Real-Time Security Services for SDN-based Datacenters

Pál Varga[2] Georgios Kathareios[1] Ákos Máté[1] Rolf Clauberg[1] Andreea Anghel[1]
Péter Orosz[2] Balázs Nagy[3] Tamás Tóthfalusi[2] László Kovács[3] Mitch Gusat[1*]

[1]IBM Research – Zurich, Switzerland; Email: {ios,kos,cla,aan,mig}@zurich.ibm.com
[2]Budapest University of Technology and Economics, Hungary; Email: {pvarga,orosz,tothfalusi}@tmit.bme.hu
[3]AITIA International Inc., Budapest, Hungary; Email: {bnagy,lkovacs}@aitia.ai

Abstract—**While the scale, frequency and impact of the recent cyber- and DoS-attacks have all increased, the traditional security management systems are still supervised by human operators in the decisional loop. To cope with the new breed of machine-driven attacks - particularly those designed to overload the humans in the loop - the next-generation anomaly detection and attack mitigation schema, i.e. the network security management, must improve greatly in speed and accuracy: become machine-driven, too. As infrastructure we propose an FPGA-accelerated Network Function Virtualization that potentially enhances the current multi-Tbps switching fabrics with SDN-based security capabilities of vastly higher performance and scalability. As key novelties, we contribute (i) sub-ms detection lag (ii) of the top 9 Akamai attacks [1] with (iii) a real-time SDN feedback loop between a distributed programmable data plane and a centralized SDN controller, (iv) coupled via a global N:1 mirror. We validate the concept in an actual datacenter network with a new security application that can detect and mitigate real-world dDoS attacks, with lags from 430 us up to 3 ms - several orders of magnitude faster than before.**

Index Terms—**SDN, dDoS, switching, datacenter networking, online datapath monitoring, intrusion detection and prevention.**

I. Introduction and Motivation

Security experts lately notice more high-frequency ephemeral attacks in the data path, engineered specifically to overload the human component in the loop [2]. These may entail sudden traffic volume swells directed towards a (subset of) targets, lasting perhaps just milliseconds. Such high-frequency transient anomalies hardly detectable with the current security instrumentations – and even harder to prevent their potentially harmful impacts. Indeed, although today's datacenter networks (DCNs) are built up of high-speed low-latency Tbps fabrics – often supported by versatile SDN controllers – still their control and management mechanisms are slow in reacting to and mitigating such anomalies.

As an example, TCP flows can stay in SYN received state between 1 and 5 minutes, depending on the configuration. The number of SYN state entities in regular TCP stacks at server or other endpoints range between 1,024 (the default backlog size in the linux kernel) and a few thousands for high-performance webservers [3]. In case such a server or endpoint is connected via at least 10Gbps Ethernet, an aggressive DoS attack can fill it with TCP SYNs within a few ms. Such attacking traffic can remain completely unnoticed, since the reaction time of current security solutions is a few orders of magnitude higher. This server cannot establish any further TCP connections for the remaining 1-5 min. – until the SYN timers will expire.

The increasing occurrence of such high-frequency anomalies requires high-performance, low-delay *automated* SDN security solutions. The available commodity switching fabrics, however, have limited capabilities both in delay and data granularity when providing reports to SDN controllers. This reporting can be improved and accelerated, e.g., via an FPGA-based traffic analyzer. Such an FPGA-based element can provide low-latency insights on security issues and other traffic anomalies to the SDN controller. The newer DCN fabrics also allow to attach these elements via a *switch global* mirroring port [4], [5], ie., a dedicated switch port to which the entirety of the switch traffic is mirrored.

Accordingly, we introduce an automated method to speed-up the reaction lag in SDN-based datacenters via a new closed, passive monitoring loop, accelerated by FPGA-based processing units. Our solution aims to enhance the security capabilities of DCNs and large Cloud networks with a new FPGA-accelerated Network Function Virtualization (NFV). Contrary to the common practice today, we aim to *autonomously* detect and mitigate the attacks designed to overwhelm the human operator. The performance of our approach is 5-7 orders of magnitude higher than the state of the art (us/ms-scale vs. hour-scale), based on the automatic attack mitigation instead of the typical human-in-the-loop. Our solution can mitigate the most common 9 attacks [1] and is universally compatible with most modern Tbps-class DCN switches, needing neither deep packet inspection, nor the encryption keys. We present here the results of our proof-of-concept implementation for 3 out of the 9 most common attacks: UDP flood, DNS reflection and SYN flood. Our solution can be deployed both to the entry point(s) of the DCN and also in strategically-selected switches within the DC.

The rest of the paper is structured as follows. In Section II we briefly describe the common fabrics used in today's DCNs. In Section III we introduce our FPGA-based SDN solution for network security monitoring. In Section IV we describe a proof-of-concept implementation of our solution. Section V presents FPGA system we used, followed by experimental results and discussion in Section VI. We conclude in Section VII.

*Corresponding author.

We are thankful to the National Information Infrastructure Development Institute of Hungary for their support. This research is partly supported by the EU H2020 research and innovation programme under the grant agreement 644960.

978-3-901882-98-2 © 2017 IFIP

TABLE I
COMMODITY SWITCH CHIPS

DCN Switch	Port rate [Gbps]	100Gb ports	Total Bw [Tbps]	SDN support and key features
BCM56860 (Trident II+)	1/10/40/100	8	0.8	OF1.3+ with Broadcom OF-DPA VXLAN/NVGRE overlay and tunneling support
BCM56960 (Tomahawk)	25/40/50/100	32	3.2	OF1.3+ with Broadcom OF-DPA, VXLAN/NVGRE overlay and tunneling support; FleXGS™ flow processing for configurable forwarding /match /actions
Tomahawk II	40/50/100	64	6.4	
Mellanox Spectrum	40/56/100	32	6.4	OF1.4+ VxLAN/NVGRE overlay and tunneling support
Intel FM10840 Red Rock Canyon	1/10/25/100	6+4 PCIe	0.6-3.5	L2/L3/L4 OF forwarding, VxLAN and NVGRE tunnels, 4 MB shared memory, 32K 40-bit TCAM entries, 16K NextHop tables, multi-switch SWAG configuration
Mellanox NP-5	1/10/40/100	2	0.24	Task Optimized Processing engines allow flexible packet classification and modification, internal TCAM, internal memory, 24 DDR3 SDRAM I/F, 44 x 10.3125 Gb/s Interlaken

II. BRIEF SURVEY OF DCN SWITCHING FABRICS

Let us compare the key features and capabilities of ASIC-based commodity fabrics and OpenFlow (OF-native) switches.

