FIXIDS: A High-Speed Signature-based Flow Intrusion Detection System

Felix Erlacher and Falko Dressler
Heinz Nixdorf Institute and Department of Computer Science
Paderborn University, Germany
{erlacher,dressler}@ccs-labs.org

***Abstract*—Signature-based Network Intrusion Detection Systems (NIDS) are the state-of-the-art when it comes to precise attack detection and intrusion prevention. However, they experience critical performance problems in modern high-speed networks. At the same time, flow-based network monitoring has been investigated for high data rates. In the last years, such flow-monitoring went beyond collecting statistical information about network connections and more recent techniques are able to include selected samples of the payload of these flows. Most recently, we extended this concept to HTTP flows. We now go one step further and combine IPFIX-based flow monitoring with NIDS. We developed IPFIX-based Signature-based Intrusion Detection System (FIXIDS), a system that exploits the recently introduced HTTP related flow Information Elements (IEs) to do signature-based flow intrusion detection in high-speed networks on commodity hardware. FIXIDS makes use of HTTP intrusion signatures from the widely used Snort NIDS and applies them to incoming IPFIX Flows. In the experimental evaluation, we are able to show a performance gain of a factor of three compared to Snort while maintaining the same detection ratio.**

I. Introduction

Our modern Internet is facing an increasing number of threats focusing on the most recent Internet technologies [1]. In this scope, Network Intrusion Detection Systems (NIDS) have been an important tool for network operators already since more than a decade to detect and defend against attacks but also to enforce usage policies to avoid internal misuse [2]. According to established taxonomies (e.g., [3]), intrusion detection systems can be categorized according to the used detection method: Anomaly-based NIDS use behavior-based techniques by defining a model of normal network behavior and then detecting deviations to this model [4]. Knowledge-based systems, on the other hand, use a precise definition of the attack and match incoming traffic against this definition. The most widespread variants of knowledge-based systems are signature or rule-based NIDS. We concentrate on signature-based systems such as Snort [5] or Bro [6], which allow a very detailed description of attacks using Deep Packet Inspection (DPI) methods. They, therefore, offer a very high detection rate at the cost of comparably low performance and, thus, are limited to lower packet throughput.

In signature-based NIDS, a detection engine applies a rule-set to all received data. The majority of state-of-the-art Snort rules contain patterns that are matched against the payload of the received packets. These patterns range from selected bytes to complex Regular Expressions (RegExes) matching not only individual packets but payload in a flow of packets. The performance of a signature-based NIDS mainly depends on the number of rules [7]. Thus, in practical applications, the rule-set needs to be adapted to the domain-specific use case. Nevertheless, a high number of rules remain, and further reducing the number of rules would elevate the risk of not detecting possible intrusions.

A different approach to network intrusion detection is to exploit Flow-based traffic data [8]–[10]. Here a Flow denotes a set of elements containing aggregated information of a number of packets sharing the same properties. Typically, these properties are the same source/destination addresses, the same source/destination ports, and the same protocol. We capitalize the word Flow to emphasize the distinction from other meanings of the word flow. Compared to the standard DPI approach, the analysis on Flow-based traffic data requires an additional step in which the data is aggregated into so-called Flow records with the advantage of significantly reducing the data to be analyzed. Flow monitoring has become quite popular in a wide range of applications due to its ability to work in high-speed networks. The Internet Protocol Flow Information Export (IPFIX) protocol [11], [12] is the primary standard, which is used to aggregate packet information into Flow records for further post-processing. Flow records can be adapted to the application scenario by choosing appropriate IPFIX Information Element (IE) fields.

Given the fact that Hypertext Transfer Protocol (HTTP) has become the dominant protocol in the Internet [13], [14], we recently introduced IPFIX IEs to extend the concept of IP flows to HTTP related information [15]. As a result, a single HTTP dialog (request and response) can now be aggregated and exported as one IPFIX record. Such a Flow record contains, in addition to IP and statistical information, the most important HTTP header information as well as a configurable amount of the beginning of the HTTP payload. These HTTP related IEs are by now registered and standardized with the Internet Assigned Numbers Authority (IANA).[1] Commercial network appliance manufacturers have by now started to include HTTP IEs into their set of exportable Flow fields. Examples are

978-1-5386-3416-5/18/$31.00 © 2018 IEEE

[1] https://www.iana.org/assignments/ipfix/ipfix.xhtml

Citrix,[2] Sonicwall,[3] and Ntop.[4] The format of the exported HTTP data is basically the same, but they are not (yet) using the standardized HTTP IEs but enterprise specific IEs.

In this paper, we go one step beyond and present a novel way of Flow-based intrusion detection that we named IPFIX-based Signature-based Intrusion Detection System (FIXIDS). We use HTTP related Snort signatures and apply them to IPFIX Flows containing HTTP information. To the best of our knowledge, this is the first time that payload-based IEs are directly exploited for signature-based intrusion detection. By using Snort signatures, we ensure that community validated signatures are available for the developed system. The usage of Flow data guarantees high performance because less data has to be analyzed when searching for signatures.

FIXIDS is not intended as a replacement for Snort. We aim for high data rate scenarios where network traffic is collected in form of IPFIX Flows, whereas Snort applies signatures to all individual received packets. Here, FIXIDS analyzes HTTP signatures substantially faster than Snort while showing the same detection rate. Snort is used for the remaining signatures, showing a significant speed-up because of a lower number of rules.

In comparison to DPI, intrusion detection on Flows offers an additional advantage as it allows to distribute the task of fetching packets from the wire (and the aggregation into Flow records) and the task of analyzing the Flows. Thus, parallelization is inherently supported. This is not possible with DPI-based intrusion detection where both tasks have to be done in a tightly coupled fashion. One of the key contributions is the idea to use readily available HTTP elements in IPFIX Flows: Many modern network appliances already support the export of Flow data. This is an inherent advantage over a legacy Intrusion Detection System (IDS). Snort, in comparison, needs to preprocess and parse the HTTP part of the data before analysis. So only very little effort is needed to add our signature-based Flow intrusion detection appliance: FIXIDS is able to run on commodity hardware easily supporting line rates of 10 Gbit/sec.

