Towards IoT-DDoS Prevention Using Edge Computing

Ketan Bhardwaj, Joaquin Chung Miranda, Ada Gavrilovska
Georgia Institute of Technology

Abstract

Application-level DDoS attacks mounted using compromised IoT devices are emerging as a critical problem. The application-level and seemingly legitimate nature of traffic in such attacks renders most existing solutions ineffective, and the sheer amount and distribution of the generated traffic make mitigation extremely costly. This paper proposes a new approach which leverages edge computing to deploy edge functions that gather information about incoming traffic and communicate that information via a fast-path with a nearby detection service. This accelerates the detection and the arrest of such attacks, limiting their damaging impact. Preliminary investigation shows promise for up to 10x faster detection that reduces up to 82% of the Internet traffic due to IoT-DDoS.

1 Introduction

The IoT-DDoS Problem. Safeguarding the web infrastructure and services against the highly damaging Distributed Denial-of-Service (DDoS) attacks is a difficult and pressing need today. Conventionally, DDoS campaigns are carried out by botnets which utilize an army of infected computers/devices to overwhelm a target web service or Internet infrastructure element with malicious traffic. Such attacks have been studied extensively in the past. Recently a new and more damaging kind of application-level DDoS attacks are emerging, where compromised Internet of Things (IoT) devices are used as attackers. Examples include the attacks on KrebsOnSecurity [8] and Dyn [7] by the Mirai botnet [5]. We refer to this type of attacks as IoT-DDoS. IoT-DDoS attacks pose a critical problem to be solved for broad adoption of IoT.

IoT-DDoS challenges current DDoS security measures in new ways. The recent application-level IoT-DDoS attacks evade existing attack detection solutions such as network intrusion detection systems (NIDS) because the data in the attack traffic appears to be, or is indeed originating from legitimate IP addresses of IoT devices. The sheer amount of the traffic generated in an IoT-DDoS [5] makes the cost associated with deploying mitigation solutions so high that only a few companies with massive infrastructure can afford to handle it, such as Google's Project Shield [9], Akamai [11] or CloudFlare [12], who offer on-demand DDoS protection as a service.

Can the Edge Help? In this paper, we explore an approach which leverages computational resources in the edge of the network to accelerate the defense from IoT-DDoS attacks, and arrest them before they can cause considerable damage to both the attack target as well the Internet infrastructure itself.

We are motivated by the following observations. First, detection of a DDoS is best done close to the victim, whereas its mitigation and prevention are most effective close to the attack source. Second, the emergence of a new infrastructure tier at the edge of the network, including mobile edge computing (MEC) access points [33], fog computing gateways [26], and similar elements, presents opportunities to dynamically provision edge functions [25, 48, 44] to handle the vast amounts of Internet traffic created by IoT devices.

We posit that this edge tier provides a new vantage point closer to the source, which has not been considered in other approaches to counter IoT-DDoS. However, the edge sees limited network traffic, and is limited in terms of compute resources. The edge can neither capture the aggregate network traffic required for IoT-DDoS detection, nor can it scale resources needed for mitigation like the elastic cloud . As a result, simply deploying existing approaches to absorb DDoS using infrastructure provisioning will not suffice at the edge.

We observe, however, that the edge of the network – the wireless and cellular gateways providing connectivity for IoT devices with the Internet – have capabilities needed for limited processing of IoT packets. These resources can be sufficient for capturing lightweight information profiles of the IoT packets that stream through them. Examples can be as trivial as packet counts, or reduced packet information containing header-only data, or more sophisticated algorithms for sketching streamed data. If these information profiles can be more rapidly aggregated, relative to what's possible when aggregating the original full-featured IoT traffic, *and* if the information they carry correlates sufficiently well with the information necessary to detect an attack, they create a path to accelerate detection of an impending IoT-DDoS and deployment of appropriate defense mechanisms at the edge infrastructure.

ShadowNet. Motivated by these observations , we propose *ShadowNet* – an architecture that makes the edge the first line of defense against IoT-DDoS. It achieves its goals in the following manner. First, appropriate *edge functions* are deployed on the distributed edge infrastructure, on behalf of a backend IoT application seeking protection. The role of these edge functions are to sketch the profiles of IoT traffic streaming from a given edge location. Second, the edge function establishes a *fast path* between itself and a special ShadowNet web service. The edge function uses the fast path to send small *shadow-packets* with locally-derived information about IoT traffic to the ShadowNet web service. ShadowNet can then aggregate that information about the IoT traffic distributed across a number of edge nodes, detect an imminent IoT-DDoS attack, and respond with some proactive defensive action. The approach assumes that an edge function and web service can mutually attest each other to establish trust between them.