1) COMMODITY SWITCHES: The highest performing commodity switching chips on the DCN market offer up to 64 ports at 100 Gbps. Table I shows the main characteristics of the common merchant switch chips used in today's DCN fabrics. The internal buffering and management system, scheduling and table sizes are usually not fully documented. Thus it is relevant to differentiate features internally implemented within the chips from those added by means of external parts. Table I lists the key hardware components to enable the SDN support, i.e. internal memory, programmable processing units, and high-speed IO interfaces to the corresponding external parts. Also listed are the additional software needed, the level of OF and Overlays support. Capability-wise, the number of flow forwarding entries (FFE) for the Arista [6] and Cisco Nexus [7] switches exceed 1M entries, but only with external TCAMs or DDR3/DDR4 DRAMs. Typically no information on the number of flow modifications per second (FMPS) is available.

2) SDN/OPENFLOW NATIVE SWITCHES: The 'native' SDN/OF switches are also (partly) based on ASIC fabric components. E.g., Corsa [8] uses an ASIC and an FPGA, whereas the NoviSwitch system [9] is based on the Mellanox NP-5 [10] network processor with integrated programmable processing units (PPU). The FFE table size depends on external parts. E.g., the NoviSwitch 21100 offers up to 1M flow entries with external TCAM, versus only 16K entries with the internal TCAM. Notably different from the previous commodity switches, the SDN/OF switches often specify the number of flow modifications per second as a distinguishing performance metric. Their figures are between 15K [8] and 40K [9] FMPS.

Comparing the above switching systems, we conclude that ASIC-based commodity switches typically outperform the SDN/OF-dedicated switches. The typical commodity fabrics achieve a switching capacity of 1.2 to 3.6 Tbps in a single line card, e.g., [6], vs. 0.512 Tbps [9] or 0.2 Tbps [8]. Most commodity switches often do not specify their respective capabilities for online modification of flow table entries. In contrast, the native programmability of the SDN switches specifies up to 40K FMPS. The key for fast modifications in the native SDN switches is the use of function-specific PPU [9] or FPGAs [8]. Software implementations with standard CPUs in the control plane are too slow to achieve this goal at 40/50/100 Gbps and 10s of ports, i.e., multi-Tbps aggregate speeds.

III. WHY ACCELERATED COMMODITY SWITCHING?

A large body of the current research has focused recently on the role of FPGAs and specialized processors as datacenter accelerators [11], [12], [13], [14]. Similar to the Knowledge Plane concept [15], we propose to 'attach' an FPGA-based SDN accelerator to commodity switches. Thus we aim to reconcile the raw performance, 802-compliance and affordability of the current Tbps-class commodity switches with the versatility and programmability of native SDN designs.

Given the key role of online (i) flow modifications, and (ii) datapath network operations also during the high utilization and network congestion periods, we propose a solution in bridging the discrepancy between today's Ethernet commodity and SDN switches. We augment the generic ASIC switches by an FPGA attached via a high speed (40/100 Gbps) *switch global* mirroring port. This we prove as a simple, generic (mirroring is generally supported by today's switches), flexible (many use-cases), scalable (depending mainly on the number or mirroring ports dedicated to SDN acceleration) method.

Exploiting (i) the flexibility of FPGAs in datacenter applications, and (ii) the fact that they typically outperform in terms of packet processing performance the typical CPUs of today's commodity switches, there is a variety of use-cases whereby a commodity switching fabric can be practically enhanced with SDN-like FPGA-based NFVs. E.g.: Network security, monitoring, load-balancing, adaptive routing, SDN overlay networks etc. We will focus on the first use-case.

Security and Quality of Service (QoS) are increasingly among the key considerations in the design of modern DC

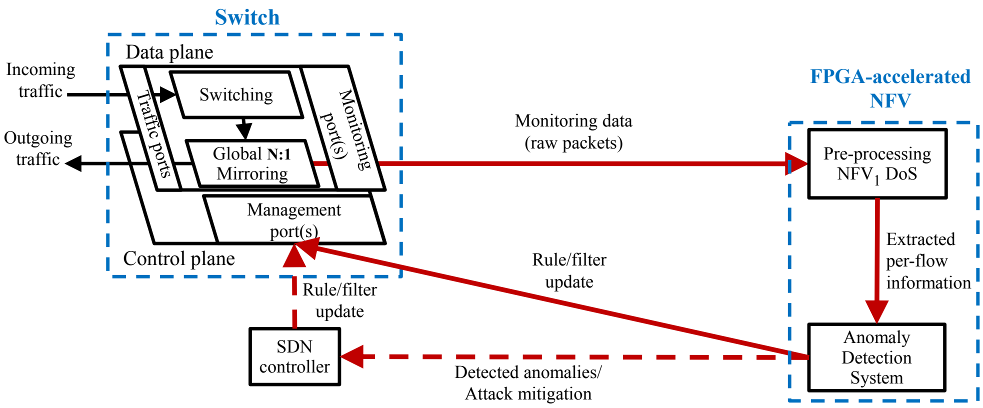

Fig. 1. Real-time network security FPGA-based design. The accelerator is attached to the switch via a global mirroring port. The FPGA can be responsible for making very fast local decisions and/or informing the centralized SDN controller for slower, but more informed reaction. The accelerator collects the 10-100 Gbps streams of monitoring data, processes them and issues real-time updates to the fabric reconfiguration engine.

fabrics and DCNs. Especially in multi-tenant datacenter environments, the protection from intended or unintented harmful activity and the enforcement of agreed-upon Service Level Agreements (SLAs) is of paramount importance for a cloud provider. Similarly, in single-tenant PaaS and IaaS datacenters, the detection of hardware malfunction and miss-configurations – as a major source of disruptions! – along with the detection of intrusion attempts are essential for ensuring the uninterrupted operation of the offered services. As there is a diverse set of monitoring and security operations that depend heavily on the intended use of a datacenter, it is not trivial for the ASIC suppliers to include and maintain all such capabilities in a low-cost commodity switching DCN fabric.

