challenge. Botnet detection will involve long-term analysis of *wide infrastructures* as well as integration of multiple detection methods (Botnets are the source of diverse attacks). Since Botnets are naturally spread over multiple networks, a single monitoring point will probably not be sufficient for detection. To overcome this problem, a third area for future work is the development of *distributed flow-based detection* systems. Distributed detection is particularly important, also because the amount of traffic on high-speed network is still increasing, suggesting that scalability will remain an issue in the future.

Acknowledgement

This work was supported in part by the IST Network of Excellence EMANICS funded by the European Union under contract number FP6-2004-IST-026854-NoE.

References

[1] Computer Economics, “2007 malware report: The economic impact of viruses, spyware, adware, botnets, and other malicious code,” Jul. 2008. [Online]. Available: http://www.computereconomics.com

[2] M. Roesch, “Snort, intrusion detection system,” Jul. 2008. [Online]. Available: http://www.snort.org

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

[4] H. Lai, S. Cai, H. Huang, J. Xie, and H. Li, “A parallel intrusion detection system for high-speed networks,” in *Proc. of the Second International Conference Applied Cryptography and Network Security (ACNS’04)*, May 2004, pp. 439–451.

[5] M. Gao, K. Zhang, and J. Lu, “Efficient packet matching for gigabit network intrusion detection using TCAMs,” in *Proc. of 20th International Conferece on Advanced Information Networking and Applications (AINA’06)*, 2006, pp. 249–254.

[6] L. Heberlein, G. Dias, K. Levitt, B. Mukherjee, J.Wood, and D. Wolber, “A network security monitor,” in *Proc. of IEEE Computer Society Symposium on Research in Security and Privacy*, May 1990, pp. 296–304.

[7] S. Staniford-Chen, S. Cheung, R. Crawford, M. Dilger, J. Frank, J. Hoagl, K. Levitt, C. Wee, R. Yip, and D. Zerkle, "GrIDS - a graph based intrusion detection system for large networks," in *Proc. of the 19th National Information Systems Security Conference (NISS '96)*, 1996, pp. 361–370.

[8] Cisco.com, "Cisco IOS NetFlow Configuration Guide, Release 12.4," http://www.cisco.com, Jul. 2008.

[9] B. Claise, "Cisco Systems NetFlow Services Export Version 9," RFC 3954 (Informational), Jul. 2008. [Online]. Available: http://www.ietf.org/rfc/rfc3954.txt

[10] J. Quittek, T. Zseby, B. Claise, and S. Zander, "Requirements for IP Flow Information Export (IPFIX)," RFC 3917 (Informational), Jul. 2008. [Online]. Available: http://www.ietf.org/rfc/rfc3917.txt

[11] H. Debar, M. Dacier, and A. Wespi, "Towards a taxonomy of intrusion-detection systems," *Computer Networks*, vol. 31, no. 9, pp. 805–822, Apr. 1999.

[12] ——, "A revised taxonomy for intrusion detection systems," *Annales des Telecommunications*, vol. 55, no. 7–8, pp. 361–378, Jul. 2000.

[13] S. Axelsson, "Intrusion detection systems: A survey and taxonomy," Chalmers Univ., Tech. Rep. 99-15, Mar. 2000.

[14] H. Debar and J. Viinikka, "Intrusion detection: Introduction to intrusion detection and security information management," in *Foundations of Security Analysis and Design III*, Sep. 2005, pp. 207–236.

[15] A. Lazarevic, V. Kumar, and J. Srivastava, "Intrusion detection: A survey," *Managing Cyber Threats*, pp. 19–78, June 2005.

[16] CERT Coordination Center, http://www.cert.org/certcc.html, Jul. 2008.

[17] Internet2 NetFlow Weekly Reports, http://netflow.internet2.edu/weekly, Jul. 2008.

[18] H. Dreger, A. Feldmann, V. Paxson, and R. Sommer, "Operational experiences with high-volume network intrusion detection," in *Proc. SIGSAC: 11th ACM Conference on Computer and Communications Security (CSS'04)*, 2004, pp. 2–11.

[19] Z. Fadlullah, T. Taleb, N. Ansari, K. Hashimoto, Y. Y. Miyake, Y. Nemoto, and N.Kato, "Combating against attacks on encrypted protocols," in *IEEE International Conference on Communications (ICC '07).*, June 2007, pp. 1211–1216.

