2017 IEEE 41st Annual Computer Software and Applications Conference

Secure Double-layered Defense against HTTP-DDoS Attacks

Mohamad Samir A. Eid
Dept. of Electrical Engineering & Information Systems
The University of Tokyo
Bunkyo-ku, Tokyo, 113-8656, Japan
Email: eid.msa@acm.org

Hitoshi Aida
Dept. of Electrical Engineering & Information Systems
The University of Tokyo
Bunkyo-ku, Tokyo, 113-8656, Japan
Email: aida@ee.t.u-tokyo.ac.jp

***Abstract*—A major cyber-security concern to date for webservers are Distributed Denial of Service (DDoS) attacks. Previously we proposed a novel overlay-based method consisting of distributed network of public servers (PS) for preparation, and access nodes (AN) for actual communication. The AN's performance is evaluated under difficult to detect HTTP(S)-DDoS attacks [1]. Yet, attackers may attempt service denial by attacking the PS instead.**

The focus in this paper is on mitigating complex slow-requesting HTTP-DDoS attacks that target the PS. A proof-of-concept prototype is implemented with simplified countermeasures and tested. We report on the results of two experiments. Results suggest that the simple PS role can enable high mitigation factors of both high-rate and low-rate attack traffic per source, even with 10,000 unique attack sources per target PS, acting as a second layer of defense with the AN. Yet, with a cost of longer time to load the requested resource file in comparison to direct access.

1. Introduction

Distributed denial of service (DDoS) attacks are a top cyber threat to webservers, due to its simple yet effective nature [2], [3]. Attackers overwhelm servers with requests from distributed sources, so users are denied service. Generally, attacks are either on the low-level (utilizing transport or network layer protocols) or high-level (utilizing application layer protocols). Low-level DDoS generally rely on traffic volume to overwhelm the server. On the other hand, high-level DDoS rely on asymmetry, in which a small number of client requests can cause considerable server resource consumption. So, their request rate is much smaller and harder to detect [4].

DDoS mitigation solutions can be either; locally-based (i.e., at server's premise), or remotely-based (i.e., at source, infrastructure, ISP, or overlay network). Locally-based solutions can be effective, but the increase in attack volumes and sophistication require vast spending on purchasing, maintaining, and upgrading such solutions [5]. Only large-sized enterprises may afford such expenditures. On the other hand, small and medium-sized enterprises (SME) require affordable, scalable, and practical solutions (i.e., overlay-based mitigation, managed by a specialized third-party).

Overlay-based mitigation of low-level DDoS is well established, given enough resources at the third-party mitigation service provider. On the other hand, mitigating high-level attacks far from the server is a challenging task. Not surprisingly, recently more frequent high-level attacks are reported [2], especially HTTP based DDoS (i.e., HTTP-DDoS) attacks [6].

Also, conventional overlay-based solutions share at least one of two demerits. 1) *Traffic decryption*: The client-server SSL connections are split to inspect the content for mitigation, with overlay nodes decrypting then re-encrypting traffic [7], [8], [9]. Yet this prompts trust concerns, especially given the recent rise in consumer awareness [10]. This has caused the majority of financial service organizations for example to control encryption keys within their organization rather than at a third-party provider [11]. 2) *Limited identification*: Mitigation per-IP or per-connection (i.e., two-level identification) is a conventional best practice [12]. Yet, this limits the ability to detect complex HTTP-DDoS. For example, mitigation per-connection fails in case of a single-request per-connection attack, or an attack below the detection thresholds. Likewise, mitigation per-IP may result in errors in case of a multi-behavior per-shared-IP traffic, and in punishing a group of clients as a single one, even after the attack stops.

In [1], we proposed the first overlay-based method that enables effective mitigation of complex high-level attacks by enhanced identification of clients far from the server, while complying with true-end-to-end encryption of client-server transactions (or non-split SSL). Distributed special purpose Access Nodes (AN) are implemented, through which alone the client-server communication takes place. In addition, special purpose Public Servers (PS) act as initial preparation points. Evaluations with complex attack conditions towards the AN show promising results. Yet, attackers may attempt to deny clients from reaching to the AN by attacking the PS instead. So, it's necessary for the PS to mitigate such anticipated attacks an extra step away from webservers.