In this paper, we present the ShadowNet idea, and argue that a possible outcome is accelerated detection and response to an attack, potentially even before it reaches the target. Concretely, we present encouraging preliminary results from experiments we carried out on an initial ShadowNet prototype. We demonstrate that ShadowNet can detect an impeding IoT-DDoS attack up to 10x faster than the best case detection at victim, and prevents injection of up to 82% of the traffic in the Internet infrastructure by compromised IoT devices. In that sense, ShadowNet represents a major contribution towards defenses against IoT-DDoS attacks.

2 Motivation

We present a brief discussion of the technology landscape that motivates the design of ShadowNet.

DDoS Attacks. DDoS attacks were observed first in the early 2000's when several web sites (e.g., Yahoo, eBay, Amazon, and ZDnet) were taken offline, incurring significant financial losses [41]. In December 2010, the group *Anonymous* targeted the web sites of Mastercard, PayPal, Visa, and PostFinance [49]. Most recently, in 2016, the Krebs On Security [8] and Dyn [7] websites were victims of massive DDoS attacks facilitated by IoT devices. Based on reports from industry leaders in DDoS protection services, such as Arbor Networks [22], Akamai [19], F5 [32], Incapsula [13], and Neustar [6], DDoS attacks in general are becoming bigger, more complex and more frequent. In summary, IoT-DDoS is a formidable problem due to its scale, complexity and frequency, which can prove to be a roadblock for IoT adoption.

DDoS Network Traffic Patterns. DDoS network traffic patterns mostly depend on the tools used to launch the attack. Bukac [27] studied the traffic characteristics of common DoS tools such as High Orbit Ion Cannon (HOIC) [14], HULK [15], Low Orbit Ion Cannon (LOIC) [16], OWASP HTTP tool [18], and Slowloris [21]. Based on this study, we can classify the attack buildup patterns of these tools in two categories: quick response and gradual buildup. A more recent and prevalent

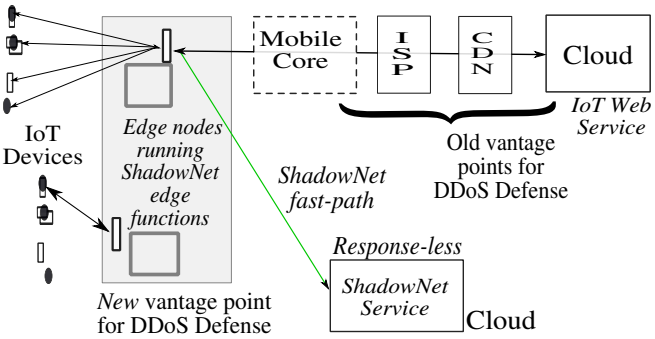


Figure 1: ShadowNet Overview

technique is hit-and-run DDoS attacks [5] in which the attacker sends multiple short bursts of quick response traffic followed by periods of inactivity, making it very challenging to manage.

DDoS Defenses. The objective of DDoS defenses is to detect the attack as soon as possible, and to mitigate it. DDoS defenses can be source-based, destination-based, network-based, or a hybrid. They can be deployed before the attack (prevention), during the attack (detection), and after the attack (source identification and response). In the many years of DDoS history, many detection and mitigation mechanisms have been proposed (source-end detection [45, 39], secure overlay networks [46, 28, 42], SDN/NFV based approaches [40, 43, 31, 30] and cloud based approaches [41, 49]). However, in all cases, Internet users and infrastructure still incur damage, as the Dyn attack of 2016 can testify [7].

IoT and Edge Computing. Edge computing—the use of computational resources closer to end devices, at the edge of network—is an attractive approach to addressing latency and bandwidth demands of emerging applications. Going beyond point solutions, the vision of edge computing, where web services deploy their *edge functions* in a multi-tenant infrastructure, is proposed by researchers [25, 44] and industry initiatives [1, 3]. Edge computing and IoT are deeply interrelated. First, the anticipated data deluge from tens of billions of connected devices, potentially related to critical infrastructure and services, is a major driver for edge computing – the only way to operate on this data with low latencies and low data movement costs is by moving computation closer to them, along the network edge. Second, the architecture of IoT deployments is a natural fit for edge computing, as, for energy-related reasons, IoT device connect through the Internet through low-power protocols via wireless or cellular gateways [37, 47, 29], examplifying edge nodes.

