Real-time Analysis of Flow Data for Network Attack Detection

Gerhard Münz, Georg Carle
Computer Networks and Internet
Wilhelm Schickard Institute for Computer Science
University of Tuebingen, Germany
Email: {muenz|carle}@informatik.uni-tuebingen.de

***Abstract*— With the wide deployment of flow monitoring in IP networks, the analysis of the exported flow data has become an important research area. It has been shown that flow data can be used to detect traffic anomalies, DoS attacks, and the propagation of worms. In practice, anomalies and attacks should be detected as fast as possible in order to allow taking appropriate countermeasures. We describe the necessary steps from the raw flow data to the detection result in a systematic way. Furthermore, we present TOPAS, a system and framework for real-time analysis of flow data, that has been developed in order to meet these requirements. Performance measurements and various application examples point out the capabilities and benefits of our approach.**

***Index Terms*— network monitoring, flow analysis, anomaly and attack detection**

I. Introduction

The operation of a computer network is a challenging task, especially if the network is connected to or part of the Internet. One reason is the increasing complexity of the Internet itself, which is characterized by the interconnection of lots of different devices and device types operated in multiple network domains and controlled by separate network management authorities. Even though Internet protocols have been designed to cope with such heterogeneous environments, unpredicted failures and unintended misconfigurations as well as deliberate attacks against the network infrastructure cause frequent problems. On the other hand, a pure best-effort service as provided in the early days of the Internet is neither sufficient for many modern interactive or real-time applications nor acceptable to the more and more discerning users. As a consequence, operators need to monitor and supervise their network permanently in order to guarantee proper functioning and detect and mitigate network problems and quality-of-service degradations rapidly.

As an important prerequisite for time-critical network operation tasks, many modern network devices support monitoring functions that offer monitoring data with low latency. The export of flow data is such a monitoring function that allows retrieving information about the traffic currently observed at a monitoring device, which may be a router or a stand-alone network monitor. A flow is defined as a unidirectional stream of IP packets that share a set of common properties; typically, the IP-five-tuple of protocol, source and destination IP addresses, source and destination ports is used. The exported flow data comprises statistics about the observed flows, such as the number of octets and packets measured within a given time interval. One well-known representative of this technique is the Netflow technology developed by Cisco [1]. In order to provide interoperability among different monitoring device manufacturers, the IETF is currently standardizing the IPFIX (IP Flow Information Export) protocol [2] which can be used for the export of monitoring data.

While flow data is currently mainly evaluated for accounting purposes, it also carries valuable information for traffic analysis. It allows examining the traffic composition and drawing conclusions concerning the originating applications and services. It also enables discovering periodical changes and temporal trends. Furthermore, it can be used to detect and analyze traffic anomalies caused by network problems or illegitimate attack traffic. Especially in the latter case, it is important that the examination of the flow data occurs in a timely manner, enabling the network administrator to evaluate the detection results and to decide on appropriate countermeasures.

For this purpose, we designed and implemented a system called TOPAS (Traffic flOw and Packet Analysis System). TOPAS works as collector for flow data exported via Cisco Netflow and/or IPFIX and provides a framework for user-defined detection modules that simultaneously analyze the received data in real-time. Similarly, packet data exported via PSAMP [3] can be received and analyzed. We currently deploy TOPAS in the demonstrator of *Diadem Firewall*[1] to detect various kinds of denial-of-service (DoS) and distributed denial-of-service (DDoS) attacks. The main contributions of this paper are as follows:

- We describe the generic procedure of analyzing flow data for real-time anomaly and attack detection by decomposing it into three principal steps. These steps represent a generic model for all kinds of detection mechanisms, yet their actual implementations have to be adapted to the deployed detection algorithm (Section II).
- We present the functional properties of TOPAS and describe its system design. Performance measurements show that TOPAS is able to cope with high amounts

[1] *Diadem Firewall* is a European research project and partly funded by the European Commission (FP6 IST-2002-002154).

of flow data as measured in enterprise and university networks or even larger carrier networks (Section III).
- We explain how TOPAS is deployed in Diadem Firewall to detect DoS and DDoS attacks and identify the entry points of attack packets with spoofed source addresses. In addition, we show that TOPAS can also be used for analyzing packet-based monitoring data (Section IV).

With an overview on related work (Section V), we point out that most existing systems are optimized for offline analysis of flow data or offer only near real-time processing capabilities. A conclusion summarizes the paper (Section VI).

II. Real-time Analysis of Flow Data

Flow accounting has become a widely deployed monitoring technique in IP networks. Flow data is composed of flow records that contain statistics about individual flows. A flow is defined as a unidirectional stream of IP packets that are observed at an observation point in the network and that share a set of common properties called *flow keys* in IPFIX terminology [4]. Even though the IP-five-tuple is usually used as flow keys, other distinguishing criteria are also possible. For example, flow aggregates may be monitored instead of individual flows [5]. Cisco's Netflow is the most popular representative of this technology. There are different versions of Netflow. Netflow.v5 is mostly supported by routers in use today. In our work, we focus on the latest version Netflow.v9 [1] and the new IETF standard IPFIX [2] which enable flexible structures for flow records by using a template mechanism. Nevertheless, our considerations are also valid for flow data received via Netflow.v5 or similar protocols.

