High-performance, platform-independent DDoS detection for IoT ecosystems

Alejandro Santoyo-González,
Department of Network Engineering
Universitat Politècnica de Catalunya
Barcelona, Spain
alejandro.santoyo@entel.upc.edu

Cristina Cervelló-Pastor
Department of Network Engineering
Universitat Politècnica de Catalunya
Barcelona, Spain
cristina@entel.upc.edu

Dimitrios P. Pezaros
School of Computing Science
University of Glasgow
Glasgow, G12 8QQ, Scotland
dimitrios.pezaros@glasgow.ac.uk

***Abstract*—Most Distributed Denial of Service (DDoS) detection and mitigation strategies for Internet of Things (IoT) are based on a remote cloud server or purpose-built middlebox executing complex intrusion detection methods, that impose stringent scalability and performance requirements on the IoT due to the vast amounts of traffic and devices to be handled. In this paper, we present an edge-based detection scheme using BPFabric, a high-speed, programmable data-plane switch architecture, and lightweight network functions to execute upstream anomaly detection. The proposed detection scheme ensures fast detection of DDoS attacks originated from IoT devices, while guaranteeing minimum resource usage and processing overhead. Our solution was compared against two widespread coarse-grained detection techniques, showing detection delays under $5ms$, an overall accuracy of $93 - 95\%$ and a bandwidth overhead of less than 1%.**

***Index Terms*—Internet of Things, DDoS, Anomaly Detection, Data-plane Programmability, BPFabric, Edge Computing**

I. Introduction

Internet of Things (IoT) is a fundamental pillar of future smart cities and next generation industries [1], and hence the amount of IoT devices and their variety (e.g., from smart TVs to virtual assistants and e-healthcare) [2] are increasing at a significant pace. Given the nature of IoT deployments where millions of end devices acquire networking capabilities, IoT-Distributed Denial of Service (DDoS) attacks (i.e., IoT devices forming botnets) have emerged as a challenge due to the number of current and forecasted devices in the network and their inability to be easily patched [3]. So far, solutions against DDoS attacks in this context have been implemented through complex, centralized software and hardware-based mechanisms [4]. Distributed detection and mitigation techniques have been studied, aiming at offering more efficient ways of dealing with scenarios such as IoT-DDoS attacks [5]–[7]. The vast majority of these approaches assume the use of purpose-built middleboxes deployed in the network, close to the victim, in order to be able to detect the attack through analyzing aggregated traffic features [3][6][8]–[10]. However, in an IoT environment, the cost of a purpose-built ecosystem to detect and mitigate DDoS poses complex deployment and operational challenges, for instance, if early detection and high-speed processing is to be achieved. Another key issue in IoT-DDoS defense is the ability and frequency of spoofed IP addresses being used by the attackers [7]. Any remote detection mechanism lacks the ability to effectively detect if a packet source IP has been spoofed, without detailed information about the sender, receiver or typical communication patterns. On the contrary, upstream detection executed as close to the attacker as possible, could accurately determine spoofing occurrences due to its fine-grained view of the underlying devices, IP subnets and typical behavior.

What is more, most current DDoS detection schemes rely on traffic redirection methods or aggregated flow statistics collection. These mechanisms introduce additional costs and performance issues into the network (e.g., longer flow completion times, bandwidth overhaul, etc.), degrading the system's effectiveness due to longer timeframes between detection and mitigation phases [11]. To partially overcome such problems, multi-stage distributed systems can be used to detect and mitigate the attacks. In these approaches, coarse-grained detection is to be executed upstream in the network, closer to the attackers. However, this leads to the use of dedicated middleboxes scattered across the network for scrubbing purposes [9][6]. For an IoT-DDoS detection solution (i.e., protecting the network against DDoS originated on IoT devices) to solve the above mentioned problem, it has to ensure: **a) lightweight processing**, by relying on traffic features and analysis methods targeting overhead minimization and coarse-grained anomaly detection; **b) platform-independence**, to minimize the need for purpose-built devices and the use of traffic redirection-based approaches; and **c) high-performance**, in order to achieve fast reaction through early detection while avoiding performance degradation.

The advent of paradigms such as Edge Computing (EC) and Software-Defined Networking (SDN) can help overcome the aforementioned issues for DDoS detection and mitigation. EC brings processing, networking and storage resources to the users' vicinity. As a result, through SDN data-plane programmability principles, lightweight functions can be placed at

This work has been supported by the Ministerio de Economía y Competitividad of the Spanish Government under the project TEC2016-76795-C6-1-R and AEI/FEDER, UE. Additionally it has been supported in part by the UK Engineering and Physical Sciences Research Council (EPSRC) projects EP/R511936/1, EP/N033957/1, and EP/P004024/1; by BT (Voucher No. 17000117); by the Huawei Innovation Research Program (Grant No. 300952); and by the European Cooperation in Science and Technology (COST) Action CA 15127: RECODIS – Resilient communication and services