3 Overview

Figure 1 shows the components of the ShadowNet architecture. With ShadowNet, these edge nodes are securely provisioned to run application-specific *edge functions* that handle requests for corresponding web services. The edge functions use a *fast path*, faster than the original path to the application web service, to relay the traffic profile sketch to a ShadowNet service, using small, self-authenticated *shadow packets*. The *ShadowNet service* aggregates these faster sketches from all

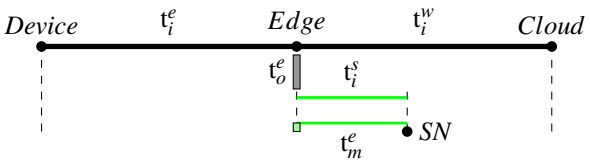


Figure 2: Opportunities afforded by ShadowNet.

edge nodes, to make more early observations regarding possible anomalies. A practical defense mechanism made possible with ShadowNet involves quickly deploying new edge functions which implement defensive actions such as blocking certain IPs at the edge nodes themselves, preventing the spread of the attack and eliminating attack-related load from the core network. As with the edge functions, we foresee that the applications using the ShadowNet service will deploy suitable anomaly detection, based on their best security practices.

Ideally, the ShadowNet service would be as accurate but faster in detecting an attack than any analysis done at the web service itself. However, to maintain lightweightness of edge functions, accuracy may be traded-off for performance by employing techniques such as sampling or predictive analytics. A similar argument is applicable for the counter measures or actions to the detected attacks.

Figure 2 provides an illustration of the opportunities that the ShadowNet architecture affords. In current systems, IoT-DDoS packets, along with regular IoT traffic, traverse the edge gateways and are propagated to a target application web service deployed in a backend cloud or datacenter, accessed via the wide area network, with the i-th packet taking time $t_i^e + t_i^w$. Since ShadowNet can be reached via the fast path t_i^s , information about the IoT traffic is aggregated more quickly, and, on a potential attack, appropriate actions can be triggered in a more timely manner, i.e., in time t_m^e . This means that IoT DDoS is detected faster if the following condition holds true for all packets in an attack: $t_o^e + t_i^s < t_i^w$; the damage prevented is function of t_m^e, i.e., the time it takes to start a mitigating action. Examples of actions can range from triggering an immediate prevention mechanism, such as blocking the attack traffic, to raising probabilistic alarms requiring additional investigation. The later may be needed when the shadow packet information is lossy with respect to the full packet signature, and cannot detect it with the same fidelity as when using the original traffic stream. ShadowNet builds on our previous work on fast and efficient *just-in-time* deployment of edge functions the edge [25]. This keeps t_m^e minimal, which is important to prevent substantial damage from an attack. It also provides for secure interactions among the ShadowNet components, and allows for operating on data over encrypted connections without violating end-to-end security guarantees [24].

4 Feasibility and Challenges

In this section, we discuss the feasibility and the challenges in addressing the requirements of the ShadowNet system.

Designing a ShadowNet fast-path. First, the effectiveness of ShadowNet depends on the ability to quickly deliver insights regarding the traffic being generated at the edge to the ShadowNet backend service. Thus, being able to create the ShadowNet fast-path is a critical element of the solution.

There are two main classes of mechanisms that can be used practically to create the ShadowNet fast-path. The first requires *hardware-assistance* for network slicing to prototype the ShadowNet fast-path using a dedicated network slice [36, 2, 4], proposed to be a key technology in 5G networks. The second set of mechanisms are purely *software-based*. These include the use of pre-population of dedicated routes to the ShadowNet service, thus reducing the network distance between the edge and the ShadowNet service, using priority packet scheduling for shadow-packets vs. the other traffic at the edge infrastructure, or reducing the protocol distance in the system software stack by using different network layers for implementing the fast-path and reducing the weight of the packets to transfer, i.e., sending a small packet containing a small keyword with self authentication vs. forwarding the full request to the ShadowNet service. In this paper, we present our preliminary results by implementing the last two.