In the evaluation, we confirm that by using aggregated Flow data the performance of signature-based intrusion detection can be raised significantly. Overall, our Flow-based solution has also an important impact on privacy concerns in network monitoring as much less data compared to DPI-based intrusion detection is needed for analysis [16], [17].

II. Related Work

In this section, we briefly outline the main performance bottlenecks of intrusion detection as well as recent approaches to improve their data throughput. In the first part, we cover signature-based DPI intrusion detection and in the second part we discuss Flow-based intrusion detection.

[2]https://www.citrix.com/products/netscaler-adc/netscaler-data-sheet.html

[3]https://www.sonicwall.com/en-us/products/firewalls/management-and-reporting/global-management-system

[4]http://www.ntop.org/products/netflow/nbox/

The biggest performance bottleneck of signature-based NIDS is the signature pattern matching process [7]. One option is to move this operation to a Graphics Processing Unit (GPU) [18]. It has been shown that the processing throughput can be increased by a factor of two. Other approaches improve this operation by parallelization [19]. For example, a Field Programmable Gate Array (FPGA)-based network interface has been used to distribute the incoming packets to multiple parallel analysis threads [20]. Similarly, the distribution of the traffic on multiple intrusion detection instances by using a system that monitors the load of the single instances has been proposed [9]. This approach should guarantee that no instance is overloaded nor idle. Another concept is to reduce the data that such a system has to analyze. Various methods that intelligently filter network traffic while maintaining a high detection rate have been explored [10], [21], [22].

Analysis on Flow data obviously implies an additional processing step where network packets are aggregated into Flow records. Also this stage has been subject of several improvements such as to use dedicated hardware for packet aggregation [23] or to exploit the hardware of modern high-speed Network Interface Controllers (NICs) [24]. Overall, Flow-based intrusion detection systems promise to have a better throughput because they inherently handle a considerably lower amount of data than DPI-based systems. Until now, they primarily relied on packet-header information – typically employing anomaly-based detection techniques. Thus, it is only possible to detect a subset of all possible network attacks [25]. Prominent examples are Distributed Denial of Service (DDOS) attacks [26], scans, and Internet worms [27].

Because of the lack of payload-based IEs, there are only very few examples of Flow-based NIDS that use knowledge-based methods. For example, honey-pot logs provide attack statistics that can be used to detect these attacks in IPFIX Flow IEs [28]. More recently, a system called SSHCure has been presented that detects SSH intrusions by identifying the different phases of such an attack in the statistical properties of Flows [29], [30]. Because they can only rely on header information, all these systems offer rather static definitions of attacks. To the best of our knowledge, there is no Flow-based NIDS that supports user-definable signatures.

III. IPFIX-based Signature-based Intrusion Detection System

In the following, we introduce and explain our novel Flow-based NIDS. The goal is to be able to perform signature-based intrusion detection at high-speed while still detecting a high number of events. By using IPFIX Flows, we can keep the amount of data to be analyzed relatively low while the inclusion of HTTP IEs, in combination with a signature-based detection method, allows us to very precisely detect specific attacks.

A. Rules/Signatures

We decided to use the same rule syntax as Snort because this allows us to use the same signatures without conversion.

TABLE I
SNORT CONTENT MODIFIERS AND THEIR CORRESPONDIG IPFIX IE

Content modifier		HTTP IE	IANA IE ID
http_method	→	httpRequestMethod	459
http_uri	→	httpRequestTarget	461
http_raw_uri	→	httpRequestTarget	461
http_stat_code	→	httpStatusCode	457
http_stat_msg	→	httpReasonPhrase	470

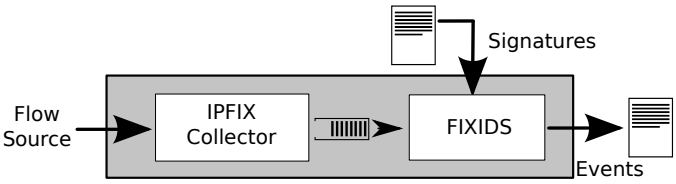

Fig. 1. Configuration of FIXIDS in Vermont

Snort is the most widespread signature-based NIDS and has the largest source of signatures. We do not support all options that are possible within Snort signatures. This would not be viable as many rule options rely on the whole packet payload and in FIXIDS we only have the HTTP part at hand. Options that are not supported are ignored during the parsing process. Among other capabilities Snort searches for content patterns in the payload of the bypassing traffic. To be able to narrow the search space, Snort offers the option to apply so called *content modifiers* to the pattern search. The idea is to restrict the search of content patterns to certain payload fields. The biggest part of the content modifiers in Snort allows to restrict the pattern search to HTTP related fields. We exploit this by using Snort rules with HTTP content modifiers and apply them to the corresponding HTTP related IEs. Table I shows the currently supported content modifiers and the corresponding HTTP related IPFIX IE together with the IANA IE ID.

FIXIDS also accepts the *nocase* modifier which enables case insensitive text search for the corresponding content pattern. The content patterns can be text and binary data (as hexadecimal numbers). Regular expressions are not supported in the current version of FIXIDS. Of course, it is also possible to include own hand-crafted rules which can be customized to a certain application scenario.

