Towards IoT-DDoS prevention using edge computing

Paper # 69, 6 pages

Abstract

Application-level DDoS attack 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 to prevent those attacks by leveraging edge computing infrastructure before the cause damaging. Our preliminary investigation shows promise for up to 10x faster detection that reduces upto 82% Internet traffic due to IoT-DDoS.

1. INTRODUCTION

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 attack are emerging, based on compromised Internet of Thing (IoT) devices. Examples include the attacks on KrebsOnSecurity [8] or Dyn [7] by the Mirai botnet. 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 do it, such as Google's Project Shield [9].

In this paper, we present a new approach, from mitigation to prevention of IoT-DDoS before it can cause damage. We firmly believe that *IoT-DDoS attack prevention* will be the only feasible approach that has to be adopted widely in future. The rationale behind taking a preventive position is derived from two observations. First, it is well accepted that detection of a DDoS is best done close to the victim and its mitigation is most effective close to the source. Second, the emergence of the new infrastructure tier at the edge of the network (e.g., MEC [30], Fog [23], etc.), presents opportunities to dynamically provision edge functions [22] to handle vast amounts of data created by IoT devices. We posit that the same edge tier also provides an optimal 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. As a result, simply deploying existing approaches at the edge infrastructure will not suffice. The edge can neither capture the aggregation of network traffic required for IoT-DDoS detection nor can it scale resources like the elastic cloud needed for mitigation.

A system designed to address those limitations, making the edge the first line of defense against IoT-DDoS, would achieve this in the following manner. First, appropriate *edge functions* are deployed on the distributed edge infrastructure to handle IoT traffic on behalf of backend IoT web services. Then, the edge function sends a very small *shadow-packet* for every application level request it sees to a another custom web service over a *fast path* established between itself and the *web service*. Finally, the web service can then aggregate that information from a number of edge nodes to detect imminent IoT-DDoS attack, and respond with some proactive preventive action. The approach assumes that an edge function and web service can mutually attest each other to establish trust.

Owing to the analogy of the approach to a three body physical system (a light source, an object and a surface) where an oblique light source shining on a moving object casts a longer shadow that reaches a point on the surface before the object itself, we call it ShadowNet. The goal of this paper is to explore the feasibility of the approach and outline different research directions that can make possible prevention of IoT-DDoS attacks by foreshadowing their imminent arrival.

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 upto 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 on the technology landscape that motivates the development 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 [38]. In December 2010, the group *Anonymous* targeted the web sites of Mastercard, PayPal, Visa, and PostFinance [44]. 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 [20], Akamai [17], F5 [29], Incapsula [11], 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 their scale, complexity and frequency which can prove to be a road-block for IoT adoption.

DDoS Network Traffic Patterns: DDoS network traffic patterns mostly depend on the tools used to launch the attack. Bukac [24] studied the traffic characteristics of common DoS tools such as High Orbit Ion Cannon (HOIC) [12], HULK [13], Low Orbit Ion Cannon (LOIC) [14], OWASP HTTP tool [16], and Slowloris [19]. Based on this study, we can classify the attack buildup patterns of these tools in two categories: quick response and gradual buildup. More recent and prevalent technique is hit-and-run DDoS attacks [5] in which the attacker send 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 as close to the sources as possible. 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). However, in all cases Internet users and infrastructure still incur damage, as the Dyn attack of 2016 can testify [7].

Emergence of 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 [22, 41] and industry initiatives [1, 3]. We posit that ShadowNet can be the killer application of such edge computing infrastructure.

3. OVERVIEW

Brief. Figure 1 shows the components of the ShadowNet system. IoT web services deploy their lightweight *edge functions* at the edge to handle/process data generated by IoT devices on their behalf. ShadowNet creates a *fast-path* between the edge infrastructure and the ShadowNet web service. The ShadowNet service is a *response-less* service which foreshadows an imminent IoT-DDoS attack build up, by relying on self authenticated fixed and small size *shadow packets* received over the fast path. ShadowNet includes system support from the edge software stack for the edge functions to create secure *one-sided authentication* in the shadow-packets using shielded execution environments such as Intel SGX. This allows the ShadowNet service to uniquely identify shadow-packets from a particular edge function.