The analysis of exported flow data for anomaly and attack detection can be decomposed into three principal steps:

1) Flow data is received from the monitoring devices and decoded.
2) The flow data is normalized and preprocessed in order to provide appropriate input to the detection algorithm. Only a subset of the received flow data may be selected for the analysis, such as flows directed to or originating from a specific network, or flows of a particular protocol (e.g. TCP).
3) Finally, a detection algorithm is applied in order to discover network anomalies or attacks.

In general, these steps can be regarded to be independent. Even though the detection algorithm requires input in a specific format, it can be used in combination with different variants of step 2 as long as the correct format is provided.

As we are interested in analyzing flow data in real-time, all the tasks mentioned above are time-critical. Especially those tasks that are applied to each record in the flow data must be simple enough to be performed at the rate of the incoming records. Furthermore, it is necessary to reduce the amount of data at an early stage since high data rates are not only challenging with respect to real-time processing but also with respect to memory requirements. In the following, we elaborate on the three processing steps and explain how they should be designed to be compatible for real-time deployment.

A. Receiving and Decoding Flow Data

Receiving and decoding flow data is the basic functionality of a collector. It is possible to simultaneously receive flow data from multiple monitoring devices. The collector functionality mainly depends on the protocol in use (e.g. Cisco Netflow or IPFIX).

The decoding of Netflow.v5 data is relatively simple as it contains flow records of constant length and format. Decoding Netflow.v9 and IPFIX data is more complex. Here, templates define the record structures using semantic field types called *information elements* in IPFIX terminology [6]. Before exporting flow data with a new template, a monitoring device sends the corresponding template information to the collector. Thus, the collector is capable of decoding and interpreting the received flow data correctly.

In order to optimize for real-time deployment, we can restrict the decoding on the part of flow data that is actually needed for the analysis. On the other hand, it is advantageous to exploit the configuration capabilities of the monitoring devices to adapt the exported data to the receiving capabilities of the collector as well as to the requirements of the detection algorithms. This adaptation can be achieved by

- activating packet filtering and/or sampling,
- deploying appropriate flow keys and/or flow aggregation [5],
- defining appropriate templates that contain the necessary information,
- tuning the flow expiration parameters as suggested in the following subsection.

Usually, such an optimized configuration also saves resources at the monitoring device and in the network since it prevents unneeded measurements and the export of useless or redundant information.

B. Normalization and Preprocessing

Raw flow data usually requires some normalization and preprocessing in order to become appropriate input for detection algorithms. This subsection explains why such normalization and preprocessing is required and how we can achieve this.

First of all, flows can be measured with different flow keys. If flow records with different flow keys are to be analyzed jointly, special care is necessary in order not to bias the analysis results.

Another normalization issue concerns flow timestamps. Flow statistics such as octet and packet counters are typically exported in conjunction with the timestamps of the first and the last counted packet. However, these timestamps do not necessarily indicate the actual beginning and end of a packet stream. They rather depend on the flow expiration and export policy deployed by the monitoring device. Generally, a flow is considered to be terminated if no more packets have been observed for a given timeout. Yet, this assumption does not hold for packet streams showing sporadic traffic only. The monitor may also utilize transport header information about the connection state (e.g. TCP FIN or RST flags) to

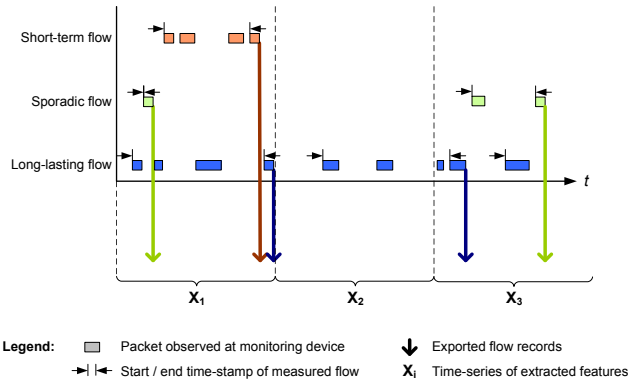

Fig. 1. Relationship between packet, flows, records, and time-series

detect the termination of a flow, however this is not possible for connection-less transport protocols. In order to keep the number of flow states in the monitoring device at a reasonable level, statistics of long-lasting flows are exported at a periodic basis. Timeouts and export intervals are often configurable parameters, i.e. they may vary from monitoring device to monitoring device. As a consequence, observing identical traffic results in different compositions of flow records, depending on the type and configuration of the deployed monitoring device.

A common solution to achieve temporal normalization of flow data is to calculate octet and packet counters for predefined equally spaced time intervals. This results in a multivariate time-series. Figure 1 illustrates the temporal normalization of flow data for short-term, sporadic, and long-lasting flows. As can be seen, the start and end timestamps of a flow record do not necessarily fall into the same time interval used to calculate a time-series value. The question is if such flow records are accounted to only one interval, which causes a certain inaccuracy, or if it can be partitioned to contribute proportionally to all concerned intervals. However, this kind of interpolation can only be a rough estimation of the traffic that was actually observed at the monitoring device. Another problem is that the existence of a flow is unknown to the flow analyzer until it has been reported. In the example shown in Figure 1, no record about the long-lasting flow is exported in the second interval although packets are being observed. Thus, if we wanted this flow to contribute to the time-series value in the second interval, we would have to wait for the delayed record. A practical way to achieve time-series values with acceptable accuracy is to configure the monitoring devices with short timeouts and to choose significantly larger values for the time-series intervals. In this case, a flow record can be accounted to an interval according to its end timestamp or simply according to its arrival time at the collector without causing to much deviation.