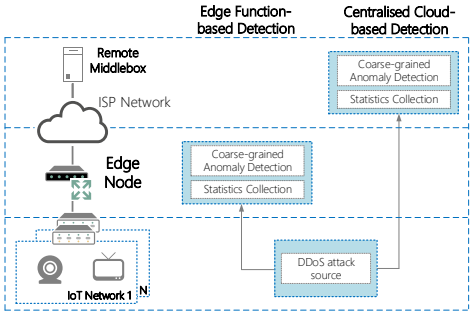


Fig. 1. Centralised Cloud vs. Edge Functions-based Detection

the edge, resulting in enhanced network capabilities. Namely, a programmable data-plane improves the network's agility and flexibility by allowing dynamic high-speed edge function allocation/deallocation. Thus, providing early detection capabilities and more efficient resource usage at the edge nodes. Additionally, edge network functions deliver the elasticity and scalability required to efficiently handle vast amounts of traffic in a distributed manner. IoT can directly benefit from the joint work of data-plane programmability and edge network functions. By placing the detection at the attackers' vicinity (cf. Fig. 1), fast reactive procedures can effectively isolate the IoT malicious devices while reducing bandwidth consumption typically produced by DDoS attack traffic, and avoiding the processing overhead of current remote centralized detection approaches. In summary, our solution outperformed two widespread coarse-grained detection techniques, showing significantly lower detection delays and attack penetration levels, while achieving an overall higher accuracy values.

In this paper, we present a lightweight, platform-independent anomaly detection mechanism to be deployed at the edge of the network (e.g., a home or IoT gateway). To achieve true platform independence while ensuring high-performance levels, our proposal is based on BPFabric, a data-plane programmability architecture presented in [12]. In a nutshell, the BPFabric platform allows to program the data-plane of SDN network nodes and therefore, it can be partially considered complementary to other solutions, e.g., P4[1]. Unlike the latter, however, BPFabric focuses on high-speed processing and, for that reason, it is based on the Extended Berkeley Packet Filtering (eBPF) [13] instruction set, rather than a higher-level Domain Specific Language (DSL). Previous work has successfully tested the line-rate capabilities of eBPF [12]–[14]. Namely, it has been demonstrated that eBPF-based packet processing, by acting at the socket level, significantly improves both throughput and latency, while still offering the advantages of kernel integration (i.e., full network stack processing) when required. BPFabric provides true platform-independent execution on account of eBPF, avoiding the PISA[2]-based device restriction imposed by P4. Furthermore, BPFabric goes beyond the data-plane programming capabilities of P4 by defining a fully developed architecture specifying: the SDN controller and remote agent behaviors, the controller-agent interactions, mandatory core packet processing functionalities and message exchange procedures. The framework hence allows to defined and deploy diverse network functions as part of the forwarding behavior of each switching element, from a remote centralized location.

Overall, our main contributions are: 1) the design and implementation of a lightweight, platform-independent anomaly detection mechanism based on edge functions defined as part of the BPFabric architecture, exploiting SDN-based data-plane programmability, 2) the implementation of an eBPF-based detection method using Shannon's Entropy and Exponentially Weighted Moving Averages (EWMA) and, 3) the evaluation of our edge-based anomaly detection scheme against a fully centralised cloud-based approach considering carefully selected traffic features (cf. Section II) matching the particularities of traffic anomalies in IoT ecosystems.

II. IoT-DDoS anomaly detection

Typical DDoS attacks are said to be characterized by high frequency of incoming packets, endpoint communications asymmetry, and high number of source IP addresses [6][9]. However, such characteristics are directly linked to a close-to-target detection approach and fail to describe IoT-DDoS if they are to be detected at the attackers' vicinity [15]. For instance, by pushing the detection mechanism to the network edge, the benefits from traffic aggregation are lost and hence, outliers such as high packet rate/volume and source IP diversity cannot be considered. For a joint scenario mixing IoT and upstream attack identification, a tailored set of metrics is required. The works in [9][10][15][16] provide a solid baseline for a set of metrics in order to identify anomalous traffic in IoT environments. Based on these findings, our set of IoT-DDoS detection parameters is summarized below:

Destination/Source IP Address Distribution: given their reduced functionality scope, IoT devices usually communicate with a small set of endpoints. Therefore, anomalous traffic can be identified by analyzing the destination IP address distribution [9][10]. Furthermore, DDoS attacks usually employ forged source IPs to communicate with a victim host. Therefore, to effectively identify abnormal traffic, the destination/source address space entropy can be used. According to the findings from [10], IoT devices should mostly have an overall low entropy. As a consequence, any change in the entropy value over a given timeframe can be considered a sign for an ongoing attack.