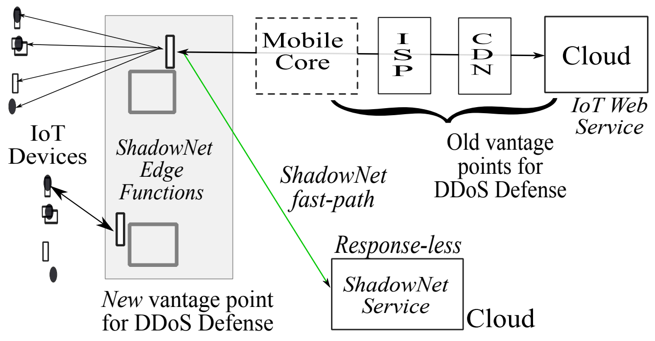


Figure 1: ShadowNet Overview

ShadowNet builds on previous research around fast and efficient *just-in-time* deployment of edge functions at the edge, to interact with ShadowNet service, and to counter the attack before it reaches the targeted web service or hits the Internet infrastructure, and on mechanisms to handle data over encrypted connections without violating end-to-end security guarantees.[1] In a nutshell, the threat model assumes that for ShadowNet edge functions and the ShadowNet service can be trusted trusted. Fortunally, technologies such as Intel SGX make these reasonable assumptions as shown in existing work [21, 22]. Next, we formally present our hypothesis:

Model and Hypothesis: We start by presenting a simplified model of a DDoS attack using the parameters listed in Table 1 and depicted in Figure 2. Our intentions with presenting a formal model are twofold: first, to present the observations and rationale behind ShadowNet, and second, to derive concrete design requirements (performance and functional) for ShadowNet to be effective as IoT-DDoS prevention system. Instead of using probabilities of achieved latencies, we use this simplistic model, for sake of clarity and without any loss of generality. Using the model, the time taken to detect a DDoS (t_{DDoS}) after it has been mounted can be stated as:

$$t_{DDoS} = f(\Sigma(t_i^e + t_i^w).w_i^w) \geq T^w$$
where for any i, j: $(t_i^e + t_i^w)$ - $(t_j^e + t_j^w) \leq \delta$

Similarly, the time a DDoS can be detected by ShadowNet (t_{DDoS}^s) can be represented as:

$$t_{DDoS}^s = f(\Sigma(t_i^e + t_i^s).w_i^s) \geq T^s$$
where for any i, j: $(t_i^e + t_i^w)$ - $(t_j^e + t_j^w) \leq \delta$

In above expressions, Σ represents the aggregation of requests that is needed for a DDoS detection and f represents the technique (e.g., counting or calculating gradient) used to

[1]references elided for blind review

detect DDoS. As earlier mentioned, ShadowNet is independent of f. In practice, there will be another term in the time that can be detected by ShadowNet representing the overhead of processing needed at the edge function, lets call it t^e_o. t^e_o represents overheads associated with inspecting incoming requests, generating appropriate shadow-packets and or a difference in processing capability at edge vs. backend web servers. For ShadowNet to deliver on early detection, the following condition must hold:

$$f(\Sigma(t^e_i w^w_i + t^s_i . w^s_i) + t^e_o \leq f(\Sigma(t^e_i + t^w_i).w^w_i)$$
For any i-th request

We posit that $t^e_o \lll (t^e_i w^w_i + t^s_i . w^s_i)$ because of two factors. First, the edge infrastructure is inherently designed to process requests in edge functions so it will be appropriately provisioned with enough compute capabilities so as to minimally impact transfer times while operating on them.

Second, the size of shadow-packets will be minimal packet size, i.e., $w^s_i \lll w^w_i$. Considering these factors, we can safely neglect t^e_i term. This brings us to the first ShadowNet hypothesis:

H1: *A practical DDoS preventions system can be made possible to detect an imminent IoT-DDoS if:*

$$f(\Sigma t^s_i . w^s_i) \leq T^s$$
with appropriately chosen T^s for edge.

In simple terms, H1 states that shadow-packets reach the ShadowNet service faster than actual request packets reach the web service, or that it is possible to create fast path for quicker detection. Practically, the ShadowNet will remain effective utill the following condition holds:

$$f(\Sigma t^s_i . w^s_i) \leq f(\Sigma t^w_i . w^w_i)$$

Next, to prevent any IoT-DDoS ShadowNet must be able to quickly and timely deploy mitigating edge function on the edge infrastructure before the attack builds up. This brings us to our second hypothesis:

H2: *If H1 is satisfied, mitigation edge functions can be deployed just-in-time to prevent a large amount of traffic to hit the Internet infrastructure.*

$$f(\Sigma(t^e_i . w^w_i + t^s_i . w^s_i) \ggg t^e_m$$

In simple terms, H2 states that with H1 we can detect DDoS quickly but we can only prevent traffic to cross the edge if the time to deploy a mitigating edge function is less than the duration of the attack bursts.