In practice the typical switch CPUs do not have proper capabilities to detect and timely react to security threats. Examples are mentioned in the work of Suh et al. [16], where it is shown that a common implementation of the sFlow packet sampling functionality can only sample up to approximately 350 packets per seconds, or in studies showing that the switch CPU can be overwhelmed by OpenFlow operations [17], [18], [19]. We will show a practical example of how the attached FPGA can accelerate DCN monitoring by extracting packet-level information exclusively from the switch's data plane and utilizing the information for real-time DoS attack mitigation.

IV. SDN Monitoring: Platform and Setup

As a proof of concept we have used an FPGA-based network accelerator, the C-GEP board [20], [21] attached in a closed feedback loop to a RedRock Canyon (RRC) FM10840 fabric, in the context of high-speed monitoring for security. We use the C-GEP board as (1) a passive monitor for capturing and analyzing traffic at up to 100 Gbps, and as (2) an optional active local SDN controller able to react within us-to-ms timescales to mitigate anomalies and threats. We show an overview of our solution design in Fig. 1.

A. Port Mirroring: A Generic Monitoring Primitive

For our high-speed SDN monitoring solution we rely on an oversubscribed global (to the switch) mirroring port concept similar to the Planck [4] and Everflow [5] systems, to capture and deliver raw packets to the accelerator. We reserve a small subset – typically one, or 2-4 for special cases – of the data-plane switch ports and use them specifically to forward the monitoring data. We will refer to these reserved ports as *monitoring port*s, and they should not be confused with any special-purpose management and monitoring ports that typically operate on the control plane. The switch is configured to mirror *all* the packets (unless "Match&Mirror" is selected) to the monitoring ports, a multicast-like operation that is supported in hardware on modern switches. Due to the N:1 *oversubscription* packet drop may occur due to limited buffers and forwarding in the monitoring ports, which Planck [4] uses as a sampling function. The aggregated multi-Tbps traffic is thus randomly sampled during the N:1 ’overloads’, while the monitoring data stream remains constant, e.g 100Gbps. As the oversubscribed mirror performs this sampling, the monitoring data contain a *global view* of all traffic in the switch. Although this method may raise concerns of buffer hogging, it has been shown to have a negligible effect on the production traffic [4].

The merit of monitoring-via-global-mirroring is the speed at which the packets are *transparently* extracted from the dataplane. We preserve the monitoring data channel entirely within the data plane, circumventing the switch CPU. Thus we can achieve (much) higher throughput and lower latency in the FPGA-based closed loop than in schemes such as SNMP [22], sFlow [23] or NetFlow [24]. The faster the data can be captured, the faster we can detect anomalies and react. Additionally, this monitoring method also works well under extreme conditions, e.g., during congestion as in Denial-of-Service (DoS) [1] attacks. On the contrary, in such a scenario a control-plane CPU would be overwhelmed and could push monitoring tasks to a lower priority. This can lead to the undesirable effect of producing less and less monitoring data, meaning that our ability to analyze and react decreases with the intensity of the attack. Data-plane monitoring, on the other hand, produces data at the same rate under all conditions, limited only by the aggregate throughput of the monitoring ports used for the mirror.

B. Packet Processing: FPGA-Based Pipeline

The volume of raw data streams captured via mirroring several 100 Gbps ports raises the issues of storing and processing this information. Since it is not practically feasible to store this raw data on disk, there is a need for online, real-time processing. Furthermore, the raw packets carry a large amount of redundant information and need to be pre-processed in order to acquire estimates of the necessary metrics needed for the various security applications.

In our pipeline (Fig. 1), the pre-processing step is done by the C-GEP FPGA. Although it would be possible and beneficial in many applications, for the privacy of our network traffic we opt not to perform deep-packet inspection (DPI), but rather to rely solely on the information included in the Layer 2 to Layer 4 headers. Non-DPI monitoring will become key in the future DCN for two reasons: 1) privacy is a huge concern for users and cloud providers with the EU's GDPR update becoming law in May'18, and, 2) the ubiquity of end-to-end encryption and the use of multiple encrypted tunnels and transports will make DPI less feasible as multiple keys – often managed by different entities – will be needed for payload decoding. Thus the future security applications will have to increasingly rely on the behavioral characteristics of the traffic, rather than on its content, in order to detect the anomalous behaviors.

The preprocessing stage of our pipeline (Fig. 1) converts the mirrored data stream into a suitable format for anomaly detection. Depending on the desired application or seervice, this preprocessing may include the estimation of flow-level statistics (e.g., per-flow throughput) or traffic matrices (e.g., subnet level).

The data is then analyzed in order to (1) detect anomalies, i.e., behaviors that diverges from the 'normals' of the traffic flowing through the switch, and, (2) identify the culprits of these anomalies. The anomaly detection can be based on a wide variety of algorithms. Well-known types of anomalies can usually be detected with rule-based (signature-based) detectors [25]. Novel types of anomalies can be discovered using behavioral-based detectors [26] that use statistical modeling and machine learning techniques to detect outliers in the stream of monitoring data.

C. Closed-Loop Monitoring Feedback: Local vs. Global

The red lines in Fig. 1 represent the closed-loop feedback of our solution. After the anomaly detection stage, there are two options for utilizing the results: (i) forwarding the anomalies' report to the centralized SDN controller; or, (ii) using the locally-attached C-GEP as a decentralized controller to make autonomous decisions.