Designing a trusted ShadowNet service. After solving the fast-path challenge, the next question is how to ensure that only the right packets are accounted for in detecting an imminent IoT-DDoS. We leverage the shielded execution environment available to edge functions to send its unforgeable attestation to the ShadowNet service, which can later be used to identify packets [23, 25]. To avoid replay attacks, this attestation is sent over an encrypted channel with a pre-shared key between the edge functions and the ShadowNet service. The pre-shared key can be rotated by another entity such as the web service using ShadowNet, or the edge infrastructure owner. Additional challenges come from determining how to reasonably scale the ShadowNet service. For this paper, we designed a ShadowNet service as a web service and we use a DTLS over UDP protocol to realize a no-response service.

Designing customizable prevention. ShadowNet depends on edge functions that can be quickly deployed at the affected edge locations (either by the ShadowNet service or independently) to gracefully degrade the quality of service in response to a potential DDoS. With enough edge deployments, the ShadowNet service could potentially provide the IoT web service a peek into the traffic that is going to hit it giving it an option of either prepare to handle it, or to stop it before it hits the public Internet infrastructure.

The performance of ShadowNet in preventing IoT-DDoS attacks will rely on its ability to deploy mitigation edge functions in time. Recent attacks [5] happen in burst of 5 minutes, so if it takes more time to deploy mitigation edge functions, it would not be effective. Existing research [25], showed that containers (vs. virtual machines and applications sandboxes) can be used to deploy edge functions with acceptable delays. Further, by pre-deploying images of mitigation edge functions, this can be further shortened to order of seconds after detection happens. Concerning how an edge function

prevents the traffic at the edge or the strategies to mitigate an IoT-DDoS, we simply start selectively dropping packets at all edge functions. Exploring more sophisticated approaches such as selecting specialized edge functions or particular APIs to block to minimize impact is part of the open challenges.

Proof of concept implementation. We prototyped the ShadowNet edge function and the ShadowNet service in the Go programming language. We implemented the ShadowNet edge functions as UDP and HTTP reverse proxies, and the ShadowNet service as a corresponding server listing for shadow packets. The edge functions' fast path to the ShadowNet web service is based on transmission of a short packet containing the keyword "shadow" over a low-latency slice of our testbed network. To detect an IoT-DDoS attack at the ShadowNet service, we record the arrival time of each shadow packet with a timestamp, and compare it with the arrival time of the previous shadow packet. The difference in the arrival time of two consecutive packets is the inter-packet spacing, which is directly correlated to the HTTP request rate, or the UDP transmission rate received at the ShadowNet edge function. If the inter-packet spacing drops below certain threshold, i.e., the HTTP request rate, or the UDP transmission rate at the ShadowNet edge function had surpassed a certain threshold, we raise an alert.

5 Preliminary Experimental Results

Experimental setup. We created the testbed using the GENI platform, an open infrastructure for at-scale networking research in the US. It allows researchers to request virtual machines and software-defined networking (SDN) [35] switches for their own experiments. For our experiments, we requested four virtual machines (VM): attacker, ShadowNet edge function, ShadowNet service, and victim. All VM's are hosted at our institute's GENI location, and interconnected by Fast Ethernet (100 Mbps), as shown in Figure 3(a). Each VM has one Intel(R) Xeon(R) CPU X5650 @ 2.67GHz core, 1 GB of RAM running Ubuntu 14.04. To emulate the fast-path in ShadowNet, we used the *tc* and *netem* Linux utilities to add delay RTT to the server NICs. The RTTs between different components are as shown in Figure 3(a) with assumption of symmetrical links.

Attack characteristics. For our experiments, we considered attack characteristics of IoT botnets such as HTTP GET flooding generated by all type of sensors and UDP flooding generated by video surveillance cameras [38]. Furthermore, these attacks may be launched for extended periods of time or using hit-and-run tactics (i.e., short burst of 5 minutes) as in the Mirai campaigns against KrebsOnSecurity and Dyn [5]. We generated the attacks using BoNeSi, a DDoS botnet simulator capable of generating UDP and HTTP GET flooding attacks [10]. For the HTTP GET flooding attack, we used BoNeSi to generate HTTP GET requests at a rate of 500 req/s. We programmed a web server in the Go language to generate a response of 260 KB for every request. This value is the average size of an HTTP web page without including videos, taken from `httparchive.org`. The HTTP GET flooding attack forces the web server to generate a maximum of 1 Gbps response traffic. We measured that this attack is able to take down our web server in less than one minute. Similarly, the UDP flooding attack generates 1000 bytes UDP packets at a rate of 8500 packets per second (pps). This attack puts 68 Mbps of traffic in the network almost instantly. For both types of attack, BoNeSi generates requests using 252 attacker IP addresses from the class C network connected to the attacker VM. We lost 2 IP address that are assigned to the link between the attacker VM and the edge function VM.