H3: *The conditions for ShadowNet effectiveness are aligned with IoT-DDoS attacks characteristics.*

In simple terms, H3 states that ShadowNet will be most effective for IoT-DDoS attacks. For example, in an IoT-DDoS attack mounted using wireless surveillance cameras connected to the Internet via a mobile network, a single request could span one of more video frame(s). Intuitively, we can see that $w^s_i / w^w_i \lll 1$ as a shadow-packet would be in bytes vs. video frame will be orders of magnitude larger, depending on video encoding, changes in scene, etc.

Param	Description
δ	Time b/w two consecutive requests at the victim
w^w_i	Weight or size of i-th request
w^s_i	Weight or size of shadow-packet for i-th request
f	Function to detect a DDoS using network traces
Σ	Aggregation of requests needed to detect a DDoS
T^w	Threshold used if detecting DDoS at the victim
T^s	Threshold if detecting DDoS at ShadowNet

Table 1: Parameters in DDoS model & ShadowNet hypothesis.

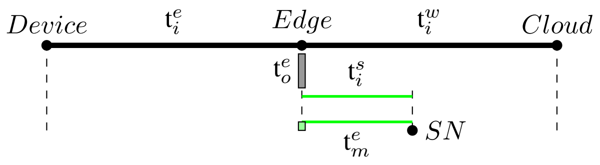


Figure 2: Time parameters in DDoS model & ShadowNet hypothesis.

Requirements: Based on the model above, ShadowNet's effectiveness will depend on the following:

D1: The ratio of the the shadow-packet size to the total request size, i.e., w^s_i / w^w_i, and the ratio of the transfer time between the edge and the ShadowNet service vs. the application web service, t^s_i / t^w_i, i.e., the fast-path needs to be faster.

D2: The system's ability to quickly deploy lightweight detection and mitigation edge functions on the edge infrastructure ensuring a constant $t^e_o \lll (t^e_i . w^w_i + t^s_i . w^s_i)$ and minimize t^e_m

Beyond the performance-related requirements ShadowNet also depends on the following functional requirements:

D3: A functional requirement is being able to design a response-less ShadowNet service.

D4: Security concerns for IoT data in motion and shadow-packets generated at the edge functions.

4. FEASIBILITY AND CHALLENGES

In this section, we discuss the feasibility, the design challenges and possible approaches to address the design and functional requirements of ShadowNet system.

Designing ShadowNet fast-path (D1). 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 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 ShadowNet fast-path using a dedicated network slice [33, 2, 4] as 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, reducing the protocol distance in the system software stack by using different network layers of implementing 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 Response-less ShadowNet service (D2). Second, the effectiveness of ShadowNet will depend on response-less design of the ShadowNet service. The ShadowNet service needs to be designed as a sink with no response whatsoever for incoming requests. With no request from devices triggering a response at ShadowNet service, it is theoretically not directly susceptible to a denial of service. Its main functionality is to apply appropriate function on shadow-packets to detect a DDoS attack at the granularity of a whole web service domain, or of a specific API.

Designing a response-less service seems trivial, however, there are non-intuitive design choices that lead to the technical challenges. First, how to create a web service that only receives shadow-packets without responding even for receipt of the shadow-packets? This can impact what mechanisms chosen for fast-path as well. For instance, if we chose TLS over TCP based interaction of ShadowNet edge function and ShadowNet service, then how do we defend this service against SYN/ACK flooding attacks for it to be able to effectively operate. One approach is to address this is by creating a very narrow interface that is only accessible to appropriate edge-functions. Other approach is to utilize a special purpose protocol between edge functions and ShadowNet services that aids in achieving those goals. Second, how to ensure that only the the right packets are accounted for in detecting an imminent IoT-DDoS? We leverage shielded execution environment available to edge functions to send its unforgeable attestation to ShadowNet service, which can later be used to identify packets. 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 is rotated by another entity such as the web service using ShadowNet, or the edge infrastructure owner. Third, how do we reasonably scale ShadowNet service? For this paper, we designed a ShadowNet service as a web service and we use a UDP protocol to realize a no-response service.

Designing customizable prevention. (D3) 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.