With option (i), the discovered anomalies are forwarded to a centralized controller, which takes the responsibility for (a) deciding on a reaction plan (either on its own or in conjunction with other security entities), and, (b) updating the configuration of the switch with new rules. This option is more suitable for anomalies detected by a behavioral-based detector (novel anomalies). For this kind of anomalies, there is usually a lower confidence that they constitute malicious traffic, as there can also be novel benign traffic. Therefore, it is not desirable to act upon them before validating that they indeed constitute a danger to the network. The central controller also has a global view of the network and can thus cross-validate information and react in multiple points across the fabric.

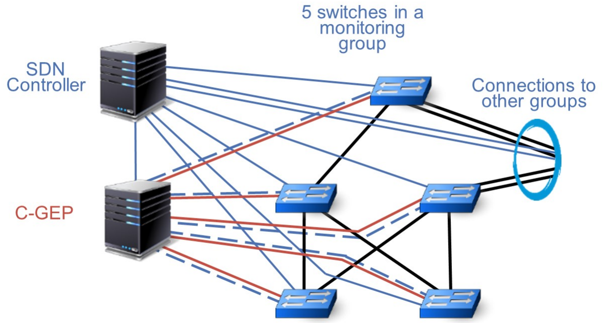

Fig. 2. An example of a jointly monitored switch group comprising of 5 switches. Monitoring links between the switches and the C-GEP (in orange) operate entirely in the data plane and provide high-speed data capture. Reaction to anomalies detected in the traffic can be initiated either directly from the C-GEP (dashed blue links) or by communicating the information to the SDN controller (solid blue links), both operating in the control plane.

With option (ii), the C-GEP makes local decisions and reacts by online (re)updating the switch configuration on its own, thus acting as a decentralized, local SDN controller. The major benefit of option (ii) is that the reaction time can be reduced by a few orders of magnitude, as there is neither a communication delay with the centralized controller, nor a waiting period for the validation process. In addition, the centralized controller can become a bottleneck in the case of distributed attacks in larger-scale networks, since the reaction may not reach all across the network fast enough to adequately mitigate the attack. Selecting between the two options is a trade-off that needs to be balanced based on the services provided by the DC, network, anomaly detection methodologies and the portion of total network control performed by the centralized controller.

D. Monitoring Groups: SDN Switch Clustering

To fully utilize the resources of each C-GEP board, we assign the monitoring responsibility of a group of K switches to a single C-GEP as depicted in Fig. 2. The parameter K depends on the total processing capabilities that the C-GEP can provide to the given application. Currently, a single C-GEP can process approx. 100 Gbps of traffic for the detection of volumetric anomalies. Hence, for a monitoring application that could tolerate a higher rate of packet sampling and 10 Gbps monitoring ports, a single C-GEP would suffice to monitor up to $K=10$ switches in parallel. Thus a single accelerator enables to create a monitoring group for several Tbps of managed traffic.

V. C-GEP: A Universal SDN Accelerator

C-GEP [20], [21] is a high-performance FPGA-based networking hardware platform. Its reconfigurable architecture enables a wide variety of networking applications (from media gateways, traffic generators or monitors to SDN devices) to

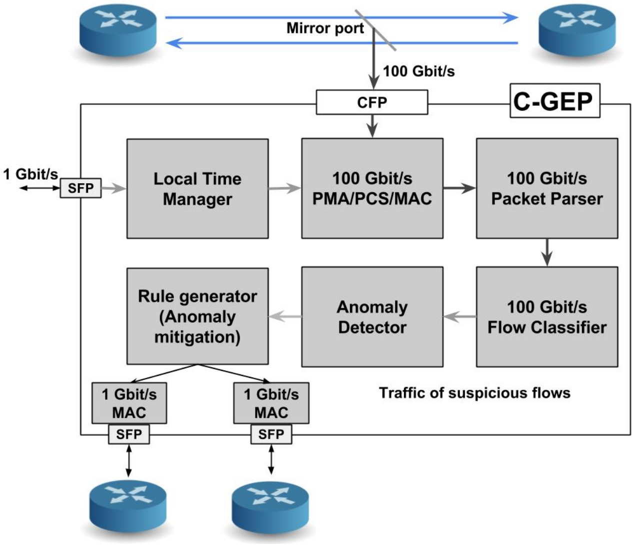

Fig. 3. C-GEP anomaly detection pipeline.

be implemented and accelerated at 100Gbps. Here we present how C-GEP can enhance the SDN functionality of commodity Tbps-class datacenter switches. We show one practical use-case dedicated to DCN security in which the mirror-attached FPGA assists the switch fabric with security-related tasks: packet parsing, flow classification, anomaly detection and mitigation of malicious traffic by using dynamic ACL rules.

The main design principle for enhancing commodity switches with SDN functionality at 40-100Gbps is a multi-pipeline architecture that enables a high degree of paralellism in a single FPGA chip. The modular architecture of the platform was introduced in [20], [21].

The 512-bit internal C-GEP data path involves 4 pipeline stages (modules) operated at 312.5 MHz: 100Gbps Ethernet MAC with packet timestamping, packet parser, packet classifier, anomaly detector and mitigator, respectively (Fig. 3). The packet parser engine operates on protocol headers based on a parse graph that is a compact representation of the predefined header structures. To cover the spectrum of network protocols present in DCNs, the parser engine allows the identification of multi-encapsulated packets beyond decoding the classical 5-tuple.

One essential functionality in many networking systems is packet classification. This is the third phase in the internal processing path of C-GEP in the intrusion detection use-case. It requires input about the protocol fields of each incoming packet from the packet parser stage, with the number and size of header fields determining the complexity of the lookup engine. The packet classification phase also requires a field-set of classification rules as input.