Flow Asymmetry: during a DDoS attack, the interaction between the attacker and the target has been found to be asymmetrical [9]. Under a DDoS attack from an IoT botnet, the underlying IoT devices send a high number of requests to the victim. Eventually, the target capacity is exceeded and the symmetry of outgoing requests and incoming responses is affected, a situation that can be identified by detection methods placed at the attacker's vicinity. To use traffic asymmetry as a detection feature, the method presented in [9] is adapted and used in this paper (cf. Section III).

[1]https://p4.org/

[2]Protocol Independent Switch Architecture

Inter-packet Interval: within the time domain, the traffic patterns of IoT devices are often quite stable with each device sending information to, for example, remote control systems at clearly pre-defined, arbitrary, and immutable time intervals. In contrast to regular IoT traffic, DDoS attack incoming traffic from an IoT device is often characterized by high burstiness in short and periodic timeframes [8][10][15].

Packet Size: under a DDoS attack, the packet size distribution for IoT devices varies greatly over time. Typically, malicious traffic comprises bursts or steady flows of incoming small packets around 100 bytes, while normal traffic packet sizes are unevenly distributed from 100 to more than 1000 bytes [10]. This behavior allows us to detect anomalous traffic by analyzing the packet size variation -i.e., number of packets with length under 100 bytes- over arbitrary controlled timeframes.

Packet Volume: the transferred data volume is a key parameter when detecting volumetric DDoS [10]. Given the reduced and typically fixed amount of traffic periodically sent by IoT devices, analyzing the packet volume a the network edge can effectively lead to detect an ongoing attack.

A. Edge function-based detection

Leveraging SDN data-plane programmability and EC principles, coarse-grained detection mechanisms can be deployed at the network edge close to potential attackers (i.e., IoT devices). This can be achieved through in-line edge functions and technologies tailored to the edge node resources and characteristics. BPFabric allows functions to be implemented at the network edge, encoded as part of the data-plane behavior of the device (e.g., a switch). Therefore, BPFabric provides the added flexibility of being able to deploy the system on a wide variety of devices already in use at the user's vicinity (e.g., home gateways, access routers, etc.).

When selecting the anomaly detection mechanism, the inherent limitations of the edge nodes (e.g., limited resources, rigid programmability, etc.), the goal of achieving line-rate performance to avoid throughput or latency degradation, for instance, and the need for fast detection, significantly reduce the list of mechanisms that can be used. For instance, complex detection techniques based on machine learning require high processing capabilities unavailable on the data path of an edge node. Moreover, the use of BPFabric, based on the eBPF instruction set, introduces additional particularities (e.g., limited program size) that should be taken into account in order to achieve high-speed and bound execution time. Taking the above into consideration, the coarse-grained detection running at the edge is forced to be fairly simple, while still ensuring an adequate level of accuracy. In an IoT context, we believe such tradeoff can be sorted out through adequate traffic statistics collection combined with multi-feature analysis. Such an idea is supported by the nature of the attack traffic characteristics identified above (cf. Section II). For example, let us consider a DDoS TCP SYN flood attack with variable packet burstiness and overall low packet volume. Through a joint evaluation of the volume, destination IP addresses and inter-packet intervals, the attacker could be pinpointed. This is possible because the anomaly detection system is able to conclude that, for instance, for a certain target IP, the traffic burstiness and packet volume (e.g., average packet size under 100 bytes) do not follow the expected behavior.

Nevertheless, as the detection method is forced to be simple and even following the above multi-feature analysis approach, the detection accuracy will highly depend on the attack complexity and the dynamic adjustment of the mechanisms (e.g., thresholds) used to detect a suspicious event. To overcome these limitations and based on the promising results found on previous work [5]–[7], we envision a multi-stage detection architecture where the coarse-grained detection is carried out close to the attackers, and the upper and more advance analysis layers can be executed in more powerful edge nodes scattered within the service provider network (either in a centralized or distributed fashion). Overall, the idea behind such scheme is to periodically collect traffic information through eBPF filtering rules on the IoT network gateways (cf. Fig. 1). The detection analysis is carried out through a pipeline of condition evaluation steps injected into the IoT gateway running BPFabric (the data collection is also part of the eBPF program inserted). If any anomalous behavior is found, an alarm is then sent to the upper detection layers on the architecture via the controller (assuming an SDN implementation). In case an anomaly is found, the upper layer executes further processing (after requesting additional information if required) and confirms if an attack has been made. In the event of a false positive, the detection parameters on the coarse-grained mechanism are to be adjusted to increase its accuracy.