B. Implementation

FIXIDS is implemented as part of the Open Source network monitoring toolkit Vermont and is freely available[5]. Vermont is written in C and C++ and its functionality is divided among different modules, each having its own purpose. The FIXIDS process is implemented in an own module, accepting as input IPFIX Flows. A minimal working configuration is shown in Figure 1. An external IPFIX exporter sends IPFIX Flows containing HTTP IEs to Vermont (currently, TCP, SCTP, UDP, and DTLS over UDP and SCTP are supported transport protocols). In Vermont, the IPFIX Collector module receives the Flows and hands them over to the FIXIDS module via a Flow buffer. At startup, the FIXIDS module parses all signature and options from the rules file given in the configuration. At run-time it compares the IEs of every incoming Flow to the corresponding signature patterns of every rule. To compare the string patterns of a rule to the (utf encoded) content of the HTTP IEs, FIXIDS uses the *strstr()* (and *strcasestr()* for case insensitive search) C string-compare function. This function is written in assembler and makes direct usage of CPU registers and thus outperforms other C and C++ string-compare functions. A rule might contain multiple patterns to compare (e.g., a "GET" in the httpRequestMethod IE and multiple patterns in the httpRequestTarget IE). If one of the matches fails, the execution of the remaining pattern matches of the current rule is aborted. If all patterns match, an alarm is triggered and the event information is written to the alert file given in the configuration.

The FIXIDS module can also be configured to pass the analyzed IPFIX Flows to other modules in Vermont or to export them to the next IPFIX collector. Vermont can also be configured to read network packets from a NIC or a captured network trace (pcap file) and aggregate the packets to IPFIX Flows before handing these Flows to the FIXIDS module.

IV. EVALUATION

We divide the evaluation into two parts. For both parts we rely on real-world traffic traces. The first part showcases the functionality of FIXIDS and the second part validates its network data throughput performance. As far as possible, we provide configuration files and other resources to reproduce the results on the author's homepage.[6]

A. Experiment setup

To represent typical HTTP traffic, we use as base traffic a trace that was created by capturing the traffic going through an HTTP proxy used by a scientific work group for one week (from now on called proxy trace). The proxy trace contains only HTTP traffic and consists of 2.2 million packets with an average packet size of 825 B.

We downloaded and used the most current Snort rules from three sources:[7]

- All Snort rules (snapshot 2990) as provided to Snort.org subscribers;
- the community rule-set from Snort.org; and
- all rules from the Emerging Threats rule-set.[8]

We only employ rules that make use of one of the HTTP content modifiers from Table I. To make sure that rules are applied only in the specific cases they are written for we exclude all rules that contain options that are not supported by FIXIDS. The resulting ruleset consists of 1514 rules.

Because of its widespread usage we choose the Snort NIDS to generate a baseline both in functionality and performance to compare against. We use Snort to produce the ground truth

[5]https://github.com/felixe/ccsVermont

[6]http://www.ccs-labs.org/∼erlacher/resources

[7]as of May 9th 2017

[8]http://doc.emergingthreats.net/bin/view/Main/AllRulesets

of events that are present in the test traces and compare the results with the detected events of FIXIDS. In this work it is not our goal to investigate if the events produced by Snort are false-positives.

For the tests in this work we used Snort version 2.9.9.0 in IDS mode, with the default snort.conf configuration file which is shipped with the installation archive. The only relevant changes made to the Snort configuration are the following: We increased the queue sizes of the detection engine (max_queue_events to 1000, max_queue to 1000, log to 1000, and the max_queued_bytes for the stream5_tcp preprocessor to 1.5 MB). This was necessary because if the number of events per packet/message exceeds the queue size, they are not reported anymore by Snort. The queue changes were necessary to be able to realistically compare the outcome of the two tested NIDS as FIXIDS reports always all events. To avoid Snort skipping packets with checksum errors, we turned this behavior off (*-k none* switch).

In the evaluation part, when talking about FIXIDS, we refer to the Vermont configuration with the FIXIDS module depicted in Figure 1. When using Vermont as an IPFIX Flow exporter one has to configure the length of the HTTP Uri that is going to be exported. An examination of the URI lengths in the proxy trace showed the following results: average length is 99.8 B, median length is 55 B, maximum length is 2913 B, and the 85% percentile is 143 B. This matches with the results of a similar investigation on HTTP header lengths done by Google and presented in the spdy whitepaper[9]. According to this results, we configured Vermont to export the first 150 B of the HTTP URI. This also follows the findings in [15], [22] that the most relevant data for intrusion detection is found at the beginning of a Protocol Data Unit (PDU).

In order to show that the FIXIDS system also works with third-party IPFIX exporters, we also conducted experiments using Ntop's network probe Nprobe (Version 8.1) [31]. It is available as software or incorporated on dedicated hardware under the name Nbox.[10] With its HTTP plugin it also exports HTTP related information in IPFIX Flows but uses proprietary enterprise specific IEs. For our experiments, we thus had to modify FIXIDS to also accept the enterprise specific HTTP related IEs of Nprobe.

For all tests we used the following setup: For replaying the network traces we used a workstation with an Intel i5-4440 CPU and 32 GB RAM running Linux 4.4.0. The receiving workstation doing the IPFIX aggregation and exporting (and in one experiment also the FIXIDS intrusion detection) has the same hardware configuration. Both workstations were interconnected using two Intel 82599 10 Gbit/sec NICs. As a third workstation (where needed in our experiments), we used an Intel i7-2600 CPU with 16 GB RAM running Linux 4.4.0 for the FIXIDS intrusion detection, connected to the IPFIX exporting workstation with a 1 Gbit/sec link.

[9]http://dev.chromium.org/spdy/spdy-whitepaper
[10]http://www.ntop.org/products/netflow/nbox/

TABLE II
NUMBER OF EVENTS TRIGGERED BY SNORT AND BY FIXIDS USING THE SAME NETWORK TRACE AND THE SAME RULES

Rule SID	Snort	FIXIDS
2001595	*30*	*30*
2101350	*17*	*13*
2018329	25	25
2017531	25	25
2014999	25	25
2014331	25	25
2013780	25	25
2013291	25	25
2010054	25	25
2010051	25	25
2009950	25	25
2008545	25	25
2007743	25	25
2000580	25	25
40079	25	25
40357	25	25
39190	25	25
37226	25	25
27244	25	25
24482	25	25
24225	25	25
16924	25	25
Total	547	543

B. Functional Evaluation

In the functional evaluation, we show that the methods and algorithms implemented in FIXIDS work as expected. The most important metric is the number of detected events.