So, the focus in this paper is on evaluating the proof of concept prototype of the PS under high-level DDoS. More specifically, the evaluation focus is on slow-requesting HTTP-DDoS as we explain in section 3.2. The prototype is deployed on DeterLab [13], where several experiments are conducted to evaluate the soundness of the concept. Among them, we report on the results of two experiments with complex slow-requesting HTTP-DDoS conditions. The popular Slowloris tool (among attackers) and a custom-built attack tool are utilized for evaluation. Results suggest that the PS with its simple role can act as a second layer of defense with the AN enabling high mitigation factors of both high-rate and low-rate attack traffic per source, even against 10,000 unique attack sources per target PS. Yet, with a cost of longer time to load the requested file (i.e., service time) in comparison to direct access.

The rest of this paper is organized as follows. Section 2, surveys selected related work on the DDoS problem. Section 3 describes our proposed method. In section 4 and 5, the evaluation results are explained and discussed. Finally, section 6 concludes the work.

0730-3157/17 $31.00 © 2017 IEEE
DOI 10.1109/COMPSAC.2017.207

2. Related Works

Related commercial overlay-based solutions can mitigate HTTP-DDoS attacks [7], [8], [9], [12]. In addition, many overlay-based approaches are proposed in literature. For example, employing software defined networking (SDN) [14], [15], dynamically instantiated replica servers [16], human verification challenges [17], and content caching [18]. However, they either suffer from traffic decryption, limited identification, or both.

Enhanced identification can be achieved remotely for example by utilizing packet header fields as marking fields to distinguish between different clients per-IP [19], [20]. These can be combined for example with the work in [21], which utilizes an ensemble of adaptive and hybrid neuro-fuzzy systems and extract several features from TCP flows for detection. Or, the work in [22] which identifies attack and non-attack flows by analyzing packet sizes. Yet, requiring changes to core routers is problematic.

Additionally, several statistical and machine learning approaches have been previously proposed. For example, the work in [23] adopt Adaptive Selective Verification (ASV) to detect slow-requesting HTTP-DDoS. Similarly, the research in [24] propose constructing connection scores based on traffic attributes, including the browsing behavior (e.g., requested page type and popularity). Further, entropy-based approaches can detect DDoS attack connections, such as the work in [25]. However, content inspection for detection is a common demerit.

In addition, several mitigation approaches utilize game-theoretic techniques [26], [27], [28], [29]. In general, DDoS attacks are modeled as a game between each attacker node and the defender. The concepts of such techniques are generally based on deciding the optimal defense parameters (e.g., detection thresholds' values), against the attack parameters (e.g., number of attack nodes and attack rate per node). However, such methods if deployed remotely, inherit the common demerit of limited identification. Consequently, an attacker with a single-request per-connection attack category can evade similar detection methods.

3. Proposed Method and Implementation

3.1. System Overview

The proposed method utilizes a third level of client identification, in addition to the conventional per-IP and per-connection, which we call per-session identification [1]. A protected web server, i.e., the secret server (SS), is locally-based, and hidden from direct low-level access. Per-session identification is realized far from the SS by adding a preparation stage, which is handled by public servers (PS), while access nodes (AN) relay the C-SS encrypted transactions. Thousands of the special purpose ANs and PSs are assumed that are geographically distributed and managed by mitigation-service providers. So, all the SSs are subscribers to the mitigation-service, sharing all ANs and PSs, making it a cost-efficient solution especially for SMEs.