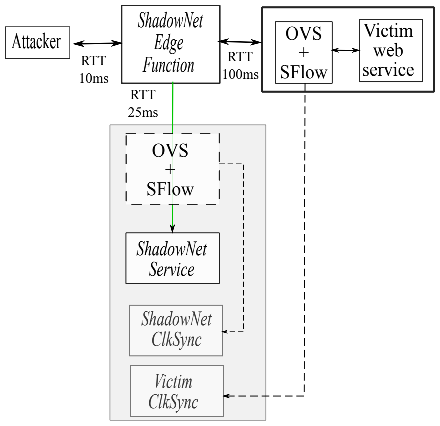
ShadowNet performance to prevent IoT-DDoS attack will rely on its ability to deploy mitigation edge functions in time. Looking at recent attacks [5], it shows that they happen in burst of 5 minutes, so if it takes more time to deploy mitigation edge functions, it would not be effective. Existing research [22], showed that containers (vs. virtual machines and applications sandboxes) can be used to deploy edge functions within an acceptable delays. Further, by pre-deploying images of mitigation edge functions, this can be further shortened to order of seconds after detection happen. 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 particular edge functions or selecting particular APIs to stop servicing to minimize its impact is part of future work in this project.

In addition to the above edge functions must be secured and capable of processing encrypted traffic without compromising its security properties to address D4. ShadowNet leverages existing technology on secure deployment of containers and secure processing of encrypted traffic at the edge using shielded execution environment (Intel SGX).

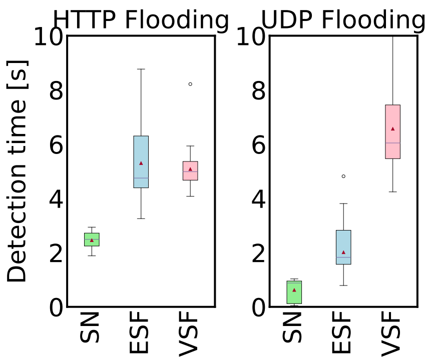
Proof of concept implementation: We prototyped the ShadowNet edge function and the ShadowNet service in the Go programming language. We implemented the ShadowNet edge function as UDP and HTTP reverse proxies, and the ShadowNet service as a corresponding server listed on shadow packets. The edge functions' fast path to the ShadowNet web service is based on transmission of a short packet containing the keyword "shadow". 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 surpassed certain threshold, we raise an alert.

5. PRELIMINARY RESULTS

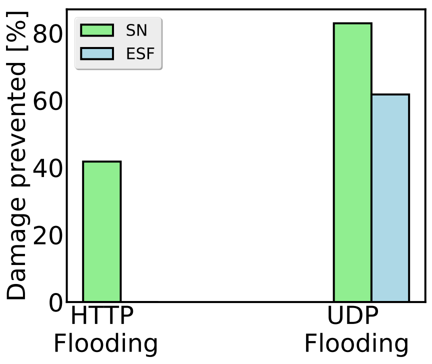
Experimental setup: We created the testbed using the GENI platform, an open infrastructure for at-scale networking research in the US. It allows researchers to request virtual machines and software-defined networking (SDN) [32] switches for their own experiments. For our experiments, we requested four virtual machines (VM): attacker, ShadowNet edge function, ShadowNet service, and victim. All VM's are hosted at the 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). We assumed symmetrical links, so we programmed half of the RTT on each interface of the link e.g., for the attacker-ShadowNet edge function link, we added 5 ms delay using *tc* and *netem* on the attacker interface, and another 5 ms on the ShadowNet edge function interface to obtain a 10 ms RTT in total.



(a)



(b)



(c)

Figure 3: Showing (a) our experimental setup, (b) detection time results, and (c) damage prevented by ShadowNet and sFlow under HTTP GET and UDP flooding attacks.

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 [35]. 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 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 address 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) [15] switches between the ShadowNet edge function and the ShadowNet service and victim VMs (see Figure 3(a)). These switches run sFlow [18] 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.

Experimental Results

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 that any of the systems raises an alert after the attack has started. For an HTTP GET flooding attack, 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 the ShadowNet server is counting UDP packets (Layer 4) and the sFlow agent is counting Ethernet frames (Layer 2) at the switch, the ShadowNet service provides faster detection because it does not rely on connection-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 five minutes. We assume a mitigation system that provisions network policies with a configuration delay 0.5 seconds [22]. 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 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, injecting shadow packets of size 70 bytes at a rate of 8500 pps on the fast path. This represents 4.76 Mbps for 0.62 seconds, or 368.9 KB of traffic. Similarly, the HTTP flooding attack is detected in 2.46 seconds, generating 280 Kbps, or 86.10 KB of traffic.