In order to comply with the above mentioned system limitations, we decided to use Exponential Weighted Moving Average (EWMA) and Shannon's Entropy for outlier detection. After the data collection interval finishes, the EWMA (according to Eq. 1) is calculated for the following features: packet count, rate, volume, and size distribution. Flow completion (for the traffic asymmetry feature) is determined through source/destination IP address pairs, keeping track of the number of outgoing communications and the associated responses. Finally, the endpoint/source variation over time is checked based on the source/destination IP entropy (referred to as $H(X)$) calculated using Eq. 2 [6].

$$EWMA = \alpha \cdot value + (1 - \alpha) \cdot last\_prediction \tag{1}$$

$$H(X) = -\sum_{i=1}^{N} p_i \log_2 (p_i) \tag{2}$$

III. Performance Evaluation

We conducted experiments to evaluate the suitability of the proposed EWMA and Shannon's Entropy-based detection within BPFabric architecture (for convenience this method is hereinafter referred to as "ESE-Detection") to effectively detect IoT-DDoS. The testbed used in our experiments is shown in Fig. 2. A set of IoT networks is emulated, connected through access routers (AR) to the Wide/Metropolitan Area

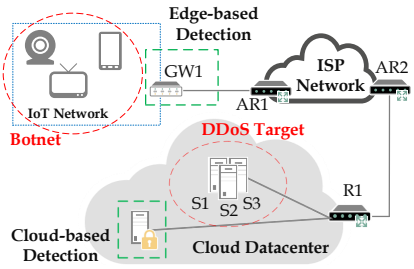


Fig. 2. Testbed architecture used for the experiments

Network (WAN/MAN), and finally to the remote cloud where the attack targets are located. The metrics for the scenario analysis are presented in Table I. They are selected in order to thoroughly assess the behavior of our solution and its overall performance. The detection pipeline collecting traffic data and executing EWMA and Shannon's Entropy-based detection is run in node GW1 (i.e., the IoT gateway). The cloud-based detection is executed within the emulated remote cloud collecting traffic data from node R1. Two additional detection methods were implemented for evaluation purposes: Cosine Similarity [8] and Shannon's Entropy [6][17]. Both detection strategies were adapted to use the metrics presented in Section II, and were selected considering their previous use in coarse-grained DDoS detection [6][8]. Since a thresholding approach was adopted for all detection strategies, the methodology presented in [17] was used to optimize the threshold selection process and enhance the overall accuracy. To emulate an attack, we developed a Python script using the Scapy library[3] to generate spoofed source address and destination ports, targeting an arbitrary server within the remote cloud in Fig. 2 simulating a TCP SYN flood attack.

To generate the IoT traffic for the experiments, the tool "Distributed Internet Traffic Generator" [18] was selected due to its flexibility and granularity in controlling the traffic characteristics. Furthermore, the findings presented in [19] and [20] allowed us to model IoT traffic of a home network, assuming each setup comprises the following elements: 3 smart appliances (e.g., refrigerators, washers), 4 climate control sensors and 6 lighting control devices. The values in Table II were assumed to generate the device data flows and model the device normal behavior. Mininet[4] was employed to emulate the IoT network due to its simplicity and flexibility (cf. Section IV). Overall, a round-trip delay of 100 ms was assumed for the end-to-end communication from the IoT networks to the servers in the remote cloud, accounting for the processing, routing/switching, and propagation delays involved.

Estimating the entropy of the IP distribution was quite challenging considering the limitations of the envisioned underlying hardware (e.g., no support for float point operations) and the eBPF instruction set characteristics. Consequently, Eq. 2 was adapted to overcome these restrictions. To efficiently find the base 2 logarithm we adapted the Taylor Series expansion method described in [21], hence approximating the base 2 logarithm through Eq. 3. The K constant value defined in Eq. 4 was calculated beforehand and predefined in the eBPF program.

$$\log_2(\frac{x}{y}) = K \cdot (-\log(\frac{x}{y})) \qquad K \in \mathbb{R}, \quad x, y \in \mathbb{Z} \tag{3}$$

$$K = \frac{-1.0}{\log(2.0)} \tag{4}$$

Eq. 5 allowed us to effectively approximate the logarithm of x/y (e.g., x: destination IP count, y: total destination IPs).

$$\log(\frac{x}{y}) = a + \frac{a^2}{2} + \frac{a^3}{3} + .. + \frac{a^n}{n} \quad a \in \mathbb{R}, \quad n \in \mathbb{Z} \tag{5}$$

$$\text{Where:} \qquad a = \frac{(y-x)}{y} \tag{6}$$

$$\frac{\begin{array}{l} y^{N-1} \cdot (2 \cdot 3 \cdot 4 \cdot ... \cdot N) \cdot (y-x) + \\ \quad + y^{N-2} \cdot (1 \cdot 3 \cdot 4 \cdot ... \cdot N) \cdot (y-x)^2 + ... \end{array}}{y^N \cdot (1 \cdot 2 \cdot 3 \cdot .. \cdot N)} \tag{7}$$

Unrolling and computing Eq. 5 as a running product allowed us to handle all operations using integers and comply with the unbounded loop restrictions in eBPF. As a result, $-\log(x/y)$ was implemented as shown in Eq. 7. Where N is an arbitrarily chosen integer ($N = 2$ was empirically selected, the estimation error analysis is shown in Fig. 3) providing the desired precision. Finally, to compute the entropy for the destination IP variation, for instance, fast map iteration through eBPF `bpf_map_get_next_key`[5] was employed and eBPF maps were used as immutable global counters when needed. To enhance the accuracy while avoiding register overflow (i.e., likely to occur for large N values), we decided to multiply the numerator by 1000. Consequently, the entropy estimation resulted in an integer comprising up to three of the decimal values of the real result (e.g., for an entropy equal to 1.123, the estimated entropy found was 1120). The performance and resource overhead is minimized by removing user-kernel space interactions, as all computations on the detection pipeline are executed within kernel space. The bandwidth analysis depicted in Fig. 4 shows the minimum performance impact of ESE-Detection processing, causing a bandwidth reduction of around 1%.