A. DoS Attack: Detection and Prevention with C-GEP

According to Akamai [1] the top 9 Internet (D)DoS threats are: **UDP flood, UDP fragmentation, TCP Syn flood, TCP ACK flood, CHARGEN, DNS, NTP, RIP and SSDP**. The anomaly detector stage of C-GEP supports the identification of these 9 volumetric attacks, with varying confidence-levels. The internal structure of the detector stage consists of a hash table-based rate-classifier, i.e., it detects and calculates the packet rate for each destination IP address hash.

The rate-classifier gives an output according to a configurable packet rate threshold. The next stage implements a disaggregation module that provides the destination IP addresses from the hash values. Finally, a detector stage calculates protocol-specific statistics, such as TCP flag statistics, variance of the source IP addresses, variance of the L4 payload, UDP flood and fragment statistics, and vulnerable protocol suite statistics, e.g., DNS, NTP.

To exemplify the operation and performance of the proposed scheme, we describe the detection of the UDP flood, DNS flood and TCP SYN flood attacks, respectively.

UDP flood is known as a hard-to-detect L3 attack, since it can force the denial of service in different manners. In order to detect UDP floods, C-GEP runs 5 algorithms in parallel, besides 2 other UDP fragmentation detectors:

1) *SRC detection:* determines the distribution of source addresses. This detector will detect the respective pattern in some cases where the attacker uses millions of spoofed IP source addresses to throw off common defenses.
2) *DST check:* checks if there is a valid service on the destination port. Most UDP floods target random unused ports to maximize its effectivity. In this case, the target will reply with an ICMP destination (port) "Unreachable" message, which can be a good indicator of a flood attack.
3) *Volumetric gradient:* measures the change of the traffic sent by the target under attack. The goal of a DoS attack is to decrease the amount of useful outgoing traffic. E.g., the sharp increase of incoming traffic and the correlated (potentially causated) decrease of the outgoing traffic can be a hint of an attack.
4) *RTT estimation:* calculates the round trip-time with ping messages. A successful DoS attack will render the target unresponsive, thus, the FPGA detector can use a separate link to send ICMP type 16 messages to determine the state of the target. This method, however, is a few magnitude orders slower than the rest; hence the detector will abort the test if the other algorithms yield a conclusive result before the completion of the RTT evaluation.
5) *Statistical assessment:* checks the traffic data variance. As anecdotal evidence shows, many attacks clone the same packets over and over, often to minimize the resource usage of the attacker. We include in Alg. 1 a pseudo-code of this algorithm since it was also proven effective against certain botnet/hacktivist tools.

DNS flood, our second example, is one of the most common type of reflection attacks. Our approach was to consider every unsolicited DNS response as part of an attack. We monitored the number of DNS requests and responses for each endpoint. If the number of responses exceeded the number of requests by a defined safety margin, the detector signals the attack. False negative detection is very unlikely for this threat type.

Algorithm 1 Pseudo-code snippet from the UDP flood detection module. After activation, the detector captures the first 200 packets with a destination IP address that appears under unusually heavy load. The first four bytes of the payload is used to create a 8-bit XOR checksum that is stored in a byte-array. Afterwards, the mean and variance of the checksums are calculated. If the variance is 0 or smaller than a preset limit, then we have a good indicator of an UDP flood attack.

```
for a = 0 to 200
   checksum[a] = pkt_payload[0] xor ... xor pkt_payload[3];
m = 0; var = 0;
for a = 0 to 200
   m += checksum[a];
m = m/200;
for a = 0 to 200
   var += (checksum[a]-m)*(checksum[a]-m);
return var/200;
```

If spoofed DNS requests are coming from outside the DCN they won't be routed to the detector. Due to the common DCN security policy spoofed messages originating internally from the datacenter are blocked on the local access switch.

Finally, for our third attack, i.e., the SYN flood detection consists of two algorithms: (1) one monitors the TCP flags of the suspicious flows, and (2) another one verifies the source addresses of the SYN messages. If the ratio of incoming SYN to ACK surpasses a certain threshold we register it as an attack. (2) monitors the variance of the source addresses of the incoming SYN messages looking for anomalies such as multiple copies of a SYN message coming from a single source.

Thus far our FPGA-based detection has been tested in the wild with real, live traffic of UDP flood, UDP fragmentation, DNS flood and SYN flood attacks. Each of the FPGA modules (i.e., packet parser, flow classifier and anomaly detector) could process at full 100Gbps. The enhanced SDN functionality of C-GEP could support multiple Tbps switches concurrently as long as their aggregated mirrored traffic does not exceed 100Gbps. Since a minimum-sized Ethernet frame fits into one 512-bit word, the pipeline stages must accept in every clock cycle a new incoming packet, at 100Gbps. Hence a processing speed of 1.5x10^8 PPS in each of the 4 stages. The clock within the FPGA device is limited to a few 100s MHz. A realistic upper limit is ca. 400MHz. This constraint is determined by the divided FIFO and cascadable Block RAM layout, besides the latency of the internal signal paths. The latter depends on the signal routes between the allocated physical elements. Accordingly, the width of the datapath is the primary design factor for FPGA-based packet processing at 100Gbps and above. Another performance factor is the physical resource requirement of the overall design. A reconfigurable chip has a given number of simultaneously usable logical elements, which sets a limit to the size, functionality and complexity of the FPGA implementation. The hardware resource requirements of the pipeline stages are detailed in Table II.

TABLE II
RESOURCE REQUIREMENTS FOR THE XILINX XC6VHX255T MODULES. 1ST NUMBER IS THE 100GBPS ETHERNET MAC, PACKET PARSER AND FLOW CLASSIFIER, 2ND NUMBER IS THE ANOMALY DETECTOR.