Measurement setup. Our measurement setup is composed of Open-vSwitch (OVS) [17] switches between the ShadowNet edge function and the ShadowNet service and victim VMs (see Figure 3(a)). These switches run sFlow [20] to collect network traffic statistics, and report them to a central collector using HTTP. To ensure that the measurements are accurate we used a common clock synced using sFlow collectors at all components and ran all detection techniques at the same time. Also, we configured the polling interval of the switches to 1 second. This configuration allows us to achieve fastest detection with sFlow. To show *how fast a ShadowNet service detects an IoT-DDoS attack compared to a traditional sFlow-based approach*, we compare the detection time for the HTTP GET flood and UDP flood attacks for the following:

1. ShadowNet using UDP (SN): an edge function sending shadow-packets to ShadowNet service and thresholding inter-packet spacing to detect an imminent DDoS.

2. ShadowNet using sFlow (ESF): an edge function sending shadow packets to ShadowNet service using an sFlow-enabled OVS and an sFlow collector while thresholding incoming packet rate to detect an imminent DDoS.

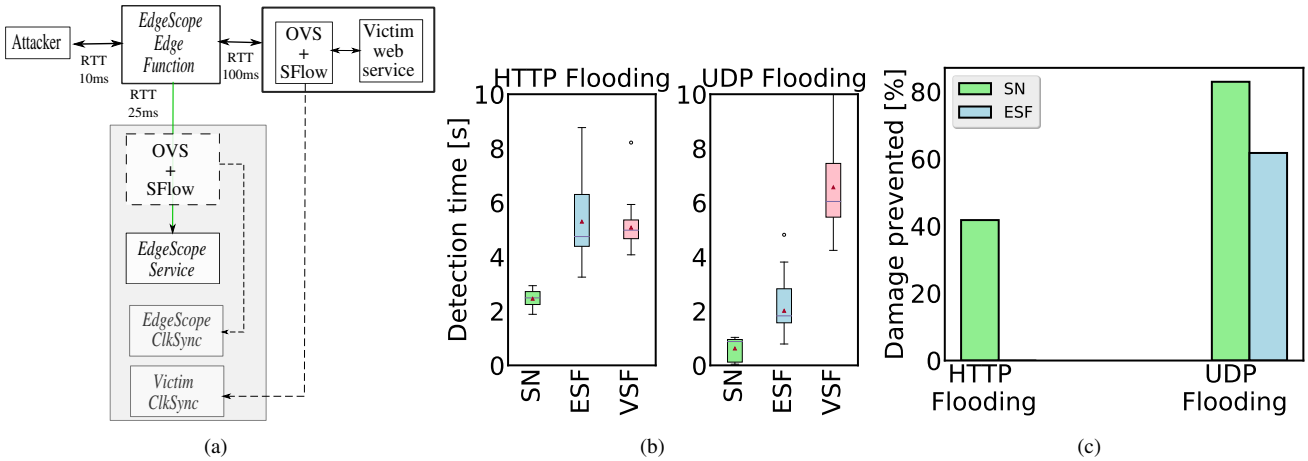
3. Detection at victim using sFlow (VSF): an sFlow-enabled OVS at the victim server and thresholding incoming packet rate to detect an ongoing DDoS.

Results. In our preliminary experiments, we seek answers to the following questions:

Q. How fast can ShadowNet prevent damage by IoT-DDoS?

Figure 3(b) shows detection times for ShadowNet and sFlow under HTTP GET and UDP flooding attacks, measured in seconds needed for any of the systems to raise an alert after the attack has started. For an HTTP GET flooding attack, the ShadowNet service detects an attack in 2.46 seconds by monitoring inter-packet spacing of shadow-packets, while by using sFlow detection, the ShadowNet service detects the attack in 5.30 seconds, and the victim in 5.08 seconds. Similarly, the ShadowNet service detects an UDP flooding attack in 0.62 seconds, while sFlow detects the attack in 2.01 seconds at the ShadowNet service, and 6.57 seconds at the victim.