Although the estimation error is thoroughly described in [21], we decided to conduct experiments to determine the impact of the entropy estimation over the detection accuracy. The findings can be observed in Fig. 3 where the cumulative step histogram for the estimated entropy error is shown. Overall, the estimation error fell most of the times within 1% to 3% for typical IoT traffic and within 6% to 11% in the event of an attack. This precision level gave us sufficient margin to effectively determine an anomaly was occurring. The estimation error was significantly lower than the error

[3]https://scapy.net/

[4]http://mininet.org/

[5]http://man7.org/linux/man-pages/man2/bpf.2.html

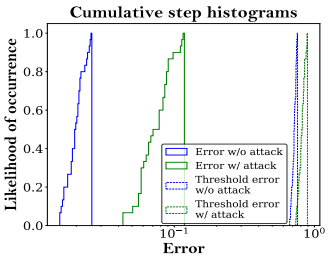


Fig. 3. Entropy estimation error.

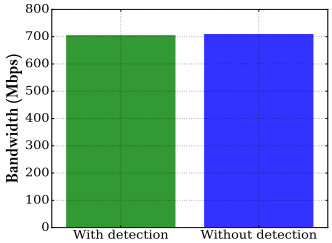


Fig. 4. Bandwidth consumption with and without detection processing.

TABLE I
EVALUATION PARAMETERS

Parameter	Description
Bandwidth	Bandwidth overhaul caused by traffic redirection or data collection in centralized cloud-based approaches.
Detection time	Time elapsed between the anomaly occurrence until an alarm is raised.
Attack penetration	Amount of anomalous traffic (attack packets) inserted into the network until an alarm is raised.
Accuracy	The false positive and false negative ratio achieved.
Cost	Overall expenses based on [22]: cost of information gathering, data processing and detection implementation.

TABLE II
EVALUATION PARAMETERS

Feature	Smart Appliance	Climate & Lighting Control
Dst. IPs	$2 \sim 10$	$2 \sim 5$
Num. Dst. Ports	$2 \sim 5$	$2 \sim 5$
Avg. Load (Kbps)	$5 \sim 25$	$5 \sim 15$
Packet Size (bytes)	$100 \sim 600$	$100 \sim 200$
Act./Idle intervals (s)	$2 \sim 10$ / $80 \sim 100$	$2 \sim 5$ / $10 \sim 20$

required to incur in a miss-detection, represented by the rightmost dashed lines in Fig. 3, for both regular and attack traffic. In order to result in a false positive/negative, our estimation error should have been at least 60% higher in any case.

Fig. 5a shows the cumulative step histogram for the detection delay for each of the implemented schemes. As expected, the performance of the BPFabric approach is significantly better due to the low processing delay introduced by the enhanced data plane pipeline. The eBPF-based detection engine is able to reduce over 80% of the anomaly identification time when compared to both Entropy and Cosine Similarity. Such results show the potential of BPFabric for early anomaly detection. Powered by its high-speed and lightweight processing potential, BPFabric-based detection is capable of drastically reduce the data processing overhead, thus resulting in significantly lower detection timescales. Since the BPFabric detection is running in kernel space, an inherent limitation is the lack of access to a proper timer due to the absence of an eBPF in-kernel function for this purpose (within the scope of the eBPF program type we are running). As a solution, the timing is followed using the incoming packet timestamps provided by BPFabric.

The accuracy of the detection methods was measured using the typical False Positive Ratio (FPR) and False Negative Ratio (FNR) definitions, as shown in Equation 8 and Equation 9. Maintaining per-flow data statistics using in-kernel processing on an resource-constrained IoT gateway is unfeasible due to: memory requirements to hold the generated data in the event of an attack, eBPF map limitations and performance degradation due to longer processing timeframes. Consequently, the detection analysis was not performed considering the benign/malicious flow count. Instead, we decided to run several experiments executed at both fixed and random time intervals, in order to emulate a more realistic botnet scenario, while measuring the accuracy through the number of attacks detected by the implemented methods. The results are presented in Fig. 5b and Fig. 5c, where the number of executed attacks is depicted, alongside the attacks detected by each algorithm.