While the normalization of flow data is very similar for different kinds of analysis, we need additional preprocessing to adapt to the input requirements of the detection algorithm. As an example, volume-based anomaly detection methods require the total number of octets or packets of all flows or a specific subset of the flows [7]. An additional informative measure is the number of active flows or the number of flows with identical source or destination address as used in [8] and [9] respectively. More sophisticated algorithms rely on entropy measures calculated for specific feature value distributions such as port numbers and/or IP addresses [10], [11]. As already mentioned, normalization and/or preprocessing may be applied to a selection of the entire flow data only in order to restrict the analysis to specific flows or kinds of traffic.

Flow records enter the normalization and preprocessing step more or less at the same rate as they were exported by the monitoring devices, i.e. we have to cope with large amounts of data that arrive in short periods of time. This means that the required computation complexity per flow record has to be low in order to allow real-time deployment. Furthermore, the normalization and preprocessing efforts have to aim at data reduction. With a transformation into time-series, this is usually fulfilled since all data received within one interval is aggregate into a sample of constant size.

C. Detection Algorithm

Different flow-based anomaly and attack detection approaches have been presented in literature. We can classify the most important ones into four categories:

- *Threshold-based detection algorithms* that rely on a priori knowledge such as predefined or adaptive thresholds for specific measures (e.g. [7], [12]).
- *Principal component classifiers (PCC)* that detect anomalies in multivariate time-series [8], [10], [11].
- *Outlier detection algorithms* that determine the similarity of a sample vector to the learned normal behavior [13].
- *Rule learning algorithms* that learn classification rules from training data containing labeled normal and attack data (e.g. [14]).

The computational complexity varies from very simple decisions (thresholds, rules) to more complex distance calculations (outlier detection) and linear transformations (PCC). Nevertheless, all of them are practicable solutions for real-time analysis since we trigger the algorithms with the periodicity of the incoming time-series only.

The result of the detection algorithms is used to raise an alert when an attack or anomaly has been detected. Even though some detection algorithms provide very good results, they all suffer from false positives and false negatives. Hence, deploying several algorithms in parallel and correlating the alerts is an interesting approach to improve the detection rate and to reduce the number of false positives.

III. System Design and Performance Evaluation

Starting from the considerations in Section II, we designed and developed TOPAS as a system and framework that fulfills the specific requirements of real-time flow analysis. In the scope of Diadem Firewall, TOPAS is to provide the functionality of the violation detection facility with the capability to publish detected anomalies and attacks as alert notifications. We decided to encode alerts in XML (Extensible Markup

Language) using the Intrusion Detection Message Exchange Format (IDMEF) [15] which is also supported by many other intrusion detection systems and thus enables interoperability. The alerts serve as input to a policy-based system manager that triggers appropriate response mechanisms such as inserting firewall rules, rate limiting, rerouting etc. We will come back to this application scenario later in Section IV.

The next subsection exposes the key features of the system. Subsequently, we briefly describe the system architecture and provide some information about its implementation. In throughput measurements, TOPAS achieved excellent performance results that we will discuss at the end of this section.

A. Key Features

The key features of TOPAS can be summarized as follows:

- TOPAS enables real-time analysis of exported flow data. In contrast to existing systems, the reception and analysis of new flow data are tightly coupled, i.e. newly received flow data is directly forwarded to a detection module that performs the necessary normalization and preprocessing of the data and executes the detection algorithm.
- TOPAS supports parallel deployment of multiple detection algorithms on the same data. The detection algorithms are implemented in separate detection modules that can be started and stopped independently at arbitrary instances in time without requiring a restart of the whole system.
- TOPAS integrates a notification service that enables detection modules to publish detection results as IDMEF messages to an event system. IDMEF allows integration with other intrusion detection systems (IDS). Furthermore, it eases the comparison and correlation of alerts issued by different detection modules.
- TOPAS facilitates the implementation and integration of new detection algorithms through its modular and object oriented approach. Thus, authors of new detection modules can concentrate on the specific data preprocessing functions and the detection algorithm.

B. Architecture and Implementation

The architecture of our system is depicted in Figure 2. As can be seen, we adopted the functional separation into three steps as described in Section II and realized them in different processes and threads.

A collector process receives the flow data from the monitoring devices supporting both Cisco Netflow.v9 and IPFIX protocol. The received data is stored to a ring buffer located in a memory block that is shared by the collector and the detection modules. A module manager running in a second thread is responsible for synchronization between the collector and the detection modules. It notifies the detection modules about newly arrived data and grants read access for a predefined period of time. After timeout or after all modules have signalized that they have read the data, the module manager frees the corresponding buffer segment and proceeds with the next block of data. Modules that have not read the data before timeout are considered to be overloaded or hanging. The module manager terminates these modules since they risk to destabilize and slow down the whole system. This way it is guaranteed that the collector and the well-behaving detection modules keep synchronized. Optionally, terminated modules can be automatically restarted to give them another try.

The detection modules are running in individual processes with two threads. The first thread (container) listens for notifications from the module manager, reads new flow data from the ring buffer, normalizes and preprocesses it according to the input requirements of the detection algorithm, and buffers the result. The second thread empties the container at regular intervals and triggers the detection algorithm with the new input data. Optionally, the detection results can be sent to an event system, e.g. for dissemination to post-processing modules correlating the alerts of different detection modules. Therefore, we integrated an API that allows publishing IDMEF messages to the publish-subscribe system *xmlBlaster* [16].