These results demonstrate that a fast-path combined with a response-less service can be 2.1x and 10.6x faster than traditional detection methods for HTTP and UDP flooding respectively. Despite that ShadowNet server is counting UDP packets (Layer 4) and the sFlow agent is counting Ethernet



(a) (b) (c)

Figure 3: Showing (a) our experimental setup, (b) DDoS early detection, and (c) DDoS damage prevented by ShadowNet.

frames (Layer 2) at the switch, the ShadowNet service provides faster detection because it does not rely on connection-based protocols like HTTP to report measurements, which is the case for the detection application using sFlow. It is important to note that the dynamics of the HTTP GET flooding attack generate a moderate request rate to pass undetected by inbound traffic meters. However, it is the size of the response payload multiplied by the amount of concurrent connections in a short period of time that produces the DDoS. The fact that ShadowNet is able to detect the attack 2.06 times faster than the victim, while sFlow at ShadowNet detects the attack just 1.04 times slower that the victim, showcases the benefits of our approach. Moreover, Figure 3(b) shows that sFlow at ShadowNet has a high standard deviation and a maximum detection time around 9 seconds, making this detection technique unsuitable for connection-oriented IoT-DDoS attacks.

Q. How much damage can be prevented by using ShadowNet?

Considering the attack characteristics and the detection time measurements, Figure 3(c) presents how much damage is prevented by using ShadowNet in terms of percentage of traffic that did not enter the network. We compare ShadowNet detection time against our measurements at the victim, although industry standard mean-time-to-mitigation (MTTM) is 5 minutes. We assume a mitigation system that provisions network policies with a configuration delay 0.5 seconds [25]. Figure 3(c) shows that ShadowNet using UDP shadow packets prevents from 40% to 82% of the damage for HTTP and UDP flooding, respectively. ShadowNet sFlow prevents around 60% of the damage for a UDP flooding attack, while it is not able to prevent any damage for an HTTP flooding attack. In the best case scenario, ShadowNet detects a UDP flooding attack in 0.62 seconds, while injecting shadow packets of size 70 bytes at a rate of 8500 pps on the fast path. This is equivalent to 4.76 Mbps for 0.62 seconds. Similarly, for the HTTP flooding attack detected in 2.46 seconds generating 280 Kbps.

6 Discussion

The preliminary results are encouraging but they leave out a number of important open questions. In order to be effective, ShadowNet would need to be applied across multiple networks and creating a fast path across them presents additional challenges. Further, it may not be sufficient for the ShadowNet service to be in a single location. Deployment in multiple locations can impact the fast-path assumptions and too much geographical replication would affect aggregation performed at the ShadowNet service. Given that most recent attacks are orchestrated from geographically distributed locations, there are important research questions around replication models for ShadowNet. Concerning ShadowNet edge functions, there are open questions on *what and how much* should be incorporated in them? For example, in our prototype shadow-packets are generated *per request* for each service ShadowNet is protecting. This may pose high overhead at the edge for ShadowNet to be practical. An appropriate sampling approach can reduce that overhead. However, the trade-offs in using sampling and accuracy of detection need to be carefully considered. Further, to distinguish between flash-crowds and a DDoS, ShadowNet can use known techniques [34] that can be incorporated in edge-functions.

7 Summary

We propose ShadowNet as an approach to prevent application-level IoT-DDoS attacks by leveraging the emerging edge infrastructure. It not only protects the web services by detecting IoT-DDoS 10 times faster than existing approaches but also prevents 82% traffic to enter the Internet infrastructure, reducing the damage. We presented encouraging preliminary evaluation with a prototype implementation. Future work will complete the ShadowNet implementation and evaluations to study the trade-offs under realistic conditions.

Acknowledgements

We thank the anonymous reviewers and shepherd, Rolf Schuster, for their insightful comments. This work was supported with funding from the Cisco University Research Program, the Georgia Tech Center for Development and Applications of IoT (CDAIT), and VMware's University Research Fund.

References

[1] Etsi mobile edge computing. http://goo.gl/Qef61X.

[2] Network sliceing for 5g networks: 5g americas. http://www.5gamericas.org/files1414/8052/9095/5G_Americas_Network_Slicing_11.21_Final.pdf.

[3] Open edge computing. http://openedgecomputing.org/about-oec.html.

[4] Towards 5g network slicing - motivations and challenges. http://5g.ieee.org/tech-focus/march-2017/towards-5g-network-slicing.