To asses the general detection functionality we compare the detected events from Snort and FIXIDS. For this experiment, we added 20 Flows (each 25 times with different IP addresses) that contain a pattern that triggers an event from a randomly chosen rule from our set of 1514 rules (from now on called random pattern traces). We than analyzed the produced pcap file with Snort and with FIXIDS. Table II shows the results of this experiment. The first two events (in italics) are the ones that were already present in the proxy trace. FIXIDS misses only four events out of 547, all triggered by the rule with SID 2101350. This rule searches for a pattern followed by a whitespace. Snort matches this whitespace also to the binary sequence 0x7C while FIXIDS does not. This sequence translates to the pipe character ('|') and it is debatable if a whitespace should match this character.

To assess the robustness of the detection functionality of FIXIDS, we conducted the following experiment: We configured Vermont to read packets from the NIC, aggregate them to IPFIX and hand the Flows to the FIXIDS module for intrusion detection. We replayed the standard proxy trace and used a tcpreplay script to randomly replay the 20 random pattern traces 25 times. This results again in 543 patterns to be detected by FIXIDS in the network trace and contains a variety of all possible patterns. We repeated this experiment 100 times, always with a different random distribution of the 500 random pattern traces, to evaluate if FIXIDS is able to detect all attacks in all of the repetitions. FIXIDS always

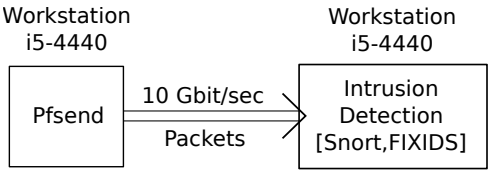

Fig. 2. Experiment setup with pfsend sending packets to Snort or FIXIDS doing intrusion detection. In this case FIXIDS also aggregates packets to IPFIX Flows before the intrusion analysis.

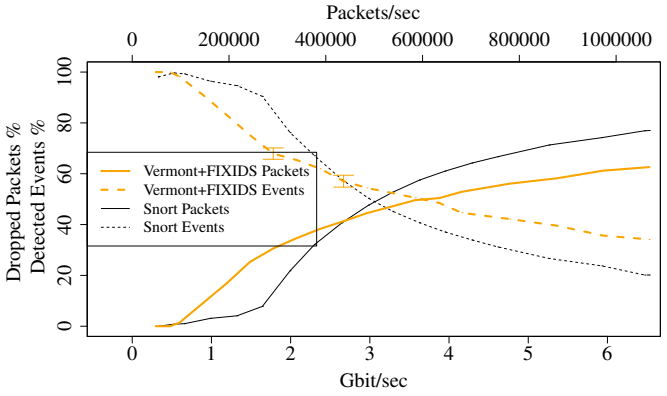

Fig. 3. Comparison of Snort and Vermont FIXIDS performing packet aggregation and intrusion detection on the same workstation receiving the proxy trace (plus 500 additional events) $\times 5$ over the network at different packet rates. Average over 10 runs (confidence interval (95%) is shown if $> \pm 2\%$).

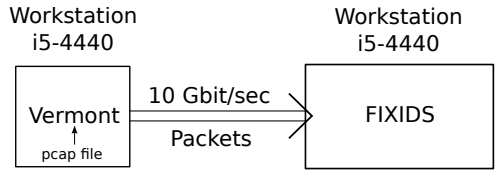

Fig. 4. Experiment setup with Vermont reading the prepared pcap trace file and sending the aggregated IPFIX Flows to FIXIDS.

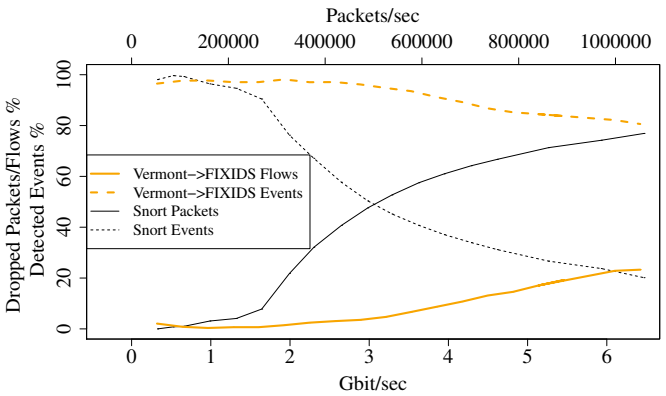

Fig. 5. FIXIDS receiving IPFIX Flows from a remote Vermont entity (over PR-SCTP) reading the proxy trace (plus 500 additional events) $\times 5$ over the network. In comparison to Snort. Average over 10 runs (confidence interval (95%) is shown if $> \pm 2\%$).

(100 %) detected the full number of events. This makes us confident that the detection functionality of FIXIDS is robust and reliable.

C. FIXIDS Performance

In the next set of experiments, we evaluate the network data throughput performance of FIXIDS and compare it to Snort's performance. To asses the network performance of FIXIDS, we used again the proxy trace with the $25 \times 20 = 500$ additional random pattern traces. We then concatenated this trace 5 times, always rewriting all IP addresses. In all tests we replay this concatenated trace multiple times, gradually increasing the data rate to assess the maximum throughput rate without data loss, and the influence of data loss to the detection rate. To replay the above network trace we use the *pfsend* application from the PF_Ring[11] program suite, because it proved to be much more accurate in replaying traffic at a certain rate than other similar programs. For all experiments we use the 1514 Snort rules described in Section III-A for Snort and FIXIDS.