At first (step 1), each client normally enquirers about the IP address of the desired web server. Assuming a DNS-based failover mechanism in place, the IP addresses of an available shared PS is returned. Each client first connects normally to one of the PSs and sends its initial HTTP request. Upon receiving the client's first request (step 2), a PS performs its own detection measures, then selects one of the suitable ANs for the client (step 3). In turn, the

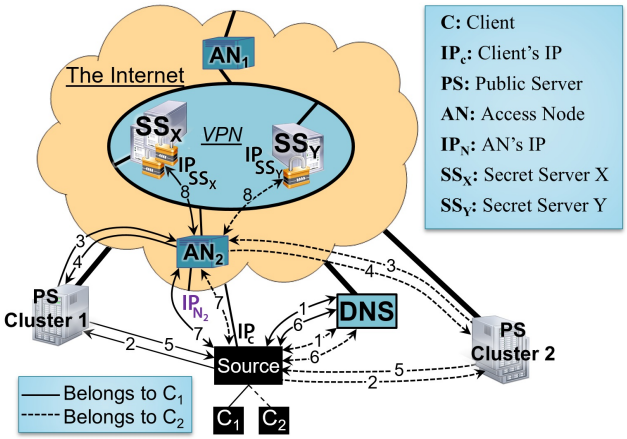

Figure 1. Architecture of the proposed method.

AN replies with a new session identifier (SID) to each PS (step 4). The PS's preparation stage ends in step 5 by sending a HTTP redirection response with the new location of the service, at the selected AN's assigned port for each client. A successful client must make use of the leased SID in time. This means, firstly, enquirering about the selected AN's address in step 6. In this example, each client is given the same IP_{N_2}. Then successfully connects to the designated AN port and passes the initial detection measures at the AN (step 7). If all steps are completed as intended, the client is now qualified to get the desired service (step 8) and proceed with its end-to-end encrypted communication phase.

Since the SS's private key is strictly only managed locally, so the PS can't redirect to a non-authentic server without the client realizing it, and the AN cannot read or modify any C-SS data. So, server authenticity, data confidentiality and integrity are preserved.

3.2. Attack Countermeasures

From the perspective of attack impact, single-request per-connection attacks would release the PS's resources fast as it repeats step 2, while multiple-requests per-connection attacks are detectable promptly by the PS (since only 1 request per connection is expected). So, in the discussed evaluations in section 4 we consider the worst case where the attacker knows the likely most effective HTTP-DDoS attack category against the PS, which is slow-requesting HTTP-DDoS. We later discuss other attack conditions in section 5.2.

A slow-requesting HTTP-DDoS attack source typically opens several seemingly-legitimate connections to the target server and then starts sending never-completing partial HTTP request headers, part by part. This kind of behavior is designed to tie down the server that's waiting for the rest of the slowly-arriving request. The attack source typically reestablishes the connections over which the server times out their requests.

In the current proof of concept prototype, we include a simplified countermeasure against slow-requesting HTTP-DDoS. Assume a client that connects at a time reference $t = T_C$. Then the PS sets two timers. The first, $T_{r(max)}$, is the maximum allowed duration between the client's connection establishment and the arrival of the first request part. So, if the first part arrives at

$t = T_1$, then the condition $T_1 < T_C + T_{r(max)}$ must be satisfied. Otherwise this client's attempt is refused and the client's penalty is updated accordingly. Finally, if the full message (i.e., the whole first request) arrives at $t = T_n$ (for n request parts), i.e., takes $T_n - T_1$ seconds to be fully aggregated, consequently the condition $T_n < T_1 + T_{a(max)}$ must also be met, where $T_{a(max)}$ is the maximum duration allowed for a slow request to arrive fully after its first part. Note that if the request arrives as a single part (n = 1), then $T_n = T_1$.

Additional local client reputation component at the PS is omitted from this paper, being beyond its scope and limited size. In brief, the PS monitors behavior attributes, including the number of simultaneous attempts (A_i), number of attempts updated every time step (i.e., per α [sec]) (A/α), failed attempts per α (FA/α), and the flag Warn_{AN} raised in case of a warning message received form the AN. Also, the PS is equipped with a unique slow-response client-probing countermeasure as with the AN [1].