Realizing the detection module as separate processes has some performance drawbacks compared to a single process solution with multiple threads, but it also has clear advantages with respect to robustness and stability. As an example, it is possible to dynamically start and stop individual detection modules on demand during operation, and misbehaving detection modules can be easily stopped by killing the corresponding processes.

TOPAS has been implemented as an open framework that enables easy integration of new detection algorithms. The base functionality of the container and the detection algorithm thread are provided as abstract C++ classes. The author of a new detection module needs to implement no more than the virtual methods for the normalization and preprocessing of the flow data and the detection algorithm itself. The collector also offers a recorder function that saves flow data persistently on RAM disk or harddisk. In replay mode, the saved flow data can then be used to run detection modules offline, e.g. for debugging and testing purposes.

C. Performance Measurements

We conducted performance measurements to evaluate the maximum throughput in terms of the number of IPFIX packets and flow records that can be processed. Of course, the achievable performance depends on the computational complexity of the detection algorithm including the normalization and preprocessing of the flow data. Another factor is the characteristic of the IPFIX packet stream. At the monitoring devices, the data export is triggered by expiration and export timeouts, which often results in short bursts of IPFIX packets sent to the collector. As a consequence, the socket receive buffer at the collector has to be large enough to prevent packet losses.

We tested TOPAS at constant IPFIX packet rates with varying numbers of active detection modules. In these tests, we employed detection modules that performed only very simple operations on the flow records since we were mainly interested in the performance of the framework itself, i.e. its capability

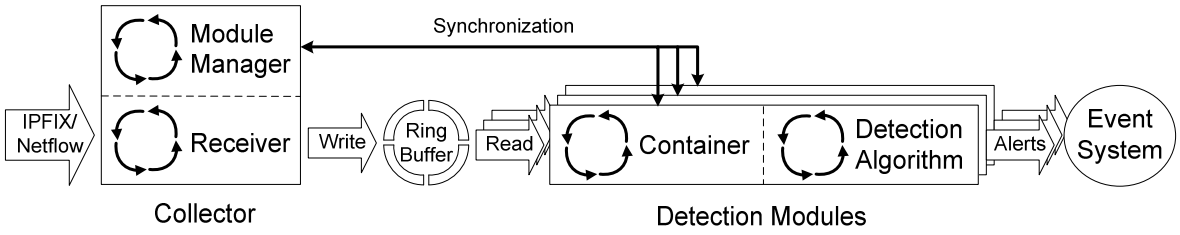

Fig. 2. TOPAS architecture

TABLE I
MEASUREMENT RESULTS

Packet rate	1 module	2 modules	4 modules
5,000/sec	+	+	+
10,000/sec	+	+	-
15,000/sec	+	+	-
20,000/sec	+	-	-
25,000/sec	-	-	-

+ = passed - = failed

to receive flow data and forward it to the detection modules in real-time. The test conditions were as follows:

- IPFIX packets were sent to the collector at different rates. We used repeated sequences of 29 IPFIX packets, where the first packet carried a template and the following 28 packets contained a corresponding set of 10 flow records each. Even though real monitoring traffic would show much more variation over time, this procedure allowed us to achieve reproducible results.
- TOPAS was running on a dual-processor Linux PC[2] with one, two, and four detection modules.

The measurement results are shown in Table I. With four active detection modules, TOPAS was capable of processing rates up to 5,000 IPFIX packets per second, which corresponds to more than 48,000 flow records per second. With only one or two detection modules, 20,000 and 15,000 packets per second could be processed, corresponding to more than 190,000 and 140,000 records per second respectively. At higher packet rates, TOPAS terminated because the detection modules were not able to read the flow data at the same average speed as it was received by the collector.

Afterwards, TOPAS was tested with traces of flow records that were originally exported by a router in the backbone network of the University of Tuebingen. The network of the University of Tuebingen is a class B network divided into more than 150 smaller subnets. It is connected to the Internet at a link speed of 2.4 Gbit per second. The backbone router generates about 1.6 million Netflow.v5 flow records every 15 minutes, corresponding to 1,800 records per second on average. For this test, we used flow records exported by this router in May 2006 and re-exported them as IPFIX packets. The test conditions were as follows:

- We chose a flow trace lasting for 30 minutes and containing 2,854,052 flow records. This corresponds to 1,585 records per second on average.
- The flow records were exported in the order of their end timestamps. The difference between end timestamps was used to determine the relative export times of the records. As a result, the records were exported in burst, similarly to how they were originally exported by the backbone router. Every flow record was sent in an individual IPFIX packet.
- TOPAS was running on the same PC as before. Three detection modules implementing different threshold-based detection schemes were activated.

With the standard socket buffer size (128 kilobytes), we observed significant packet losses (40 percent) due to the burstiness of the IPFIX packet stream. However, after increasing the buffer size to two megabytes, TOPAS passed this test successfully and no packet losses occurred. This shows that TOPAS is capable of performing real-time flow analysis in enterprise and university networks.

In order to estimate how TOPAS would perform in carrier networks, we compare the measured values to the amount of flow records that accumulates in the SWITCH network. The SWITCH network [17] is the Swiss Academic And Research Network carrying about 5% of all Swiss Internet traffic in 2003. In this backbone network, 60 billion flows per hour are measured, corresponding to almost 17,000 flows per second on average. Under the realistic assumption that each Netflow or IPFIX packet contains multiple records, this is still below our measured maximum even though we do not know whether special care is necessary if the data is received in short bursts. A possible countermeasure against bursty Netflow traffic is traffic shaping as applied in Thomas Dübendorfer's UPFrame [18].