[20] T. Taleb, Z. M. Fadlullah, K. Hashimoto, Y. Nemoto, and N. Kato, "Tracing back attacks against encrypted protocols," in *Proc. of the 2007 international conference on Wireless communications and mobile computing (IWCMC '07)*, 2007, pp. 121–126.

[21] S. Song and Z. Chen, "Adaptive network flow clustering," in *IEEE International Conference on Networking, Sensing and Control (ICNSC07)*, April 2007, pp. 596–601.

[22] B. Claise, "Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of IP Traffic Flow Information," RFC 5101 (Proposed Standard), Jul. 2008. [Online]. Available: http://www.ietf.org/rfc/rfc5101.txt

[23] G. Schaffrath and B. Stiller, "Conceptual integration of flow-based and packet-based network intrusion detection," in *Proc. of 2nd International Conference on Autonomous Infrastructure, Management and Security (AIMS '08)*, 2008, pp. 190–194.

[24] T. Fioreze, M. O. Wolbers, R. van de Meent, and A. Pras, "Finding elephant flows for optical networks," in *Proc. of 10th IFIP/IEEE International Symposium on Integrated Network Management (IM '07)*, May 2007, pp. 627–640.

[25] S. Leinen, "Evaluation of Candidate Protocols for IP Flow Information Export (IPFIX)," RFC 3955 (Informational), Jul 2008.

[26] "Packet Sampling (PSAMP) working group," http://www.ietf.org/html.charters/psamp-charter.html, Jul. 2008.

[27] D. Brauckhoff, B. Tellenbach, A. Wagner, M. May, and A. Lakhina, "Impact of packet sampling on anomaly detection metrics," in *Proc. of the 6th ACM SIGCOMM conference on Internet measurement (IMC '06)*, 2006, pp. 159–164.

[28] J. Mai, C.-N. Chuah, A. Sridharan, T. Ye, and H. Zang, "Is sampled data sufficient for anomaly detection?" in *Proc. of the 6th ACM SIGCOMM Conference on Internet Measurement (IMC'06)*, 2006, pp. 165–176.

[29] T. Zseby, T. Hirsch, and B. Claise, "Packet sampling for flow accounting: Challenges and limitations." in *Proc. of 9th International Conference on Passive and Active Measurement (PAM'08)*, 2008, pp. 61–71.

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

[31] H. Wang, Y. Lin, Y. Jin, and S. Cheng, "Easily-implemented adaptive packet sampling for high speed networks flow measurement," in *Proc. of 6th International Conference on Computational Science (ICCS'06)*, 2006, pp. 128–135.

[32] G. He and J. C. Hou, "An in-depth, analytical study of sampling techniques for self-similar internet traffic," in *Proc.of 25th IEEE International Conference on Distributed Computing Systems (ICDCS'05)*, 2005, pp. 404–413.

[33] C. Estan and G. Varghese, "New directions in traffic measurement and accounting: Focusing on the elephants, ignoring the mice," *ACM Transactions on Computer Systems (TOCS)*, vol. 21, no. 3, pp. 270–313, 2003.

[34] N. Duffield, C. Lund, and M. Thorup, "Flow sampling under hard resource constraints," *SIGMETRICS Perform. Eval. Rev.*, vol. 32, no. 1, pp. 85–96, 2004.

[35] ——, "Learn more, sample less: control of volume and variance in network measurement," *IEEE Transactions on Information Theory*, vol. 51, no. 5, pp. 1756–1775, May 2005.

[36] N. Alon, N. Duffield, C. Lund, and M. Thorup, "Estimating arbitrary subset sums with few probes," in *Proc. of the twenty-fourth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems (PODS '05)*, 2005, pp. 317–325.

[37] V. Igure and R. Williams, "Taxonomies of attacks and vulnerabilities in computer systems," *Communications Surveys & Tutorials, IEEE*, vol. 10, no. 1, pp. 6–19, 2008.

[38] N. Weaver, V. Paxson, S. Staniford, and R. Cunningham, "A taxonomy of computer worms," in *Proc. of 2003 ACM workshop on Rapid malcode (WORM'03)*, 2003, pp. 11–18.

[39] S. Hansman and R. Hunt, "A taxonomy of network and computer attacks," *Computers & Security*, vol. 24, no. 1, pp. 31–43, Feb. 2005.