[5] Breaking down mirai: An iot DDoS botnet analysis. https://www.incapsula.com/blog/malware-analysis-mirai-ddos-botnet.html, 10 2016.

[6] DDoS & cyber security insights. https://hello.neustar.biz/2016-soc-report-security-lp.html, 12 2016.

[7] Dyn analysis summary of friday october 21 attack. http://dyn.com/blog/dyn-analysis-summary-of-friday-october-21-attack/, 11 2016.

[8] KrebsOnSecurity hit with record DDoS. https://krebsonsecurity.com/2016/09/krebsonsecurity-hit-with-record-ddos/, 11 2016.

[9] Project shield. https://projectshield.withgoogle.com/public/, 7 2016.

[10] BoNeSi. https://github.com/Markus-Go/bonesi, 7 2017.

[11] DDoS mitigation. https://www.akamai.com/us/en/resources/ddos-mitigation.jsp, 4 2017.

[12] DDoS protection. https://www.cloudflare.com/ddos/, 4 2017.

[13] Global DDoS threat landscape q4 2016. , 3 2017.

[14] Hoic - high orbit ion cannon. https://www.incapsula.com/ddos/attack-glossary/high-orbit-ion-cannon.html, 4 2017.

[15] Hulk, web server DoS tool. http://www.sectorix.com/2012/05/17/hulk-web-server-dos-tool/, 4 2017.

[16] Loic - low orbit ion cannon. https://www.incapsula.com/ddos/attack-glossary/low-orbit-ion-cannon.html, 4 2017.

[17] Open vSwitch. http://openvswitch.org/, 2017.

[18] Owasp http post tool. https://www.owasp.org/index.php/OWASP_HTTP_Post_Tool, 4 2017.

[19] Q4 2016 state of the internet / security report. https://www.akamai.com/us/en/about/our-thinking/state-of-the-internet-report/global-state-of-the-internet-security-ddos-attack-reports.jsp, 2 2017.

[20] sFlow - making the network visible. http://www.sflow.org/, 2017.

[21] Slowloris. https://www.incapsula.com/ddos/attack-glossary/slowloris.html, 4 2017.

[22] Worldwide infrastructure security report. https://www.arbornetworks.com/images/documents/WISR2016_EN_Web.pdf, 1 2017.

[23] ARNAUTOV, S., TRACH, B., GREGOR, F., ET AL. SCONE: Secure Linux Containers with Intel SGX. In *Proc. of Symposium on Operating Systems Design and Implementation (OSDI'16)* (Savannah, GA, 2016).

[24] BHARDWAJ, K. *Frames, Rods and Beads of the Edge Computing ABACUS.* PhD thesis, 2016.

[25] BHARDWAJ, K., SHIH, M., AGARWAL, P., GAVRILOVSKA, A., KIM, T., AND SCHWAN, K. Fast, scalable and secure onloading of edge functions using airbox. In *Proceedings of first IEEE/ACM symposium on Edge Computing*.

[26] BONOMI, F., MILITO, R., ZHU, J., AND ADDEPALLI, S. Fog computing and its role in the internet of things. In *Proceedings of the first edition of the MCC workshop on Mobile cloud computing*.

[27] BUKAC, V. Traffic characteristics of common dos tools. Tech. rep., Masaryk University, Technical report FIMU-RS-2014-02, 2014.

[28] CHOWRIWAR, S. S., MOOL, M. S., SABALE, P. P., PARPELLI, S. S., AND SAMBHE, N. Mitigating denial-of-service attacks using secure service overlay model. *International Journal of Engineering Trends and Technology (IJETT) 8*, 9 (2014), 37.

[29] Cisco IoT Networking. https://www.cisco.com/c/m/en_us/solutions/internet-of-things/iot-system.html.

[30] FAYAZ, S. K., TOBIOKA, Y., SEKAR, V., AND BAILEY, M. Bohatei: Flexible and elastic ddos defense. In *Usenix Security* (2015).

[31] GIOTIS, K., ARGYROPOULOS, C., ANDROULIDAKIS, G., KALOGERAS, D., AND MAGLARIS, V. Combining openflow and sflow for an effective and scalable anomaly detection and mitigation mechanism on sdn environments. *Computer Networks*.