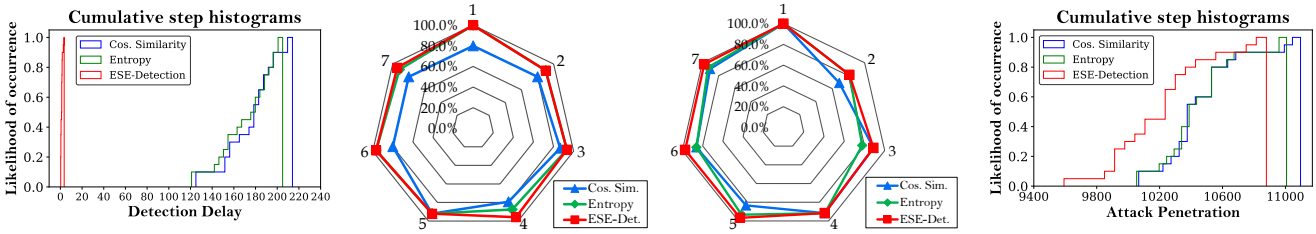
$$FPR = \frac{FP}{(FP + TN)} \quad (8) \qquad FNR = \frac{FN}{(FN + TP)} \quad (9)$$

Where:

- FP: Number false alarms
- FN: Number of undetected attacks
- TN: True benign traffic
- TP: Number of detected attacks

Throughout our experiments, ESE-Detection had an average accuracy of around 95%, superior to the entropy and cosine similarity strategies (93% and 83%, respectively) for the experiments where the attacks were executed at fixed intervals (cf. Fig. 5b). For the randomly timed SYN flood attacks, ESE-Detection outperformed again the remaining methods, achieving an average of 93% versus 88% and 86% of entropy and cosine similarity respectively (cf. Fig. 5c). Overall, ESE-Detection superior accuracy can be expected for this ecosystem given its segregated view of the traffic (i.e., detection executed closer to the attackers). Conversely, the typical traffic patterns of IoT devices cannot be effectively analyzed through cloud-based scrubbing due to the traffic convergence. Some interesting facts were found when estimating FNR/FPR. The ESE-Detection engine was able to ensure less than 20% FNR for both fixed and randomly timed attacks in the worst case scenario, surpassing the maximum of 50% found for cosine similarity and entropy. On the contrary, both these methods performed slightly better, overall, than ESE-Detection regarding the FPR. ESE-Detection reached a maximum of 33%, equaling the cosine similarity results and below the 25% reached by the entropy method. However, ESE-Detection showed higher FPR values in more experiments when compared to the tested strategies. This is actually expected, because of the EWMA limitations, i.e., the time it takes for the moving average to adapt to significant changes in the input data. A solution to this problem is to force the upper layers of the detection architecture to continuously monitor and update the analysis thresholds.

The attack penetration was tested to determine how much malicious traffic could be injected into the network before an alarm was raised by the detection engine. In Fig. 5d, the results show that the BPFabric edge-based approach performed better than both cloud-based methods. Less attack packets were



(a) Detection delay analysis.

(b) Accuracy for attacks executed at fixed intervals (number of attacks varying from 1 to 7).

(c) Accuracy for attacks executed at random intervals (number of attacks varying from 1 to 7).

(d) Attack penetration (in packets).

Fig. 5. Evaluation results for the proposed methods: ESE-Detection, Entropy and Cosine Similarity.

inserted into the network, due to the lower detection delays of BPFabric detection. Moreover, attack penetration values are directly linked to the in-place mitigation strategies. Consequently, BPFabric-based early detection provides the network with enhanced flexibility and efficiency in reducing the amount of attack traffic, by allowing upstream mitigation procedures to be executed in a fast and reactive manner. Similar results were obtained when evaluating the bandwidth consumption, i.e., the network capacity required by the detection method to collect and analyze the data. This metric was measured checking the data message size sent to the detection algorithms by the collecting device (R1 and GW1 in Fig. 2). For a detection interval of $30s$, the entropy mechanism employed an average of 3.3 MB of data while the cosine similarity was fed with around 1 MB. Conversely, BPFabric underlying ESE-Detection collected all required traffic statistics at line-rate on the IoT network gateway. Overall, BPFabric edge-based detection avoids the need to continuously poll traffic counters from the network nodes, thus preventing unnecessary bandwidth usage and enhancing scalability.

From the aforementioned results, the BPFabric edge-based detection approach stands out as the less costly solution to implement and deploy, when compared to adding a dedicated detection server/middlebox at the remote cloud or even paying for anomaly detection as a service. In a nutshell, BPFabric significantly decreases core operational/capital costs (e.g., power, cooling, processing/networking hardware), and allows an administrator or orchestration entity to easily and remotely control upstream packet processing and detection mechanisms for a significantly large number of nodes with minimum effort and low error rate. Regarding the monitoring and analysis expenses, the edge-based detection through BPFabric clearly outperforms the cloud-based scheme, as it introduces almost null overhead into the network while ensuring line-rate performance even for demanding scenarios and infrastructures.