IV. APPLICATION

The previous section has shown that TOPAS is a flexible and powerful framework that allows easy integration of various kinds of flow-based detection algorithms. In the following, we present how TOPAS is actually deployed within the demonstrator of Diadem Firewall. The second application example gives an outlook to ongoing research in which we use TOPAS

[2]Dual Intel Xeon 2,8 GHz, 1 GB RAM, 32 MB shared memory

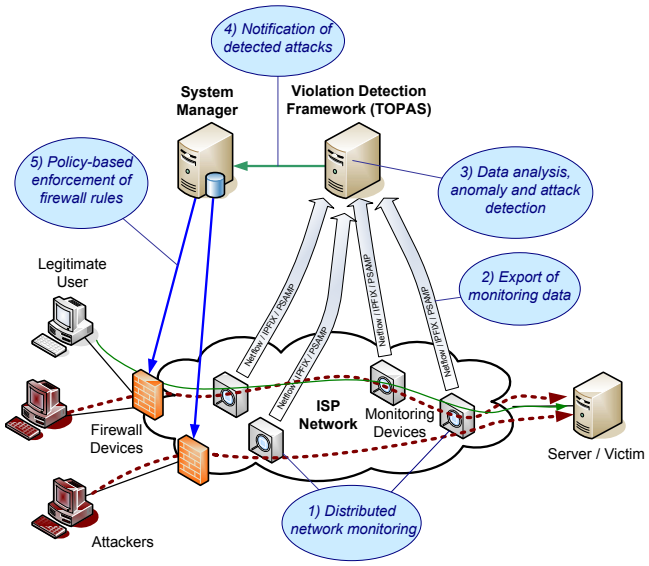

Fig. 3. DDoS attack detection and mitigation with Diadem Firewall

to analyze packet data exported by monitoring devices that support the PSAMP protocol [19].

A. Diadem Firewall

Since January 2004, we have been contributing to the European project Diadem Firewall [20] which is jointly performed by seven European partners under the leadership of France Telecom. The goal of Diadem Firewall is to develop a system that allows Internet service providers (ISPs) to protect their network and their customers from DoS and DDoS attacks. Diadem Firewall relies on distributed network monitoring and real-time analysis of monitoring data for fast detection of ongoing attacks. Based on predefined response policies, the system autonomously takes appropriate counteractive measures such as traceback, rate-limiting, blocking, or rerouting of attack traffic. Firewall devices, that are typically deployed at the edges of the ISP network, enforce the selected countermeasures. Figure 3 depicts the architecture of Diadem Firewall and illustrates its deployment. As shown in the figure, we use TOPAS as framework for the violation detection facility.

One premise of the system design was to allow easy integration into existing networks. Therefore, we decided to utilize flow data exported via Cisco Netflow and/or IPFIX as input to the violation detection. For the sake of advanced detection possibilities, we implemented additional monitoring devices with enhanced monitoring capabilities such as rule-based flow accounting and aggregation, which as been proposed as an extension to the IPFIX standard [21].

Diadem Firewall focuses on two types of attacks that have been exemplarily chosen for demonstrating the system capabilities: SYN flood attacks and web server overloading attacks. In various test scenarios, we showed that both types of attacks can be detected within the first minute after they have started, which allows enforcing a first level of countermeasures very quickly [22]. In the following, we present the detection modules that are used to detect these attacks.

SYN Flood Detection Module: During a SYN flood attack, attackers are sending a large number of TCP connection requests (SYN packets) to a single open port at the victim host. These packets have spoofed source addresses, so that the returned SYN/ACK packets do not reach any valid connection endpoint and thus remain unanswered in most cases. As a result, the victim host has to cope with many half-open TCP connections that are only released after a timeout. Hosts without specific protection mechanisms can only handle a limited number of half-open TCP connections, and if this limit is reached, new TCP connection requests cannot be answered any more.

In Diadem Firewall, SYN floods are detected with the SYN-dog mechanism by Wang et al. [23], which is implemented in a detection module. SYN-dog compares the numbers of SYN and SYN/ACK packets that a host receives and returns respectively. Under normal conditions, the two numbers should be balanced since every SYN packet is answered by a SYN/ACK packet, apart from some SYN packets that are accidentally directed to a closed port. Consequently, a high number of unanswered SYN packets is an indication of an ongoing SYN flood attack. We use our flexible monitoring probe Vermont [24] to measure the number of SYN and SYN/ACK packets and export corresponding counters using the IPFIX protocol.

Traceback Module: The purpose of traceback is to find the ingress points where attack packets with spoofed source addresses, as used in the SYN flood attack, enter the ISP network. For Diadem Firewall, a non-intrusive traceback mechanism has been developed that identifies these ingress points with the help of regular flow records measured and exported at the ingress points of the network. Under normal conditions, the traceback module is in learning mode and is building a white-list for each ingress point containing the most recent source IP addresses observed in the incoming traffic. After an attack with spoofed sources has been detected by another detection module (e.g. the SYN flood detection module), the traceback module switches to traceback mode and determines the number of unknown source IP addresses for each ingress points. The ingress points of the attack traffic can be identified by a high number of unknown source IP addresses observed in very short time.