[40] D. Moore, V. Paxson, S. Savage, C. Shannon, S. Staniford, and N. Weaver, "Inside the slammer worm," *IEEE Security & Privacy*, vol. 1, no. 4, pp. 33–39, Jul.-Aug. 2003.

[41] J. D. Howard, "An analysis of security incidents on the internet 1989-1995," Ph.D. dissertation, Carnegie Mellon University, 1998.

[42] *Botnet Detection. Countering the Largest Security Threat.* Spinger, 2008, vol. 36.

[43] D. Dagon, G. Gu, and C. Lee, "A taxonomy of botnet structures," in *Botnet Detection*, vol. 36, Oct. 2007, pp. 143–164.

[44] W. Strayer, D. Lapsely, R. Walsh, and C. Livadas, "Botnet detection based on network behavior," in *Botnet Detection*, W. Lee, C. Wang, and D. Dagon, Eds., vol. 36, 2008, pp. 1–24.

[45] L. R. Halme and R. K. Bauer, "AINT misbehaving – A taxonomy of anti-intrusion techniques," in *Proc. of 18th NIST-NCSC National Information Systems Security Conference*, 1995, pp. 163–172.

[46] B. Morin and L. Mé, "Intrusion detection and virology: an analysis of differences, similarities and complementariness," *Journal in Computer Virology*, vol. 3, pp. 39–49, Apr. 2007.

[47] P. Li, M. Salour, and X. Su, "A survey of internet worm detection and containment," *IEEE Communications Surveys & Tutorials*, vol. 10, no. 1, pp. 20–35, 2008.

[48] M. Garuba, C. Liu, and D. Fraites, "Intrusion techniques: Comparative study of network intrusion detection systems," in *Proc. of 5th International Conference on Information Technology: New Generations (ITNG '08)*, Apr. 2008, pp. 592–598.

[49] M. Almgren, E. L. Barse, and E. Jonsson, "Consolidation and evaluation of IDS taxonomies," in *Proc. of 8th Nordic Workshop on Secure IT systems (NordSec '03)*, Oct. 2003.

[50] D. Moore, C. Shannon, D. J. Brown, G. M. Voelker, and S. Savage, "Inferring Internet denial-of-service activity," *ACM Transactions on Computer Systems*, vol. 24, no. 2, pp. 115–139, May 2006.

[51] Z. Li, Y. Gao, and Y. Chen, "Towards a high-speed router-based anomaly/intrusion detection system," http://conferences.sigcomm.org/sigcomm/2005/poster-121.pdf, Aug. 2005.

[52] Y. Gao, Z. Li, and Y. Chen, "A dos resilient flow-level intrusion detection approach for high-speed networks," in *Proc. of the 26th IEEE International Conference on Distributed Computing Systems (ICDCS '06)*, 2006, p. 39.

[53] S. M. Specht and R. B. Lee, "Distributed denial of service: Taxonomies of attacks, tools, and countermeasures," in *Proc. of the ISCA 17th International Conference on Parallel and Distributed Computing Systems (ISCA PDCS'04)*, Sep. 2004, pp. 543–550.

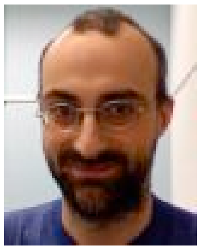
[54] Q. Zhao, J. Xu, and A. Kumar, "Detection of super sources and destinations in high-speed networks: Algorithms, analysis and evaluation," *IEEE Journal on Selected Areas in Communications*, vol. 24, no. 10, pp. 1840–1852, Oct. 2006.

[55] M.-S. Kim, H.-J. Kong, S.-C. Hong, S.-H. Chung, and J. Hong, "A flow-based method for abnormal network traffic detection," in *Proc. of IEEE/IFIP Network Operations and Management Symposium (NOMS'04)*, Apr. 2004, pp. 599–612.