IV. Discussion

As mentioned above, the proposal of a BPFabric-based detection scheme is assumed to be part of a multi-stage distributed detection/mitigation architecture. In such scheme, the fine-grained detection level would run at the existing edge nodes and would engage after receiving an alarm notification message from the coarse-grained detection phase. A platform such as the one presented in [6] could be a solid foundation, as it ensures a collaborative detection strategy assuming the use of edge nodes located at different network levels. The integration of our solution into this architecture would be straightforward. BPFabric is to be placed at the core of the architecture with a controlling role over the network functions to be deployed. Such network function goals are twofold: anomaly detection through the solution presented in Section II, and flow data collection for fine-grained detection when needed. To collect the required statistics, no additional processing overhead is involved as it could be easily implemented through an eBPF pipeline at each edge node. The communication between the detection stages would be managed by the controller, while the fine-grained detection stage could be engaged using one of two methods: a) a message sent by the controller (easier to implement but with certain performance issues regarding delay); b) through a direct message sent by the BPFabric agent (harder to implement as it would imply recoding the agent and implementing a low-level direct message exchange mechanism).

In order to ensure the applicability of BPFabric for anomaly (DDoS in this case) detection, another challenge to be overcome is the need for a high-performance implementation of the detection method developed (e.g., using the Intel DPDK framework). This would help to thoroughly verify the packet processing overheads introduced by the detection pipeline and the real throughput supported by the solution. Additionally, such implementation would allow a thorough performance profiling of the entropy calculation, thus becoming a baseline for further developments where the entropy method could be used to analyze the behavior of other traffic features. Furthermore, the integration of the solution into a multi-stage detection/mitigation architecture remains an open question (key guidelines are presented in the next section) and would significantly improve the practicality of the present work.

V. Related Work

The security issues of IoT networks, specially DDoS attack detection and mitigation, have been thoroughly studied [3][8][10][23]. Additionally, numerous papers have tackled SDN-based attack detection/mitigation, leveraging the benefits of data-plane programmability and control/data plane decoupling [4]. However, there is no research that does so, while ensuring real platform-independence, upstream execution, and line-rate performance.

Authors in [8] and [24] propose two frameworks to detect and mitigate DDoS attacks in IoT environments. The article in [10] presents a machine learning (ML)-based approach to DDoS detection originating from IoT devices. A similar solution is proposed in [23], where ML is used at the network controller level. A remote cloud-based detection strategy is followed in all of the above papers, which consequently lack early detection capabilities, and result in additional network performance degradation and longer timescales until an alarm is raised. Moreover, it requires dedicated resources (e.g., servers) to carry out the detection analysis. On the contrary, our aim is to distribute the early detection stage and place it at the network edge (cf. Fig. 1), to be executed by virtually any available edge device (e.g., IoT gateway), as a result of the platform independence ensured by BPFabric. This approach results in lower detection times and attack penetration levels, therefore adding early detection capabilities to the network (cf. Section III).

The authors in [3] move the traffic collection and detection stages to the attackers' vicinity. Overall, the authors claim that 10 times faster detection can be achieved following this approach, thus eliminating around 80% of the attack traffic injected into the network. A critical limitation of this work is that current cloud-based solution guidelines cannot be followed at the edge due to the lack of traffic aggregation and scalability issues. Namely, processing the IoT traffic requires a significant number of middleboxes scattered within the network in order to provide early detection capabilities, while avoiding performance degradation. Additionally, another issue in this study is the limited set of detection parameters, affecting the results accuracy and the ability to detect complex attacks. To overcome these limitations, we proposed an IoT-tailored set of detection features (cf. Section II) and avoid the use of dedicated resources to perform early and high-speed detection through an enhanced data-plane defined in BPFabric.

VI. Conclusion

In this paper we have presented lightweight, platform-independent and high-performance DDoS detection architecture for IoT ecosystems, based on the BPFabric programmable data plane. We have shown how DDoS detection in IoT can benefit from upstream located mechanisms. The use of BPFabric and eBPF-based detection minimizes the overall network overhead and provides effective early detection capabilities. Our results show that the proposed solution introduces a bandwidth reduction of less than 1% and reduced several times the detection delay when compared to other methods. Moreover, the overall accuracy of our strategy was at least 5% higher than the other evaluated mechanisms.

References

[1] E. Ahmed *et al.*, "Internet-of-things-based smart environments: state of the art, taxonomy, and open research challenges," *IEEE Wireless Communications*, vol. 23, no. 5, pp. 10–16, 2016.

[2] D. Hanes, G. Salgueiro, P. Grossetete, R. Barton, and J. Henry, *IoT Fundamentals: Networking Technologies, Protocols, and Use Cases for the Internet of Things*. Cisco Press, 2017.

[3] K. Bhardwaj, J. C. Miranda, and A. Gavrilovska, "Towards IoT-DDoS prevention using edge computing," in *USENIX Workshop on Hot Topics in Edge Computing (HotEdge 18)*. USENIX Association, 2018.