It's desired to not only achieve a high mitigation factor of attack, but also have the mitigation cost at its lowest (see section 4.2). Note that a slow-requesting HTTP-DDoS attack towards the PS is not expected to affect the AN, since the first client request must be fully received before contacting the AN. Also, scaling the number of PSs is simple due to its role. However, it's also desired that each PS can handle large attacks, in terms of request rate and simultaneous number of sources, and still offer acceptable service. This is investigated in the next section.

4. Evaluation

In this paper we discuss two of our conducted experiments that investigate slow-requesting HTTP-DDoS against the PS. We assume a DNS-based failover mechanism is already in place (i.e., step 1 in section 3.1), so we conduct our evaluations on a single PS. Also, attacks that target the DNS itself are out of this paper's scope, while other attack conditions are discussed in section 5.2. It's assumed that the attacker tries to cause damage to the service availability by focusing the finite attack resources on either the AN or the PS. So, the attack sources in the discussed experiments focus on the PS, while the cases of attacks that proceed to the AN are considered in separate experiments (including slow-requesting HTTP-DDoS) [1].

DeterLab [13] is chosen for deployment due to the disruptive nature of the traffic generated, the dedicated node access, and to simplify repeatability. Values for the prototype's parameters need not be optimal to demonstrate the concept's soundness. From experience, α is set to 30 [sec] being neither too large nor too small, and the values of $T_{r(max)}$ and $T_{a(max)}$ are set to 2 and 3 seconds, respectively.

4.1. Benchmarks

Before deploying the mitigation system, direct non-attack benchmarking measurements on the Apache webserver to be used as SS are conducted (i.e., before hiding the SS). Utilizing ApacheBench (version 2.3), the server's local limit of requests per second is measured, which can be a function of multiple factors, such as; the requested resource and server's configuration. For the requested 200 KB file, and with no request concurrency, the non-attacked server showed a local limit of 54.6 [r/s] (i.e., requests per second) and a 19.6 [ms] service time in average. Although this limit can be raised by tuning the server's configuration, it's kept unchanged because the focus of this research is on evaluating mitigation of attack far from the server. As we raise the concurrency setting of ApacheBench, for example to 20, 50, 100, and 300 concurrent requests, so rises the measured service time to 365.04 [ms], 912.65 [ms], 1,825.38 [ms], and 3,995.18 [ms], respectively.

On the other hand, direct attack benchmarks towards the SS utilizing only 20 Slowloris attack sources (as in section 4.3) result in a 0.0% chance of service completion (CoS) for non-attack clients. Section 4.2 explains more about CoS as a metric.

4.2. Evaluation Metrics

Two performance metrics are considered in this article to measure the effectiveness of mitigation, namely; mitigation factor and mitigation cost. The mitigation factor (MF_i) is the factor of attack traffic reduction (RF_i) multiplied by the chance of service completion (CoS_i) under attack for each time step *i*. $CoS_i = \frac{Resp_i}{Req_i}$, where Req_i is the number of non-attack requests sent, and $Resp_i$ is the corresponding number of service responses completed. RF_i is the ratio of attack traffic reaching the SS to that generated. Yet, for an HTTP-DDoS attack against the PS, attack traffic reaching the SS equals zero. Therefore, in the discussed experiments RF_i = 1 and $MF_i = RF_i \times CoS_i = CoS_i$.

However, a high CoS_i alone doesn't describe the attack's impact on the service during and after the attack. Therefore, we define the mitigation cost (MC_i) as the increase in service time (ST), for a specified resource file, from its pre-attack value. For each request we define ST as $t_{resp_{full}}$ - $t_{req_{sent}}$, where $t_{resp_{full}}$ is the time the response is fully received by client and $t_{req_{sent}}$ is the time the request was sent. So, $MC_i = ST_i - \overline{ST}$, where $\overline{ST}$ is the measured value in average before attack and ST_i is the average service time for all measurement requests within time step *i*.

4.3. Experiment 1: Slow-Requesting HTTP-DDoS via PS

This experiment investigates the effectiveness of the mitigation system against an anticipated complex DDoS attack category towards the PS: slow-requesting HTTP-DDoS.