In our first experiment, we evaluate the performance of a monolithic intrusion detection system doing packet aggregation and using the FIXIDS module for analysis on one workstation. We used the two workstations described in Section IV-A connected via a 10 Gbit/sec link. We replayed the traffic at increasing rates from one workstation and did intrusion detection on the incoming packets on the other workstation. First we used Snort on the receiving side to create an intrusion detection benchmark to compare to. This is the baseline Snort performance that we use in all of the following graphs. Then we configured Vermont to fetch packets from the NIC, aggregate them to IPFIX Flows, and hand these Flows to the FIXIDS module for intrusion detection. The experiment setup is sketched in Figure 2.

The following phenomenon holds for both intrusion detection systems: If the analysis engine is overloaded it will tailback packets, which finally results in packet drops at the NIC. The averaged experiment results of 10 runs with confidence intervals of 95 % are shown in Figure 3. For readability reasons, in all of the following graphs, we only show the confidence intervals if bigger than $\pm 2\%$. This figure shows the detected events of Snort (black lines) and FIXIDS (orange lines) as well as the dropped packets (dashed lines) of both systems over multiple experiment runs with increasing rates (in packets per second). For better comparison with the next experiments we also calculated and show the rate in Gbit/sec.

Both systems can cope with 100 000 packets/sec (0.7 Gbit/sec) with less than 1 % packet drop and thus also detect almost all events. As soon as packets have to be dropped also the detected events ratio decreases because the dropped packets will inevitably contain events. With lower rates the FIXIDS system drops more packets while with higher rates it performs better than Snort. Keep in mind that this is the only experiment where Vermont FIXIDS is configured to aggregate the packets to IPFIX Flows before

[11] http://www.ntop.org/products/packet-capture/pf_ring/

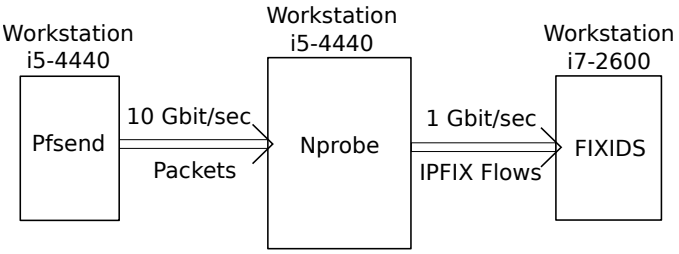

Fig. 6. Experiment setup with pfsend sending packets to Nprobe which aggregates the packets and sends Flows to FIXIDS.

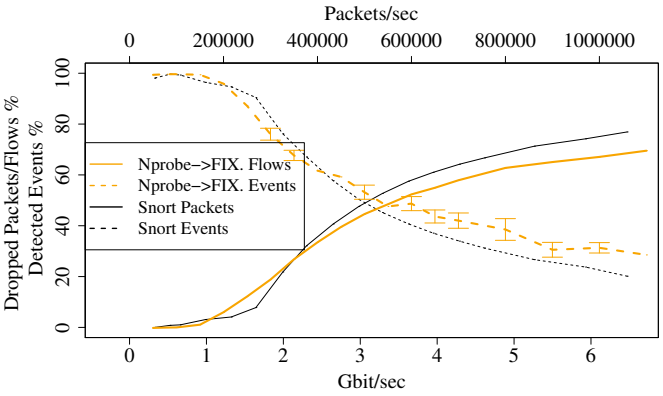

Fig. 7. FIXIDS receiving IPFIX Flows from Nprobe receiving the proxy trace (plus 500 additional events) $\times 5$ over the network. In comparison to Snort. Average over 10 runs (confidence interval (95%) is shown if $> \pm 2\%$).

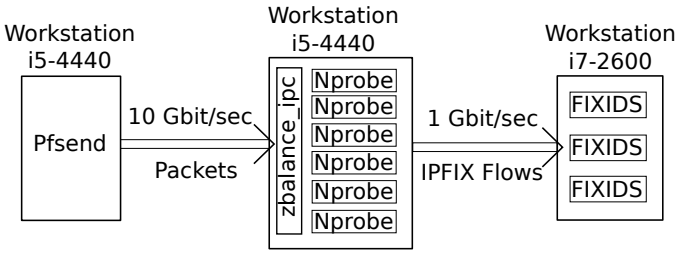

Fig. 8. Experiment setup with pfsend sending packets to 6 Nprobe instances which aggregate the packets and send Flows to 3 FIXIDS instances.

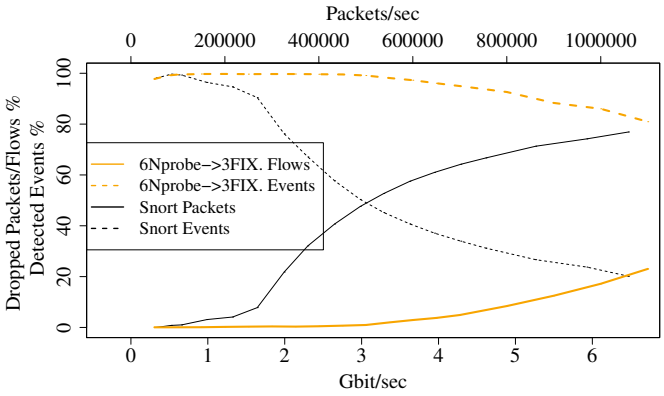

Fig. 9. 3 FIXIDS instances receiving IPFIX Flows from 6 remote Nprobe entities receiving the proxy trace (plus 500 additional events) $\times 5$ over the network. In comparison to Snort. Average over 10 runs (confidence interval (95%) is shown if $> \pm 2\%$).

analysis on the same workstation.

To be able to assess the distribution of packet drops we summed up the reported events of FIXIDS of all experiment runs. When comparing the proportions of the sums of the single rules it reflects exactly (max deviation of 0.5%) the proportions shown in Table II. This means that the packet drops are distributed equally over all events.