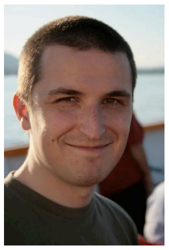
[56] G. Münz and G. Carle, “Real-time analysis of flow data for network attack detection,” in *Proc. of 10th IFIP/IEEE International Symposium on Integrated Network Management (IM’07)*, 2007, pp. 100–108.
[57] Diadem Firewall European Project, http://www.diadem-firewall.org, Jul. 2008.
[58] C. D. A. Lakhina, M. Crovella, “Characterization of network-wide anomalies in traffic flows,” in *Proc. of 4th ACM SIGCOMM conference on Internet measurement (IMC ’04)*, 2004, pp. 201–206.
[59] A. Lakhina, M. Crovella, and C. Diot, “Mining anomalies using traffic feature distributions,” *SIGCOMM Comput. Commun. Rev.*, vol. 35, no. 4, pp. 217–228, 2005.
[60] A. Lakhina, K. Papagiannaki, M. Crovella, C. Diot, E. D. Kolaczyk, and N. Taft, “Structural analysis of network traffic flows,” *SIGMETRICS Perform. Eval. Rev.*, vol. 32, no. 1, pp. 61–72, 2004.
[61] A. Lakhina, M. Crovella, and C. Diot, “Diagnosing network-wide traffic anomalies,” in *Proc. of the 2004 conference on Applications, technologies, architectures, and protocols for computer communications (SIGCOMM ’04)*, no. 4, 2004, pp. 219–230.
[62] Sprint.net, http://www.sprint.net, Jul. 2008.
[63] A. Sperotto, R. Sadre, and A. Pras, “Anomaly characterization in flow-based traffic time series,” in *Proc. of the 8th IEEE International Workshop on IP Operations and Management, IPOM 2008, Samos, Greece*, Sep. 2008, pp. 15–27.
[64] SURFnet, www.surfnet.nl, Jul. 2008.
[65] A. Wagner and B. Plattner, “Entropy based worm and anomaly detection in fast IP networks,” in *Proc. of 14th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprise (WETICE ’05)*, June 2005, pp. 172–177.
[66] C. E. Shannon, “A mathematical theory of communication,” *The Bell System Technical Journal*, vol. 27, no. 3, pp. 379–423, 1948.
[67] C. Gates, J. McNutt, J. Kadane, and M. Kellner, “Scan detection on very large networks using logistic regression modeling,” in *Proc. of 11th IEEE Symposium on Computers and Communications (ISCC’06)*, 2006, pp. 402–408.
[68] M. Stoecklin, J.-Y. L. Boudec, and A. Kind, “A two-layered anomaly detection technique based on multi-modal flow behavior models,” in *Proc. of 9th International Conference on Passive and Active Measurement (PAM’08)*, 2008, pp. 212–221.
[69] T. Dübendorfer and B. Plattner, “Host behaviour based early detection of worm outbreaks in internet backbones,” in *Proc. of the 14th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprise (WETICE’05)*, 2005, pp. 166–171.
[70] T. Dübendorfer, A. Wagner, and B. Plattner, “A framework for real-time worm attack detection and backbone monitoring,” in *Proc.of 1st IEEE International Workshop on Critical Infrastructure Protection (IWCIP’ 05)*, Nov. 2005.
[71] A.Wagner, T. Dübendorfer, B. Plattner, and R. Hiestand, “Experiences with worm propagation simulations,” in *Proc. of 2003 ACM workshop on Rapid malcode (WORM’03)*, 2003, pp. 34–41.
[72] M. Lee, T. Shon, K. Cho, M. Chung, J. Seo, and J. Moon, “An approach for classifying internet worms based on temporal behaviors and packet flows,” in *Proc. of 3rd Int. Conf. on Intelligent Computing (ICIC 2007)*, 2007, pp. 646–655.
[73] C. Zou, W. Gong, and D. Towsley, “Code red worm propagation modeling and analysis,” in *Proc. of 17th USENIX Security Symposium (USENIX Security ’08)*, 2002, pp. 138–147.
[74] F. Dressler, W. Jaegers, and R. German, “Flow-based worm detection using correlated honeypot logs,” in *Proc. of 15th GI/ITG Fachtagung Kommunikation in Verteilten Systemen (KiVS 2007)*, Feb. 2007, pp. 181–186.
[75] M. Collins and M. Reiter, “Hit-list worm detection and bot identification in large networks using protocol graphs,” in *Proc. of 10th International Symposium on Recent Advances in Intrusion Detection (RAID’07)*, 2007, pp. 276–295.
[76] D. H. A. Karasaridis, B. Rexroad, “Wide-scale botnet detection and characterization,” in *Proc.of the first conference on First Workshop on Hot Topics in Understanding Botnets (HotBots’07)*, 2007, pp. 1–8.
[77] C. Livadas, R. Walsh, D. Lapsley, and W. Strayer, “Using machine learning techniques to identify botnet traffic,” in *Proc. of 31st IEEE Conference on Local Computer Networks (LCN’06)*, 2006, pp. 967–974.
[78] G. Gu, R. Perdisci, J. Zhang, and W. Lee, “Botminer: Clustering analysis of network traffic for protocol- and structure-independent botnet detection,” in *Proc. of 17th USENIX Security Symposium (USENIX Security ’08)*, June 2008, pp. 139–154.
[79] Z. Zhu, G. Lu, Y. Chen, Z. Fu, P. Roberts, and K. Han, “Botnet research survey,” in *32nd Annual IEEE International Computer Software and Applications (COMPSAC ’08)*, Aug. 2008, pp. 967–972.
[80] Q. Xiaofeng, H. Jihong, and C. Ming, “Flow-based anti-spam,” in *Proc. of 4th IEEE Workshop on IP Operations and Management (IPOM’04)*, Oct. 2004, pp. 99–103.
[81] Symantec.com, “The state of spam, a monthly report - july 2008,” http://www.symantec.com/, Jul. 2008.
[82] A. Ramachandran and N. Feamster, “Understanding the network-level behavior of spammers,” *SIGCOMM Comput. Commun. Rev.*, vol. 36, no. 4, pp. 291–302, 2006.