[4] N. Z. Bawany, J. A. Shamsi, and K. Salah, "DDoS attack detection and mitigation using SDN: Methods, practices, and solutions," *Arabian Journal for Science and Engineering*, vol. 42, no. 2, pp. 425–441, 2017.

[5] A. Pamukchiev, S. Jouet, and D. P. Pezaros, "Distributed network anomaly detection on an event processing framework," in *2017 14th IEEE Annual Consumer Communications Networking Conference (CCNC)*, 2017, pp. 659–664.

[6] M. Iordache, S. Jouet, A. K. Marnerides, and D. P. Pezaros, "Distributed, multi-level network anomaly detection for datacentre networks," in *2017 IEEE International Conference on Communications (ICC)*. IEEE, 2017, pp. 1–6.

[7] S. Simpson *et al.*, "An inter-domain collaboration scheme to remedy DDoS attacks in computer networks," *IEEE Transactions on Network and Service Management*, vol. 15, no. 3, pp. 879–893, 2018.

[8] D. Yin, L. Zhang, and K. Yang, "A DDoS attack detection and mitigation with software-defined internet of things framework," *IEEE Access*, vol. 6, pp. 24 694–24 705, 2018.

[9] H. Liu, Y. Sun, and M. S. Kim, "A scalable DDoS detection framework with victim pinpoint capability," *Journal of Communications*, vol. 6, no. 9, 2011.

[10] R. Doshi, N. Apthorpe, and N. Feamster, "Machine learning DDoS detection for consumer internet of things devices," in *2018 IEEE Security and Privacy Workshops (SPW)*. IEEE, 2018, pp. 29–35.

[11] C. Zhang and R. C. Green, "Communication security in internet of thing: Preventive measure and avoid ddos attack over iot network," *Simulation Series*, vol. 47, pp. 8–15, 2015.

[12] S. Jouet and D. P. Pezaros, "BPFabric: Data plane programmability for software defined networks," in *2017 ACM/IEEE Symposium on Architectures for Networking and Communications Systems (ANCS)*. IEEE, 2017, pp. 38–48.

[13] D. Scholz, D. Raumer, P. Emmerich, A. Kurtz, K. Lesiak, and G. Carle, "Performance implications of packet filtering with linux eBPF," in *2018 30th International Teletraffic Congress (ITC 30)*. IEEE, 2018, pp. 209–217.

[14] M. Xhonneux, F. Duchene, and O. Bonaventure, "Leveraging eBPF for programmable network functions with IPv6 segment routing," in *Proceedings of the 14th International Conference on emerging Networking EXperiments and Technologies - CoNEXT '18*. ACM Press, 2018, pp. 67–72.

[15] M. Antonakakis *et al.*, "Understanding the mirai botnet," in *USENIX Security Symposium*, 2017, pp. 1092–1110.

[16] P. Redekar and M. Chatterjee, "Hybrid technique for DDoS attack detection," *International Journal of Computer Science and Information Technologies*, vol. 8, pp. 377–379, 2017.

[17] K. Singh, K. S. Dhindsa, and B. Hushan, "Threshold-based distributed DDoS attack detection in ISP networks," *Turkish Journal of Electrical Engineering & Computer Sciences*, vol. 26, no. 4, pp. 1796–1811, 2018.

[18] A. Botta, A. Dainotti, and A. Pescapé, "A tool for the generation of realistic network workload for emerging networking scenarios," *Computer Networks*, vol. 56, no. 15, pp. 3531–3547, 2012.

[19] I. Cvitić, D. Peraković, M. Perisa, and M. Botica, "Smart home IoT traffic characteristics as a basis for DDoS traffic detection," in *Proceedings of the 3rd EAI International Conference on Management of Manufacturing Systems*, 2018.

[20] J. Mocnej, A. Pekar, W. K. G. Seah, and I. Zolotova, "Network traffic characteristics of the IoT application use cases," School of Engineering and Computer Science, Victoria University of Wellington, Tech. Rep. ECSTR18-0, 2018.

[21] T. H. Black, *Derivations of Applied Mathematics*. Debian Project, 2018. [Online]. Available: https://www.derivations.org

[22] A.K. Marnerides, A. Schaeffer-Filho, and A. Mauthe, "Traffic anomaly diagnosis in internet backbone networks: A survey," *Computer Networks*, vol. 73, pp. 224–243, 2014-11.

[23] S. S. Bhunia and M. Gurusamy, "Dynamic attack detection and mitigation in IoT using SDN," in *2017 27th International Telecommunication Networks and Applications Conference (ITNAC)*. IEEE, 2017, pp. 1–6.

[24] Y. Meidan *et al.*, "N-BaIoT: Network-based Detection of IoT Botnet Attacks Using Deep Autoencoders," *IEEE Pervasive Computing*, vol. 13, no. 9, p. 8, 2018.