6. DISCUSSION

The promising preliminary 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 offer 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 seen 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 current form shadow-packets are generated *per request* for each service ShadowNet is protecting. This may pose high overhead at 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 [31] that can be incorporated in edge-functions. However, trade-offs in the information carried in the shadow packet necessary to support additional deeper analysis and the impact on the fast-path needs to be clearly studied.

More broadly, we are proposing to create a lightweight (binary) and faster replica of the Internet that could foretell the state of the real Internet in the future. If successful, this could pave the way for robust IoT deployments, accelerate IoT adoption, and have potentially transformative impact on enabling new zero-time response services by leveraging edge computing. Examples of such other uses of ShadowNet could include preemptive resource provisioning at a web service, performing in-network privacy preserving analytics, etc.

7. RELATED WORK

Along the many years of DDoS history, researchers have proposed many detection and mitigation mechanisms for DDoS. Application-level DDoS attacks require less bandwidth and are stealthier than the network/transport-level attacks. However, the rise of IoT-based botnets has enabled attackers to launch application level attacks at a very larger scales. The approaches include source-end detection [42, 36], secure overlay networks [43, 25, 39], SDN/NFV based approaches [37, 40, 27, 26] and cloud based approaches [38, 44]. ShadowNet requires the existence of a edge infrastructure and uses the concept of source-end detection by enabling *just-in-time* deployment on ShadowNet edge functions compared to D-WARD [36] that proposed an inline system installed in access/border routers and ADS (Large-scale Automated DDoS detection System) [42] that collects available data in the form of SNMP and NetFlow logs at a tier-1 ISP. Contrary to DefCOM [39], SOS [25], Akamai secure overlay [43], that create their overlays on top of the Internet infrastructure, ShadowNet tries to leverage hardware and software assisted mechanisms to build the *fast-path*. SDN-based DDoS mitigation solutions [26] use network infrastructure and assume ISP based deployment of solution. ShadowNet proposes to use edge infrastructure and can be deployed by web services on edge infrastructure. Going beyond cloud-based services, ShadowNet presents an approach that could prevent damage to the Internet infrastructure itself. Finally, ShadowNet leverages edge deployment research and securing communication with edge devises [28, 34, 22] to devise a solution to IoT-DDoS attacks.

8. SUMMARY

We propose ShadowNet as an approach to prevent application-level IoT-DDoS attacks by leveraging 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 its implementation and perform additional evaluations to study trade-offs and its effectiveness in realistic conditions.

9. 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] Global DDoS threat landscape q4 2016. , 3 2017.

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

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

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

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

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

[17] 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.

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

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

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

[21] Arnautov, S., Trach, B., Gregor, F., Knauth, T., Martin, A., Priebe, C., Lind, J., Muthukumaran, D., O'Keeffe, D., Stillwell, M. L., Goltzsche, D., Eyers, D., Kapitza, R., Pietzuch, P., and Fetzer, C. Scone: Secure linux containers with intel sgx. In *Proceedings of the 12th USENIX Conference on Operating Systems Design and Implementation*, OSDI'16.

[22] 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*.

[23] 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*.

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

[25] 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.

[26] Fayaz, S. K., Tobioka, Y., Sekar, V., and Bailey, M. Bohatei: Flexible and elastic ddos defense. In *Usenix Security* (2015).

[27] 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*.

[28] Ha, K., Pillai, P., Richter, W., Abe, Y., and Satyanarayanan, M. Just-in-time provisioning for cyber foraging. In *Proceeding of the 11th Annual International Conference on Mobile Systems, Applications, and Services* (2013), MobiSys '13.

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

[30] Hu, Y. C., Patel, M., Sabella, D., Sprecher, N., and Young, V. Mobile edge computingâĂŤa key technology towards 5g. *ETSI White Paper 11* (2015).

[31] 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.

[32] 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.

[33] 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).

[34] Madhavapeddy, A., Leonard, T., Skjegstad, M., Gazagnaire, T., Sheets, D., Scott, D., Mortier, R., Chaudhry, A., Singh, B., Ludlam, J., Crowcroft, J., and Leslie, I. Jitsu: Just-in-time summoning of unikernels. In *12th USENIX Symposium on Networked Systems Design and Implementation (NSDI 15)*.

[35] 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.

[36] 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.

[37] 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.

[38] 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).

[39] 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.

[40] 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.

[41] 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.

[42] 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.

[43] 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.

[44] 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.