Web Server Overloading Detection Module: Within Diadem Firewall, Olivier Paul developed a method to infer the nature of HTTP requests from passive traffic measurements based on probabilistic models [25]. The resulting information is used to trace the user behavior in an HTTP session and detect anomalous or aggressive request patterns using EWMA (Exponentially Weighted Moving Average) change detection algorithm. The inference mechanism as well as the detection algorithm have been implemented in a third detection module [26]. Evaluations showed that the detection method is fast and accurate in case of focused DoS attacks requesting a small number of objects on the web server [27].

B. Analysis of Packet Data

While flow analysis is very useful to detect anomalous and suspicious traffic, most detection methods suffer from a non-negligible false-positive rate. Additional information about a small number of individual packets selected from a given flow can be very helpful to classify the flow as harmful or benign. As an example, payload analysis has already been successfully deployed for classifying flows by Karagiannis et al. who used this method to verify a flow-based classification method [28].

As already mentioned in Section II, IPFIX makes use of a flexible template mechanism that enables variable record formats. Furthermore, the usage of IPFIX is not bound to the export of flow data since new semantic field types can be easily defined. Hence, IPFIX provides a very generic and adaptable transport mechanism for any kind of monitoring information. This was a strong argument for the PSAMP (Packet Sampling) working group at the IETF to adopt the IPFIX protocol for exporting packet-based monitoring data such as header fields and parts of the payload [19]. The packet selection at the monitoring device can be deterministic or non-deterministic, applying filters and probabilistic and non-probabilistic sampling [3].

Supporting IPFIX, TOPAS is inherently capable of receiving and processing PSAMP records as well. This led us to the idea of implementing a signature detection module that searches packet data for known attack and worm patterns. However, instead of writing a new signature detection engine, we utilize the popular open-source IDS Snort [29] for this purpose and integrate it with help of a wrapper detection module. We are currently investigating the potentials of analyzing packet data as an enhancement to flow-based detection methods.

V. Related Work

In this section, we give an overview on existing flow analysis systems that collect and analyze Netflow and/or IPFIX data. Most of them have not been designed for real-time analysis, instead they apply analysis functions to flow data that has been previously saved in databases or files. Hence, these systems provide valuable insight in past network traffic, yet they are not suitable for real-time detection of anomalies and attacks. We begin our overview with database-oriented systems, proceed with systems using files, and terminate with systems that process incoming flow data immediately.

A system that stores received flow data in a database for later analysis is *Netflow Monitor*[3] [30]. Configurable aggregation schemes can be applied in order to reduce the amount of monitoring data in the database. Netflow Monitor allows displaying the flow data in different kinds of charts and diagrams. In the scope of the History project [31], [32], we developed a tool that follows a similar approach and supports queries of flow data stored in multiple distributed databases. Such database-oriented systems allow examining historic as well as recently received monitoring data manually and offline. However, these systems are not optimized for real-time analysis since the usage of a database as intermediate storage with concurrent write and read accesses introduces non-negligible delay.

Ntop [33] stores flow data in a round-robin database (RRD) that saves the data for a limited period of time before replacing it with newly received data. This way, the size of the database is not increasing over time. Ntop is conceived for examining recently received flow data within a sliding time window. However, as it also makes use of a database, the same restriction as for the other database-oriented systems apply also to this system.

The *SiLK* suite [34]–[36] is a set of command line tools for collecting, storing, and analyzing large amounts of Netflow data. SiLK stores flow data in hourly files using a compressed binary file format. Organizing the files in a hierarchical directory tree facilitates the access to the files. The analysis tools read the flow data from the saved files and generate various types of statistics. SiLK is optimized for archiving large amounts of flow data in an efficient way for offline analysis. However, it has not been designed for real-time flow analysis.

Mark Fullmer's *Flow-tools* [37] are a similar tool suite with the main difference that a simpler binary file format without sophisticated compression is used. Files with 15 minutes of flow data (default setting) are stored in a single directory. An optional round-robin mechanism ensures that the directory size does not exceed a certain volume or number of files by replacing old files with new ones. Instead of storing the received monitoring data into files, it can also be sent to another process connected via a TCP socket. Flow-tools can be used in conjunction with *FlowScan* [38] which produces graph images with a continuous, near real-time view of the network traffic.

A system for near real-time analysis of flow data is *MINDS* (Minnesota Intrusion Detection System) [9]. A collector receives Netflow data and saves it in files covering consecutive time windows of 10 minutes each. An analyzer module processes the saved files and performs known attack detection and anomaly detection algorithms. Because of the batch mode processing of the files, it may last up to several minutes until an attack or anomaly is detected, which is to late for fast reaction.

An example of a system that was conceived for actual real-time flow analysis is *Panoptis* [7]. Panoptis applies a simple threshold-based anomaly detection algorithms to the number of octets, packets, and flows observed in time intervals of constant length in order to detect DoS and DDoS attacks. It consists of a Netflow collector that delegates the received flow data to an analyzer module. The system *Open-Eye* [12] follows the same concept but applies the threshold algorithm to different metrics. Both Panoptis and Open-Eye implement only a single detection algorithm and do not provide the possibility to run multiple detection modules simultaneously.