SLICE LOGIC	USED	PRESENT	UTILIZATION
Slice Registers	70646 / 13518	316800	22% / 4%
Slice LUTs	46208 / 16223	158400	29% / 10%
Occupied Slices	19280 / 6909	39600	48% / 17%
RAMB36E1/FIFO36E1s	99 / 43	516	19% / 8%
RAMB18E1/FIFO18E1s	61 / 0	1032	5% / 0%

TABLE III
ATTACK PATTERNS USED FOR VALIDATION AND PLATFORM SETUP

	Attack #1	**Attack #2**	**Attack #3**
Attack	UDP flood	DNS flood	SYN flood
Mirror	1:1	N : 1	N : 1
Duration	10 minutes	10 seconds	10 seconds
Replay tool	DPDK	10GED FPGA	10GED FPGA
Traffic mixing	HW	SW	SW
Attack Tput	8+ Gbps	7-8 Gbps	300-500 Mbps
Truncation	64 bytes	96 bytes	96 bytes

VI. EXPERIMENTAL RESULTS AND DISCUSSION

In order to validate our solution of a real-time SDN monitoring for security, we have tested it against real-world traffic with small-scale DoS attacks. The main attack occured in August 2016 and targeted the network of the National Information Infrastructure Development Institute (NIIFI) in Hungary, which provides internet access to academic and research institutions. Our data comes from lossless packet captures from a mirror of a 10Gbps link transfering general internet traffic, i.e., a North-South link of a small datacenter. The packets are truncated and all addresses have been anonymized for privacy.

We implemented a replay tool based on Intel DPDK [27] to support in-house testing with the collected data. The tool replays the packets according to their timestamp with an accuracy better than 3.5 microseconds for 99.99% of the packets, while also padding the truncated packets to their original size. It is about 5x faster than tcpreplay v3.4.4 with libpcap 1.7.4.

Our setup consisted of an Intel RRC switch with two hosts acting as traffic generators (replayers) and one C-GEP in the feedback loop. One of the traffic generators was replaying the traffic of normal operation, while the other overimposed the captured attack traffic. The traffic generator hosts were connected to single 10Gbps ports, while the C-GEP was connected to an aggregate of 4x 10Gbps monitoring ports[1] to receive the mirrored traffic. One management port was used to close the local feedback loop implementing the fast loop from Fig. 1.

A. Attack characteristics

Table III summarizes the attack patterns used for validation.

1) UDP Flood Attack: The captured data refers to a UDP flood attack, where 41 different sources attempted to flood a

[1] We used 4x10 Gbps, due to a lack of matching 100 Gbps transceivers at the time of the experiment.

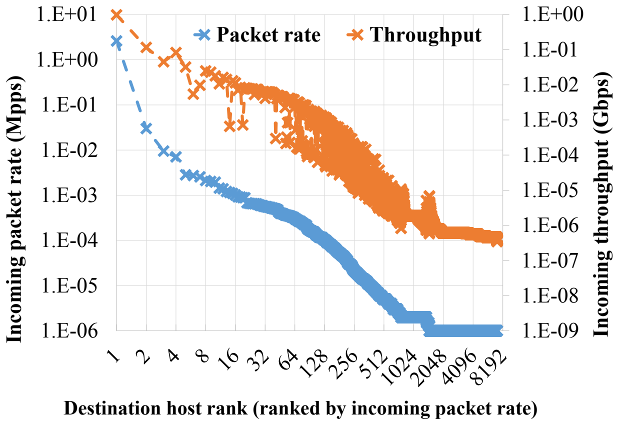

Fig. 4. Incoming packet rate and throughput for all destinations in the first 5 seconds of the UDP flood, ranked by packet rate.

Algorithm 2 The policer update service running on the switch, accepting commands from the FPGA

Main steps:

```
while running
  wait for command
  if ratelimit command received
    add rule
    send Ack
  if delete command received
    delete rule
    send Ack
```

add rule:

```
if rate != 0
  add policer
  add police rule to ACL
if rate == 0
  add drop rule to ACL
compile and apply ACL
```

delete rule:

```
find rule
if rate != 0
  delete policer
delete rule from ACL
compile and apply ACL
```

specific target within the NIIFI network with high rates of minimally-sized UDP datagrams. The attack did not target the network infrastructure (which had enough capacity to handle the traffic), but rather intended to cause the target host to be overloaded in processing the high incoming packet rate, possibly aiming at disrupting its service. The capture lasted for approximately 10 minutes. The data was captured from a 1:1 mirror port as the subset of the switch global mirror, because the concept of global mirroring was not yet deployed at the time of the attack. Fig. 4 shows the average incoming packet rate and throughtput for each destination during the first 5 sec. of the attack. The attack target is the host with the highest packet rate, ranked first among all destinations in this respect. The attack succeeds at overloading this host with a packet rate 87x higher than the next in the ranking order of packet rates. There were approx. 8.4x higher traffic volume targeted to the attacked host than the second most loaded host.

2) DNS Flood Attack: This attack was carried out against one of the hosts inside the NIIFI network, through DNS and NTP reflection amplification attacks. This attack was most likely carried out with a botnet, since the capture contained ICMP messages appearing to be botnet control signals. The pattern consists of 10 sec. of attack spikes, in which the attacker ramps up the traffic to 7-8Gbps for 3-10 sec. per attack burst. The attacker stops the transmission for 30-60 sec. between bursts. Since the first, 3 sec.-long spike was a pure DNS flood, we have reused it for testing the anomaly detector. We have mixed the attack pattern with ca. 10 sec. of normal traffic captured from the same link.

3) SYN Flood Attack: The captured SYN flood attack lasted for cca. 15 minutes. The data rate of the attack was around 500Mbps, sometimes dropping to 200-300Mbps. The attacker used millions of spoofed IP addresses. The first 5 sec. of the attack were mixed with 10 sec. of normal traffic: this pattern served as the 3-rd type of attack in our tests.

B. Implementation Details and Challenges

In order to mitigate the attack, the FPGA-based SDN controller implements filtering in the switch, using the *Access Control List (ACL)* and *Policer* features. The Intel RRC switch has 32 TCAM slices, each containing 1024 entries of 40-bit maskable keys. A configurable number of TCAM slices (50% by default) are reserved for L3 routing and other functions, while the rest are used to construct ACL rules. Logically the TCAM slices can be concatenated (i) in parallel to implement rules matching on multiple protocol fields, or, (ii) in serial to allow fitting more rules than the length of a TCAM slice into one ACL. Besides many other actions, the matching rules can drop a packet or send it to a policer. The switch has a total number of 9216 policers that can be used as simple counters, or as rate limiters, using a token bucket algorithm.