While DPI-based NIDS like Snort do its intrusion detection in one monolithic analysis process, the advantage of Flow-based intrusion detection systems is that the process of fetching packets from the wire and aggregating them to Flows can be done separated from the intrusion detection process. This is due to the fact that Flow records can be exported to a Flow collector which than can perform the intrusion detection process. As a matter of fact, most industrial-grade network switches are able to export Flows. Thus, in most cases the Flow-based NIDS does not have to aggregate packets to Flows. In the following experiment we made a Vermont instance read the network trace from a pcap file, aggregate it to IPFIX Flows and export these Flows over the 10 Gbit/sec line to a FIXIDS instance (FIXIDS configuration as in Figure 1). The experiment setup is sketched in Figure 4.

The Vermont IPFIX exporter module supports the export of IPFIX Flows over UDP, PR-SCTP, DTLS over UDP, and DTLS over SCTP transport protocols. We conducted a series of tests that showed that the Vermont IPFIX exporter module in combination with Vermont's IPFIX collector module had the highest throughput rates when using PR-SCTP [32]. Thus, for all experiments with the Vermont IPFIX exporter involved we used PR-SCTP as the transport protocol. We used an option in the IPFIX exporter module of Vermont to configure the different Flow rates. Figure 5 shows the averaged experiment results of 10 runs with confidence intervals of 95 %.

In this constellation, the FIXIDS system can cope with 5000 Flows/sec without drops (orange line) while still detecting all events (orange dashed line). When converted to Gbit/sec this results in more than 1.4 Gbit/sec. This shows that without the IPFIX aggregation module in the same Vermont instance our NIDS performs much better and is 2 times faster than Snort. The graph shows also that the Flow drop rate of FIXIDS increases much slower with an increasing data rate than the packet drop rate of Snort. For instance at 5 Gbit/sec FIXIDS drops less then 20 % of the Flows, detecting more than 80 % of the events while Snort drops about 70 % of the packets and only detects about 30 % of the events. Also in this experiment the dropped Flows are distributed equally over all events.

In the next experiment, we use the third-party network probe and IPFIX collector Nprobe to assess how the FIXIDS NIDS performs together with other network probes than Vermont. Nprobe supports the export of IPFIX Flows over UDP, TCP, and PR-SCTP. Preliminary tests showed that when receiving Flows from Nprobe with the Vermont IPFIX collector used in FIXIDS, the export over TCP outperformed the other transport protocols, however, we did not further investigate this behavior. In our experiment setup the performance of Nprobe

increased significantly, if we incremented its Flow hash size (-w switch) to 10000 and decreased the Flow timeout values (-t, -d, -l) to 1. The experiment setup is shown in Figure 6.

As Nprobe does not support to configure the Flow export rate we had to use a dedicated replay workstation to be able to control the data rate. We used pfsend to replay the network trace over the 10 Gbit/sec link. On the receiving side of this link Nprobe aggregated the incoming packets into Flow records and sent the Flows over the 1 Gbit/sec link to FIXIDS (configured as in Figure 1). If the test trace (size is 9 GB) is aggregated to IPFIX Flows as needed for the experiments it has a size of roughly 30 MB. This means that, even when adding the transport layer headers and the occasional sending of IPFIX template records, a 1 Gbit/sec link is more than enough to transport the aggregated Flows of network data coming from a 10 Gbit/sec link. It has to be noted that the workstation that FIXIDS is running on in this experiment has less memory and a slightly less modern and powerful CPU than in the experiments before.

The orange lines in Figure 7 show the results compared to the Snort performance (black). The data throughput performance is about the same as Snort and, thus, worse than in all experiments before. The reason is that Nprobe is sending Flows in massive bursts (much larger than Vermont) and, thus, at high packet rates, Flows have to be dropped by the Vermont IPFIX collector of FIXIDS. But also the weaker workstation might have a minor influence on the results. Again, also with Nprobe the dropped Flows are distributed equally over all events.

The orange lines of Figure 9 show the result of the next experiment: We exploited Nprobe's native support of the PF_Ring driver family. It allows to hash the incoming packets and distribute them to a configurable number of PF_Ring capable application instances by using hardware features of modern NICs. This guarantees that packets with the same IP address are always assigned to the same application instance. This is important because this way all packets belonging to one Flow are processed by the same aggregation instance. By using the *zbalance_ipc* application, that is shipped with the PF_Ring driver suite, we distributed the incoming packets to 6 Nprobe instances. This experiment setup is sketched in Figure 8. Every Nprobe instance uses 2 CPU cores during run-time, so this configuration makes sure that all cores on our 12-core workstation are used. The 6 Nprobe instances exported the IPFIX Flows over the 1 Gbit/sec link to 3 FIXIDS instances running on the receiving workstation. Always two Nprobe instances were connected to one FIXIDS instance.

The orange line in Figure 9 shows the dropped Flows and the dashed orange line the detected events for FIXIDS. The results shown are the sum of the three FIXIDS instances. In this configuration the three FIXIDS instances process more than 2.5 Gbit/sec (roughly 8000 Flows/sec) on one workstation. This is more than 3 times the throughput of Snort. And again, with increasing data rates the drop rate increases much slower than Snort's drop rate. This shows that the best data throughput can be achieved when parallelizing the packet aggregation process as well as the Flow-based intrusion detection of FIXIDS.

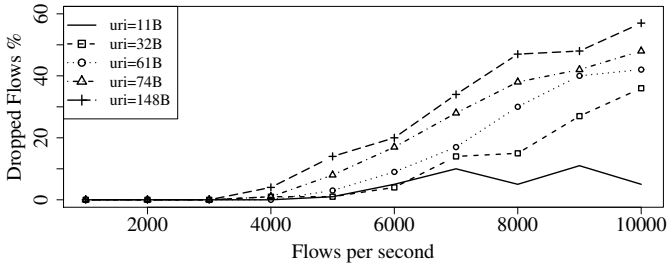

Fig. 10. Vermont FIXIDS performing intrusion detection on 100k IPFIX Flows with different HTTP Uri lengths at different Flow rates.