Anna Sperotto is a Ph.D. student at the DACS group of the University of Twente, The Netherlands. In 2006 she received her MSc in Computer Science from the Ca’Foscari University, Venice, Italy. Her main topic of interest is Intrusion Detection, Self-Learning and Graph Theory. Currently she is investigating the use of Self-Learning in Intrusion Detection in High-Speed Networks.

Gregor Schaffrath received the diploma in computer science in 2004 from Saarland University in Saarbruecken, Germany. His research interests include network management and security in distributed systems. He currently works for the FG INET group of Technical University Berlin, Germany.



Ramin Sadre is a postdoctoral researcher at the DACS group of the University of Twente, The Netherlands. In 2006 he received a PhD thesis from the same university with the title ”Decomposition-Based Analysis of Queueing Networks”. He is WP7 leader within the EMANICS Network of Excellence. Ramin Sadre has been technical program co-chair of the 3rd International Conference on Autonomous Infrastructure, Management and Security (AIMS 2009).



Cristian Morariu received his Masters Degree from Technical University of Cluj-Napoca, Romania in June 2004 after 4.5 years of studies. His major at the Faculty of Automation and Computer Science, was performed in Computer Science. While holding an ERASMUS scholarship he developed his Master Thesis at the Swiss Federal Institute of Technology (ETHZ), Laboratory of Software Technology. Since September 2004 he is a doctoral student at the University of Zurich, Department of Informatics, Communication Systems Group. His main interests are in the area of IP accounting and distributed architectures for IP traffic analysis.

Aiko Pras is Associate Professor at the Design and Analysis of Communication Systems (DACS) group at the University of Twente (UT), The Netherlands. His research interests include network management technologies, Web services, network measurements and Internet security. He is chairing the IFIP Working Group 6.6 on ”Management of Networks and Distributed Systems”, and is Research Leader in the European Network of Excellence on ”Management of the Internet and Complex Services” (EMANICS). Aiko Pras has been the technical program co-chair of the Ninth IFIP/IEEE Integrated Management Symposium (IM 2005), is Steering Committee member of the IFIP/IEEE NOMS and IM Symposia (NISC), and general co-chair of Manweek 2009.

Burkhard Stiller chairs the Communication Systems Group CSG, Department of Informatics IFI at the University of Zurich UZH since 2004. He holds a Computer Science Diplom and a Ph.D. degree of the University of Karlsruhe, Germany. During his research locations of the Computer Laboratory, University of Cambridge, U.K., the Computer Engineering and Networks Laboratory, ETH Zurich, Switzerland, and the University of Federal Armed Forces, Munich, Germany his main research interests cover, including current CSG topics, charging and accounting of Internet services, economic management, systems with a fully decentralized control (P2P), telecommunication economics, and biometric management systems.