Here our FPGA worked as a signature-based anomaly detector and closed loop autonomous mitigator. However, instead of going through a centralized controller as in a typical SDN design, our solution can send "*Drop*" or "*Limit*" ACL rules directly to a service running on the switch's CPU, to minimize the loop lag. The service on the switch creates or deletes the rules and policers in the ASIC and then acknowledges back to the external FPGA (Alg. 2). Creating a *Drop* rule only adds an ACL rule, while creating a *Limit(*-ing*)* rule needs *also* a Policer. Inserting a rule is done in 2 or 3 steps. In case of a *Limit* rule, the first step is to allocate a Policer, next is adding the rule to an ACL, and finally invoking the ACL compiler to apply the changes. Rules are also provided with a timeout mechanism, so that they can be renewed for as long as needed and essentially, automatically dropped as soon as the attack ends.

C. Performance and discussion

The detector performance was measured inside the FPGA with an added module. The measurement module was tracking the time between the first attacking packet and the detection signal. The detection time depends on the rate of the attack and the attack type, as well as the internal state of the detector – which adds a small variance to the detection-time. Table IV presents peculiar values for 20 tests of each attack.

During the verification of the the different intrusion detection algorithm implementations, the effects were visualized as Fig. 5

TABLE IV
DETECTION TIME STATISTICS

Value	DNS Flood type attack	SYN flood type attack
Min. [us]	637	3925
Max. [us]	1123	4962
Avg. [us]	890	6284

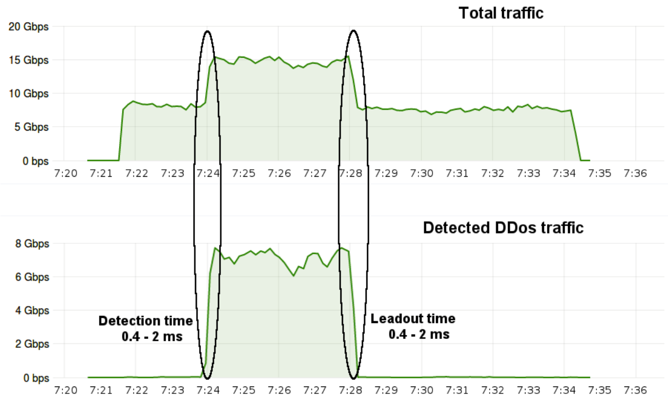

Fig. 5. Detection of a DNS attack: the attack lasted for cca. 4 seconds; detection was in the ms range.

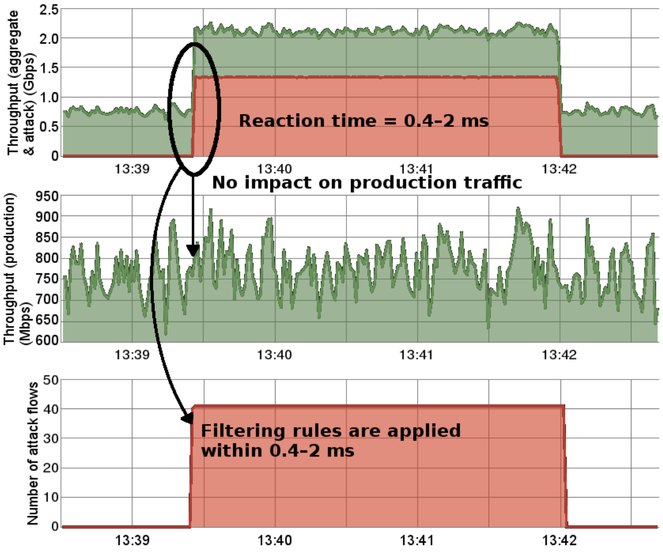

Fig. 6. Throughput and number of attack flows during the attack to NIIFI. Top: total volume of traffic (in green) and the volume flagged as attack (in red). Middle: production traffic that remains after filtering. Bottom: the number of detected attack flows. The attack has no visible impact to the production traffic because the reaction is much faster than the time resolution of the graphs.

shows for the DNS case. Our experiments show, that in case of an empty ACL, adding a filtering rule nondisruptively (*Drop*, based on source IP and L4 protocol and source port) takes between 400 us and 2 ms. In case of an ACL with 8K+ rules, it can take up to 35 ms. This is a lot faster than typical reaction methods that may take seconds or longer. The same operation with *Limit* (instead of *Drop* rules and independent policers) for each of the 8K rules takes altogether 95 ms.

The difference in the operational time may arise from the ACL compiler. Technically it could reuse the results of the previous compilation without trying to optimize, and it could be just as fast for a nearly full ACL as for an empty one. The current switch version, however, cannot seemingly optimize anything in nondisruptive mode, so we conjecture that the full recompilation with optimization is expensive. A possible solution for reducing the effect of the compilation overhead is batching, in which several outstanding insert/remove requests would be collected over a small period of time, and all changes would be applied at the same time, reducing the need to recompile to only once per batch. Fig. 6 graphically shows the results of the experiment utilizing *Drop* rules on the local feedback loop.

Although modern switches often have some built-in SYN flood and/or ICMP flood protection, we argue that when used for security, the local feedback loop complements and extends these primitives – adding the capability to detect any type of DoS attack with low reaction times and flexibility. The benefits of using an FPGA in the local feedback loop become thus evident in reconfigurability, while providing for finely-tuned *and* low-latency reactions.

VII. CONCLUSIONS

We addressed the new challenges of machine-driven cyber-attacks, designed to overwhelm the capabilities of today's standard security systems. Contrary to the current security practice, we aim to autonomously detect and mitigate in real-time also the ephemeral attacks directed at the human security operators. The average performance of our approach is 5-7 orders of magnitude higher than current solutions: (sub-)ms-scale vs. hour-scale – based on the automatic attack mitigation, instead of the typical human-in-the-loop.