D. URI Matching Bottleneck

The efficiency of the FIXIDS module is highly dependent on the string compare function that tries to match the signature pattern to the corresponding HTTP IE fields. To showcase this we conducted the following experiment: The vast majority of the employed rules contains the content modifier http_uri, resulting in most patterns being matched against the HTTP URI found in the IE field httpRequestTarget. Thus, we prepared a trace where we extracted an HTTP request and the corresponding response from the proxy trace and concatenated this trace 100 000 times, always changing the IP address. When aggregated to IPFIX, this trace results in 100 000 IPFIX Flows with the same httpRequestTarget IE field. We repeated this procedure to have 5 different traces with 5 different httpRequestTarget IE field lengths. We then performed an experiment where we used Vermont to read these traces and export the resulting IPFIX Flows (via PR-SCTP) to a FIXIDS system.

The results are shown in Figure 10. The single lines represent the rate of dropped Flows during intrusion detection on traces with different URI lengths. It shows that the longer the URI length in the Flow is, the more Flows are dropped with increasing data rates. Thus, the data throughput performance of FIXIDS is highly dependent on the signature pattern matching process.

V. Conclusion

In this paper, we presented IPFIX-based Signature-based Intrusion Detection System (FIXIDS), a signature-based Flow Network Intrusion Detection System (NIDS) for high-speed networks that runs on commodity hardware. It tackles the problem of comparably low network data throughput for signature-based intrusion detection systems in high-speed networks. It is the first signature-based NIDS that completely operates on Flow information using the novel HTTP Information Elements (IEs). As input it takes HTTP signatures from Snort rules and applies them to HTTP related IEs in IPFIX Flows. By using Flows the amount of data to be analyzed is much less compared to traditional DPI-based NIDS. It, thus, promises a higher network throughput performance.

In a comprehensive experiment campaign we show that FIXIDS detects all relevant events and can handle a much higher data throughput than Snort with the same network data and the same rule-set. We also show that FIXIDS is able to analyze Flows from third-party network probes and, thus, can be used in heterogeneous network monitoring environments. Comparing the straightforward pattern matching algorithm of FIXIDS with the sophisticated and complex algorithm of the Snort NIDS it appears that there is an additional performance gain to be expected when further improving the FIXIDS pattern matching algorithm. It is planned to adapt FIXIDS to accept also standard Flow features in rules. This way FIXIDS could also be extended to use other signature or anomaly-based intrusion detection techniques.

REFERENCES

[1] B. Stritter, F. Freiling, H. König, R. Rietz, S. Ullrich, A. von Gernler, F. Erlacher, and F. Dressler, "Cleaning up Web 2.0's Security Mess - at Least Partly," *IEEE Security & Privacy*, vol. 14, no. 2, pp. 48–57, Mar. 2016.

[2] R. Kemmerer and G. Vigna, "Intrusion Detection: A Brief History and Overview," *IEEE Computer, Special Issue on Security and Privacy*, pp. 27–30, Apr. 2002.

[3] H. Debar, M. Dacier, and A. Wespi, "Towards a Taxonomy of Intrusion-Detection Systems," *Elsevier Computer Networks*, vol. 31, no. 8, pp. 805–822, Apr. 1999.

[4] M. H. Bhuyan, D. K. Bhattacharyya, and J. K. Kalita, "Network Anomaly Detection: Methods, Systems andTools," *IEEE Communications Surveys & Tutorials*, vol. 16, no. 1, pp. 303–336, 2014.

[5] M. Roesch, "Snort: Lightweight Intrusion Detection for Networks," in *13th USENIX Conference on System Administration (LISA 1999)*, Seattle, WA, Nov. 1999, pp. 229–238.

[6] V. Paxson, "Bro: A System for Detecting Network Intruders in Real-Time," *Elsevier Computer Networks*, vol. 31, no. 23-24, pp. 2435–2463, Dec. 1999.

[7] P.-C. Lin and J.-H. Lee, "Re-examining the Performance Bottleneck in a NIDS with Detailed Profiling," *Journal of Network and Computer Applications*, vol. 36, no. 2, pp. 768–780, 2013.

[8] R. Hofstede, P. Celeda, B. Trammell, I. Drago, R. Sadre, A. Sperotto, and A. Pras, "Flow Monitoring Explained: From Packet Capture to Data Analysis with Netflow and IPFIX," *IEEE Communications Surveys and Tutorials*, vol. 16, no. 4, pp. 2037–2064, 2014.

[9] T. Limmer and F. Dressler, "Adaptive Load Balancing for Parallel IDS on Multi-Core Systems using Prioritized Flows," in *20th IEEE International Conference on Computer Communication Networks (ICCCN 2011)*. Maui, HI: IEEE, July/August 2011, pp. 1–8.

[10] F. Erlacher and F. Dressler, "High Performance Intrusion Detection Using HTTP-Based Payload Aggregation," in *42nd IEEE Conference on Local Computer Networks (LCN 2017)*. Singapore: IEEE, Oct. 2017, to appear.

[11] B. Claise, "Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of IP Traffic Flow Information," IETF, RFC 5101, Jan. 2008.

[12] J. Quittek, S. Bryant, B. Claise, P. Aitken, and J. Meyer, "Information Model for IP Flow Information Export," IETF, RFC 5102, Jan. 2008.

[13] B. Ager, N. Chatzis, A. Feldmann, N. Sarrar, S. Uhlig, and W. Willinger, "Anatomy of a Large European IXP," in *ACM SIGCOMM 2012*. Helsinki, Finland: ACM, Aug. 2012, pp. 163–174.