[32] HOLMES, D. 2016 DDoS attack trends. https://f5.com/Portals/1/PDF/security/2016_DDoS_Attack-Trends.pdf, 11 2016.

[33] HU, Y. C., PATEL, M., SABELLA, D., SPRECHER, N., AND YOUNG, V. Mobile edge computingâĂŤa key technology towards 5g. *ETSI White Paper 11* (2015).

[34] JUNG, J., KRISHNAMURTHY, B., AND RABINOVICH, M. Flash crowds and denial of service attacks: Characterization and implications for cdns and web sites. In *Proceedings of the 11th International Conference on World Wide Web*, WWW '02.

[35] KREUTZ, D., RAMOS, F. M. V., VERÃ■SSIMO, P. E., ROTHENBERG, C. E., AZODOLMOLKY, S., AND UHLIG, S. Software-defined networking: A comprehensive survey. *Proceedings of the IEEE 103*, 1 (Jan 2015), 14–76.

[36] LI, Q., WU, G., PAPATHANASSIOU, A., AND UDAYAN, M. An end-to-end network slicing framework for 5g wireless communication systems. *CoRR abs/1608.00572* (2016).

[37] LoRa Alliance. https://www.lora-alliance.org/technology.

[38] MICRO, T. Persirai: New internet of things (IoT) botnet targets ip cameras. http://blog.trendmicro.com/trendlabs-security-intelligence/persirai-new-internet-things-iot-botnet-targets-ip-cameras/, 5 2017.

[39] MIRKOVIC, J., AND REIHER, P. D-WARD: a source-end defense against flooding denial-of-service attacks. *IEEE Transactions on Dependable and Secure Computing 2*, 3 (July 2005), 216–232.

[40] PASSITO, A., MOTA, E., BENNESBY, R., AND FONSECA, P. Agnos: A framework for autonomous control of software-defined networks. In *2014 IEEE 28th International Conference on Advanced Information Networking and Applications* (May 2014), pp. 405–412.

[41] PENG, T., LECKIE, C., AND RAMAMOHANARAO, K. Survey of network-based defense mechanisms countering the dos and ddos problems. *ACM Comput. Surv. 39*, 1 (Apr. 2007).

[42] ROBINSON, M., MIRKOVIC, J., MICHEL, S., SCHNAIDER, M., AND REIHER, P. Defcom: defensive cooperative overlay mesh. In *Proceedings DARPA Information Survivability Conference and Exposition* (April 2003), vol. 2, pp. 101–102 vol.2.

[43] SAHAY, R., BLANC, G., ZHANG, Z., AND DEBAR, H. Towards autonomic ddos mitigation using software defined networking. In *SENT 2015: NDSS Workshop on Security of Emerging Networking Technologies* (2015), Internet society.

[44] SATYANARAYANAN, M. A brief history of cloud offload: A personal journey from odyssey through cyber foraging to cloudlets. *GetMobile: Mobile Comp. and Comm. 18*, 4 (Jan. 2015), 19–23.

[45] SEKAR, V., DUFFIELD, N. G., SPATSCHECK, O., VAN DER MERWE, J. E., AND ZHANG, H. Lads: Large-scale automated DDoS detection system. In *USENIX Annual Technical Conference, General Track* (2006), pp. 171–184.

[46] SITARAMAN, R. K., KASBEKAR, M., LICHTENSTEIN, W., AND JAIN, M. Overlay networks: An akamai perspective. *Advanced Content Delivery, Streaming, and Cloud Services 51*, 4 (2014), 305–328.

[47] WANG, Y.-P. E., LIN, X., ADHIKARY, A., GROÌĹVLEN, A., SUI, Y., BLANKENSHIP, Y., BERGMAN, J., , AND RAZAGHI, H. S. A Primer on 3GPP Narrowband Internet of Things (NB-IoT). In *arxiv.org* (2016).

[48] WILLIS, D. F., DASGUPTA, A., AND BANERJEE, S. Paradrop: A multi-tenant platform for dynamically installed third party services on home gateways. In *Proceedings of the 2014 ACM SIGCOMM Workshop on Distributed Cloud Computing* (New York, NY, USA, 2014), DCC '14, ACM, pp. 43–44.

[49] ZARGAR, S. T., JOSHI, J., AND TIPPER, D. A survey of defense mechanisms against distributed denial of service (ddos) flooding attacks. *IEEE Communications Surveys Tutorials 15*, 4 (Fourth 2013), 2046–2069.