Our first solution targets the most common 9 network attacks; it is universally compatible with most modern Tbps-class switches, needing neither DPI, nor the encryption keys. We prove that building an autonomous machine-driven anomaly detection and attack mitigation system for datacenter networks is feasible, and accordingly we designed a novel NFV that can uniquely and effectively counteract also the high-frequency ephemeral anomalies. For implementation at 10-100Gbps port speeds, we presented a practical solution to reconcile the performance-cost merits of the modern Tbps-class commodity switching ASICs, with the flexibility of native SDN/OF switches. This also shows that our scheme scales up with the *N:1 mirror* designed for the online monitoring of multiple 100Gbps ports.

We used 3 real-world attacks to prove the speed and accuracy of our new *autonomous* feedback loop – here implemented as an SDN-based online security application. Such attacks were detected in real-time and *automatically* mitigated within 430us up to 3ms – without loading the switch control plane and CPU, nor exceeding 60% of the FPGA utilization – and *without* human operator supervision. We argued that such SDN/NFV-enhanced switching fabrics could become essential for securing the next generation of SDNs for Cloud and datacenters.

References

[1] Akamai, https://www.akamai.com/us/en/multimedia/documents/state-of-the-internet/q4-2016-state-of-the-internet-connectivity-report.pdf.

[2] Corero, http://info.corero.com/rs/258-JCF-941/images/corero-q1-2017-ddos-trends-report.pdf.

[3] B. Veal and A. Foong, "Performance scalability of a multi-core web server," in *Proceedings of the 3rd ACM/IEEE Symposium on Architecture for networking and communications systems*. ACM, 2007, pp. 57–66.

[4] J. Rasley, *et al.*, "Planck: Millisecond-scale monitoring and control for commodity networks," *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 4, pp. 407–418, 2015.

[5] Y. Zhu, *et al.*, "Packet-level telemetry in large datacenter networks," in *ACM SIGCOMM Computer Communication Review*, vol. 45, no. 4. ACM, 2015, pp. 479–491.

[6] Arista, https://www.arista.com/assets/data/pdf/Datasheets/7500RDataSheet.pdf.

[7] Cisco, http://www.cisco.com/c/en/us/products/collateral/switches/nexus-7000-series-switches/data_sheet_c78-728423.html.

[8] Corsa, https://www.corsa.com/products/dp2400/.

[9] NoviFlow, http://noviflow.com/wp-content/uploads/NoviSwitch-21100-Datasheet.pdf.

[10] Mellanox Technologies, http://www.mellanox.com/related-docs/prod_npu/PB_NP-5.pdf.

[11] A. M. Caulfield, *et al.*, "A cloud-scale acceleration architecture," in *Microarchitecture (MICRO), 2016 49th Annual IEEE/ACM International Symposium on*. IEEE, 2016, pp. 1–13.

[12] A. Putnam, *et al.*, "A reconfigurable fabric for accelerating large-scale datacenter services," in *Computer Architecture (ISCA), 2014 ACM/IEEE 41st International Symposium on*. IEEE, 2014, pp. 13–24.

[13] S. Byma, *et al.*, "FPGAs in the cloud: Booting virtualized hardware accelerators with OpenStack," in *Field-Programmable Custom Computing Machines (FCCM), 2014 IEEE 22nd Annual International Symposium on*. IEEE, 2014, pp. 109–116.

[14] N. P. Jouppi, *et al.*, "In-datacenter performance analysis of a tensor processing unit," *arXiv preprint arXiv:1704.04760*, 2017.

[15] P. Varga and L. Gulyas, "Traffic analysis methods to support decisions at the Knowledge Plane," *Infocommunications Journal*, vol. 65, no. 4, pp. 50–56, 2010.

[16] J. Suh, *et al.*, "Opensample: A low-latency, sampling-based measurement platform for commodity SDN," in *Distributed Computing Systems (ICDCS), 2014 IEEE 34th International Conference on*. IEEE, 2014, pp. 228–237.

[17] M. Kuźniar, *et al.*, "What you need to know about SDN flow tables," in *International Conference on Passive and Active Network Measurement*. Springer, 2015, pp. 347–359.

[18] A. R. Curtis, *et al.*, "Devoflow: Scaling flow management for high-performance networks," *ACM SIGCOMM Computer Communication Review*, vol. 41, no. 4, pp. 254–265, 2011.

[19] ——, "Mahout: Low-overhead datacenter traffic management using end-host-based elephant detection," in *INFOCOM, 2011 Proceedings IEEE*. IEEE, 2011, pp. 1629–1637.

[20] P. Varga, *et al.*, "C-GEP: 100 Gbit/s capable, FPGA-based, reconfigurable networking equipment," in *High Performance Switching and Routing (HPSR), 2015 IEEE 16th International Conference on*. IEEE, 2015, pp. 1–6.

[21] P. Orosz, *et al.*, "C-GEP: Adaptive network management with reconfigurable hardware," in *Integrated Network Management (IM), 2015 IFIP/IEEE International Symposium on*. IEEE, 2015, pp. 954–959.

[22] J. D. Case, *et al.*, "Simple Network Management Protocol (SNMP)," United States, 1990.

[23] P. Phaal, *et al.*, "InMon corporation's sFlow: A method for monitoring traffic in switched and routed networks," RFC 3176, Tech. Rep., 2001.

[24] B. Claise, "Cisco systems NetFlow services export version 9," *Internet Engineering Task Force, Internet Draft*, 2004.

[25] N. Duffield, *et al.*, "Rule-based anomaly detection on IP flows," in *INFOCOM 2009, IEEE*. IEEE, 2009, pp. 424–432.

[26] J. Dromard, *et al.*, "Online and scalable unsupervised network anomaly detection method," *IEEE Transactions on Network and Service Management*, vol. 14, no. 1, pp. 34–47, 2017.

[27] http://www.dpdk.org/.