[14] P. Richter, N. Chatzis, G. Smaragdakis, A. Feldmann, and W. Willinger, "Distilling the Internet's Application Mix from Packet-Sampled Traffic," in *Passive and Active Measurement Conference (PAM 2015)*, New York City, NY, Mar. 2015.

[15] F. Erlacher, W. Estgfaeller, and F. Dressler, "Improving Network Monitoring Through Aggregation of HTTP/1.1 Dialogs in IPFIX," in *41st IEEE Conference on Local Computer Networks (LCN 2016)*. Dubai, UAE: IEEE, Nov. 2016, pp. 543–546.

[16] S. Degli Esposti, V. Pavone, and E. Santiago-Gomez, "Aligning Security and Privacy: The Case of Deep Packet Inspection," in *Surveillance, Privacy and Security: Citizens' Perspectives*, M. Friedwald, P. Burgess, J. Cas, R. Bellanova, and W. Peissl, Eds. Routledge, 2017.

[17] G. Bianchi, E. Boschi, D. I. Kaklamani, E. Koutsoloukas, G. V. Lioudakis, F. Oppedisano, M. Petraschek, F. Ricciato, and C. Schmoll, "Towards privacy-preserving network monitoring: Issues and challenges," in *18th IEEE International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC 2007)*. Athens, Greece: IEEE, Sep. 2007.

[18] G. Vasiliadis, S. Antonatos, M. Polychronakis, E. P. Markatos, and S. Ioannidis, "Gnort: High Performance Network Intrusion Detection Using Graphics Processors," in *11th International Symposium on Recent Advances in Intrusion Detection (RAID 2008)*, vol. LNCS 5230. Cambridge, MA: Springer, Sep. 2008, pp. 116–134.

[19] C. Kruegel, F. Valeur, G. Vigna, and R. Kemmerer, "Stateful Intrusion Detection for High-Speed Networks," in *IEEE Symposium on Security and Privacy*, Oakland, CA, May 2002, pp. 285–293.

[20] R. Sommer, V. Paxson, and N. Weaver, "An architecture for Exploiting Multi-core Processors to Parallelize Network Intrusion Prevention," *Concurrency and Computation: Practice and Experience*, vol. 21, no. 10, pp. 1255–1279, 2009.

[21] S. Kornexl, V. Paxson, H. Dreger, R. Sommer, and A. Feldmann, "Building a Time Machine for Efficient Recording and Retrieval of High-Volume Network Traffic," in *5th ACM SIGCOMM Conference on Internet Measurement (IMC 2005)*. Berkeley, CA: ACM, Oct. 2005, pp. 267–272.

[22] T. Limmer and F. Dressler, "Improving the Performance of Intrusion Detection using Dialog-based Payload Aggregation," in *30th IEEE Conference on Computer Communications (INFOCOM 2011), 14th IEEE Global Internet Symposium (GI 2011)*. Shanghai, China: IEEE, Apr. 2011, pp. 833–838.

[23] V. Pus, P. Velan, L. Kekely, J. Kovrenek, and P. Minavr'ik, "Hardware Accelerated Flow Measurement of 100 Gb Ethernet," in *14th International Symposium on Integrated Network Management (IM 2015)*. Ottawa, Canada: IEEE, May 2015, pp. 1147–1148.

[24] F. Fusco and L. Deri, "High Speed Network Traffic Analysis with Commodity Multi-core Systems," in *10th ACM Internet Measurement Conference (IMC 2010)*. Melbourne, Australia: ACM, Nov. 2010, pp. 218–224.

[25] A. Sperotto, G. Schaffrath, R. Sadre, C. Morariu, A. Pras, and B. Stiller, "An Overview of IP Flow-based Intrusion Detection," *IEEE Communications Surveys and Tutorials*, vol. 12, no. 3, pp. 343–356, 2010.

[26] R. Hofstede, V. Bartos, A. Sperotto, and A. Pras, "Towards Real-time Intrusion Detection for NetFlow and IPFIX," in *9th International Conference on Network and Service Management (CNSM 2013)*, Zurich, Switzerland, Oct. 2013, pp. 227–234.

[27] T. Dubendorfer, A. Wagner, and B. Plattner, "A Framework for Real-Rime Worm Attack Detection and Backbone Monitoring," in *First IEEE International Workshop on Critical Infrastructure Protection (IWCIP 2005)*. Darmstadt, Germany: IEEE, Nov. 2005, pp. 10–pp.

[28] F. Dressler, W. Jaegers, and R. German, "Flow-based Worm Detection using Correlated Honeypot Logs," in *15. GI/ITG Fachtagung Kommunikation in Verteilten Systemen (KiVS 2007)*, T. Braun, G. Carle, and B. Stiller, Eds. Bern, Switzerland: VDE, Feb. 2007, pp. 181–186.

[29] L. Hellemons, L. Hendriks, R. Hofstede, A. Sperotto, R. Sadre, and A. Pras, "SSHCure: A Flow-Based SSH Intrusion Detection System," in *6th International Conference on Autonomous Infrastructure, Management, and Security (AIMS 2012)*. Luxembourg, Luxembourg: Springer, Jun. 2012, pp. 86–97.

[30] R. Hofstede and L. Hendriks, "Unveiling SSHCure 3.0: Flow-based SSH Compromise Detection," in *International Conference on Networked Systems (NetSys 2015), Demo Session*, Cottbus, Germany, Mar. 2015.

[31] L. Deri, "nProbe: an Open Source NetFlow Probe for Gigabit Networks," in *TERENA Networking Conference (TNC 2003)*, Zagreb, Croatia, May 2003.

[32] R. Steward, M. Ramalho, Q. Xie, M. Tuexen, and P. Conrad, "Stream Control Transmission Protocol (SCTP) - Partial Reliability Extension," IETF, RFC 3758, May 2004.