Real-time analysis of UDP payload can be accomplished with *UDFrame* [18]. Thomas Dübendorfer and Arno Wagner

[3] A commercial version is available as *Caligare Flow Inspector* (http://www.caligare.com/netflow/).

developed this framework in parallel to our work and used it to process Netflow.v5 data. Although developed independently and without knowledge of the other, the architecture of UDFrame and TOPAS resemble each other in many aspects, which seems to be a strong indication that both our approaches are sound. UDFrame supports multiple plugins that may analyze the received flow data in parallel. The data exchange between the collector and the plugins is based on a ring buffer of shared memory segments. Nevertheless, UDFrame misses a couple of features that TOPAS offers, such as the support of Netflow.v9 and IPFIX, including the necessary template management, and the integrated event-based notification service. Furthermore, UDFrame does not offer the possibility to record and replay flow data, which is a very useful feature for testing and debugging detection modules offline.

VI. Conclusion

Flow data as it is exported by many modern network devices contains valuable information that allows discovering different kinds of network anomalies and attacks. This paper focused on real-time analysis of flow data and fast detection of anomalies and attacks, so the detection results can be used to timely trigger counteractive actions. We described the necessary steps of data reception and decoding, normalization and preprocessing, and the application of the detection algorithm. Then, we presented TOPAS, a system and framework for real-time analysis of Netflow, IPFIX, and PSAMP data. TOPAS enables easy integration of new detection algorithms through its modular approach. Our performance measurements show that TOPAS is able to cope with large amounts of monitoring data on a standard PC. TOPAS is publicly available as an open-source package [20].

TOPAS is deployed in the demonstrator of Diadem Firewall for detecting SYN flood attacks and web server overloading attacks and for tracing back attack packets with spoofed source addresses. We state that the detection opportunities can be increased if monitoring and detection are well adjusted. An example is the export of counters for SYN and SYN/ACK packets as required by the SYN flood detection module. As the IPFIX protocol has been chosen for PSAMP export, TOPAS can also be used for real-time inspection of packet data. Our current research interests concentrate on developing new detection schemes that make use of both flow-based and packet-based network monitoring in order to reduce the number of false positives.

Acknowledgments

We gratefully acknowledge support from the European project Diadem Firewall and thank our partners for their valuable feedback and advice. Special thank goes to Lothar Braun who contributed to the design of TOPAS and was doing most of the implementation work.

References

[1] B. Claise, "Cisco Systems NetFlow Services Export Version 9," RFC 3954 (Informational), Oct. 2004. [Online]. Available: http://www.ietf.org/rfc/rfc3954.txt

[2] B. Claise, S. Bryant, G. Sadasivan, S. Leinen, and T. Dietz, "IPFIX Protocol Specifications," Internet-Draft, draft-ietf-ipfix-protocol-24, Nov. 2006.

[3] N. Duffield, D. Chiou, B. Claise, A. Greenberg, M. Grossglauser, P. Marimuthu, J. Rexford, and G. Sadasivan, "A Framework for Packet Selection and Reporting," Internet-Draft, work in progress, draft-ietf-psamp-framework-10, Jan. 2005.

[4] J. Quittek, T. Zseby, B. Claise, and S. Zander, "Requirements for IP Flow Information Export (IPFIX)," RFC 3917 (Informational), Oct. 2004. [Online]. Available: http://www.ietf.org/rfc/rfc3917.txt

[5] F. Dressler and G. Münz, "Flexible flow aggregation for adaptive network monitoring," in *Proc. of IEEE LCN Workshop on Network Measurements 2006*, Tampa, Florida, USA, Nov. 2006.

[6] J. Quittek, S. Bryant, B. Claise, and J. Meyer, "Information Model for IP Flow Information Export," Internet-Draft, work in progress, draft-ietf-ipfix-info-14.txt, Oct. 2006.

[7] C. Kotsokalis, D. Kalogeras, and B. Maglaris, "Router-based Detection of DoS and DDoS Attacks," in *Proc. of HP Openview University Association (HP-OVUA) 8th Annual Workshop*, Berlin, Germany, June 2001.

[8] A. Lakhina, M. Crovella, and C. Diot, "Characterization of Network-Wide Anomalies in Traffic Flows," in *Proc. of 4th ACM SIGCOMM Conference on Internet Measurement*. Taormina, Sicily, Italy: ACM Press, Oct. 2004, pp. 201–206.

[9] L. Ertöz, E. Eilertson, A. Lazarevic, P.-N. Tan, P. Dokas, V. Kumar, and J. Srivastava, "Detection of novel network attacks using data mining," in *Proc. of Workshop on Data Mining for Computer Security, to be held in conjuction with IEEE International Conference on Data Mining*, Melbourne, FL, USA, Nov. 2003.

[10] A. Lakhina, M. Crovella, and C. Diot, "Mining Anomalies Using Traffic Feature Distributions," in *Proc. of ACM SIGCOMM Conference*, Philadelphia, PA, USA, Aug. 2005, pp. 217–228.

[11] J. Terrell, K. Jeffay, F. D. Smith, L. Zhang, H. Shen, Z. Zhu, and A. Nobel, "Multivariate SVD Analyses For Network Anomaly Detection," in *Proc. of ACM SIGCOMM Conference, Poster Session*, Philadelphia, PA, USA, Aug. 2005.

[12] G. Androulidakis, V. Chatzigiannakis, M. Grammatikou, and F. Stamatelopoulos, "Network Flow-Based Anomaly Detection of DDoS Attacks," in *Proc. of Trans-European Research and Education Networking Association (TERENA) 2004*, Rhodes, Greece, June 2004.

[13] A. Lazarevic, L. Ertöz, V. Kumar, A. Ozgur, and J. Srivastava, "A comparative study of anomaly detection schemes in network intrusion detection," in *Proc. of 3rd SIAM International Conference on Data Mining*, May 2003.

[14] D. Barbará, J. C. S. Jajodia, L. Popyack, and N. Wu, "ADAM: Detecting intrusions by data mining," in *Proc. of IEEE Workshop on Information Assurance and Security*, West Point, NY, USA, June 2001, pp. 11–16.

[15] H. Debar, D. Curry, and B. Feinstein, "The Intrusion Detection Message Exchange Format," Internet-Draft, work in progress, draft-ietf-idwg-idmef-xml-16, Mar. 2006.

[16] M. Ruff, "White Paper XmlBlaster," Mar. 2000. [Online]. Available: http://www.xmlblaster.org/xmlBlaster/doc/whitepaper/whitepaper.html

[17] SWITCH Homepage, http://www.switch.ch/, 2006.

[18] 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 2005)*, Darmstadt, Germany, Nov. 2005.

[19] B. Claise, J. Quittek, and A. Johnson, "Packet Sampling (PSAMP) Protocol Specifications," Internet-Draft, work in progress, draft-ietf-psamp-protocol-07, Oct. 2006.

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

[21] F. Dressler, C. Sommer, and G. Münz, "IPFIX Aggregation," Internet-Draft, work in progress, draft-dressler-ipfix-aggregation-03.txt, June 2006.

[22] A. Fessi, S. Yusuf, Y. Carlinet, O. Paul, P. Sagmeister, J. van Lunteren, V. Thing, M. Sloman, D. Thomas, D. Gabrijelcic, P. Tobis, G. Münz, D. Haage, R. Sasnauskas, and K. Dragicevic, "Evaluation Report, DIADEM Firewall Deliverable D14," Sept. 2006.

[23] H. Wang, D. Zhang, and K. G. Shin, "SYN-dog: Sniffing SYN Flooding Sources," in *Proc. of 22nd International Conference on Distributed Computing Systems (ICDCS'02)*, Vienna, Austria, July 2002.

[24] R. T. Lampert, C. Sommer, G. Münz, and F. Dressler, "Vermont - A Versatile Monitoring Toolkit for IPFIX and PSAMP," in *Proc. of*

IEEE/IST Workshop on Monitoring, Attack Detection and Mitigation (MonAM 2006), Tuebingen, Germany, Sept. 2006.

[25] O. Paul and J. E. Kiba, "RequIn, a tool for fast web traffic inference," in *48th annual IEEE Global Telecommunications Conference (GLOBECOM 2005)*, Saint Louis, MO, USA, Nov./Dec. 2005.

[26] G. Münz, A. Fessi, G. Carle, O. Paul, D. Gabrijelcic, Y. Carlinet, S. Yusuf, M. Sloman, V. Thing, J. van Lunteren, P. Sagmeister, and G. Dittmann, "Diadem Firewall: Web Server Overload Attack Detection and Response," Bordeaux, France, Dec. 2005.

[27] O. Paul, "Improving web servers focused DoS attacks detection," in *Proc. of IEEE/IST Workshop on Monitoring, Attack Detection and Mitigation (MonAM 2006)*, Tuebingen, Germany, Sept. 2006.

[28] T. Karagiannis, K. Papagiannaki, and M. Faloutsos, "BLINC: Multilevel Traffic Classification in the Dark," in *Proc. of Conference of the Special Interest Group on Data Communication (SIGCOMM'05)*, Philadelphia, Pennsylvania, USA, Aug. 2005, pp. 229–240.

[29] M. Roesch, "Snort: Lightweight Intrusion Detection for Networks," in *Proc. of 13th USENIX Conference on System Administration*. USENIX Association, Nov. 1999, pp. 229–238.

[30] Netflow Monitor Homepage, http://netflow.cesnet.cz/, 2006.

[31] History Project Homepage, http://www.history-project.net/, 2006.

[32] F. Dressler and G. Carle, "History - high speed network monitoring and analysis," in *Proc of 24th IEEE Conference on Computer Communications (IEEE INFOCOM 2005), Poster Session*, Mar. 2005.

[33] Ntop Homepage, http://www.ntop.org/, 2006.

[34] J. McHugh, "Sets, Bags, and Rock and Roll: Analyzing Large Data Sets of Network Data," in *Proc. of European Symposium on Research in Computer Security 2004 (ESORICS 04)*, Sophia Antipolis, France, Sept. 2004.

[35] C. Gates, M. Collins, M. Duggan, A. Kompanek, and M. Thomas, "More Netflow Tools: For Performance and Security," in *Proc. of Large Installation System Administration Conference (LISA) 2004*, Atlanta, GA, Nov. 2004.

[36] C. Gates and D. Becknel, "Host Anomalies from Network Data," in *Proc. of IEEE Systems, Man and Cybernetics Information Assurance Workshop*, West Point, NY, June 2005.

[37] M. Fullmer and S. Romig, "The OSU Flow-tools Package and Cisco NetFlow Logs," in *Proc. of 14th USENIX Conference on System Administration (LISA 2000)*, New Orleans, Louisiana, USA, Dec. 2000, pp. 291–304.

[38] FlowScan Homepage, http://www.caida.org/tools/utilities/flowscan/, 2006.