A single SS, AN, and PS are used, each on a dedicated bpc2133 physical testbed node [31]. In addition, 100 measurement sources, each with a unique source IP, are emulated on a bvx2200 based measurement node (MN), generating 100 HTTP GET requests per 30 seconds for a 200 KB file from the 100 different source IPs. The resulting measurements are used to plot versus time the CoS_i and ST_i with 95% confidence interval (CI).

Testbed link speeds of 100Mbps are used. Each of these node's OS is Linux version 3.2.0-64, while the SS is running the open-source Apache server unmodified (version 2.2.22).

In this experiment, the attacker is exceeding the detection threshold (i.e., high-rate attack). Attack traffic is distributed over 20 unique attack sources that are deployed on 7 bpc2133, 4 bvx2200, and 9 bpc3000 physical testbed nodes [31] targeting the PS. The OS installed on the attack nodes is Kali Linux version 1.0.7.

The slow-requesting HTTP-DDoS attack tool of choice for evaluation is the Slowloris tool, which is commonly utilized by attackers to target webservers [32]. So, Slowloris (version 1.0) is manually deployed on the 20 nodes, each representing a unique attack source.

Each attack source runs 1,000 threads (i.e., unmodified default value of Slowloris), where per attack thread the sleep time between consecutive request parts is set to 2 seconds (recommended value by the Slowloris built in (pre-attack) *tester* when run against the PS). So, each thread initiates an attempt to the PS and renews it separately, while per a single attempt there's a single fixed but incomplete request arriving at a high rate, and the tool keeps the slow-requesting connection active if the PS doesn't terminate it. In the measured resulting attack traffic, the PS is observing nearly 2,440 new connection attempts per second, with 10,000 concurrently requesting connections (i.e., simultaneous attempts per second). In total, the generated HTTP-DDoS attack rate is 5,000 request parts per second, i.e., 250 request parts per second per source.

In this setup, the PS's IP address is 10.1.1.4, and therefore the attack command is as follows:

perl slowloris.pl -dns 10.1.1.4 -port 80 -timeout 2

The attack lasts for 200 minutes, of which the first 40 minutes are shown in Fig. 2. The four indicators plotted at the top of Fig. 2 are recorded by the target PS once every α seconds describing the attack volume. The first is the number of source IPs, which equals 100 before the attack, indicating the measurement sources, then rises by an extra 20 IPs representing the start of attack at t = 4 [min]. The number of IPs at every given time step is represented on the secondary axis, to the right. The second indicator is the number of simultaneous attempts, which represents how many open connections at the moment of observation, since only one attempt is allowed per connection at the PS. Thirdly is the rate of new connections per second, as old attack connections are constantly being terminated by the PS once they fail to send the full request in time. The last indicator measured is the rate of request parts per second, which indicates that a single attack connection slowly sends approximately 2 parts before being terminated by the PS.

During attack, the PS shows a 100% MF_i with a nearly flat mitigation cost (i.e., increase in ST_i) via both the PS and the AN, which is higher than the benchmarks. Also, note that at t = 36 [min], and for about one minute, the coordinated attack has been stopped deliberately from the source then resumed. This was done to observe any possible transient effect on the PS's performance. The experiment resumed for the rest of the 200 minutes with the same performance unchanged.

4.4. Experiment 2: Distributed Low-rate Slow-requesting HTTP-DDoS via PS

We proceed with a more complex attack arrangement, also utilizing the slow-requesting HTTP-DDoS attack category towards the PS. The number of attack sources is increased to 10,000 unique sources, this time with low-rate attack conditions per source. The assumption is that the failed attacker may attempt to distribute the attack traffic requests over larger number of sources to evade detection.

All nodes are automatically assigned by DeterLab from the MicroCloud, pc2133, pc3060 physical node types [31]. The AN and SS are each running on a dedicated MicroCloud node. Additionally, the PS is running on pc2133 node. The measurements setup is similar to experiment 1, but emulated on a pc3060 based testbed node. Testbed link speeds of 100Mbps are used. Each node's OS is Linux V3.13.0-62, while the SS is the same as experiment 1.

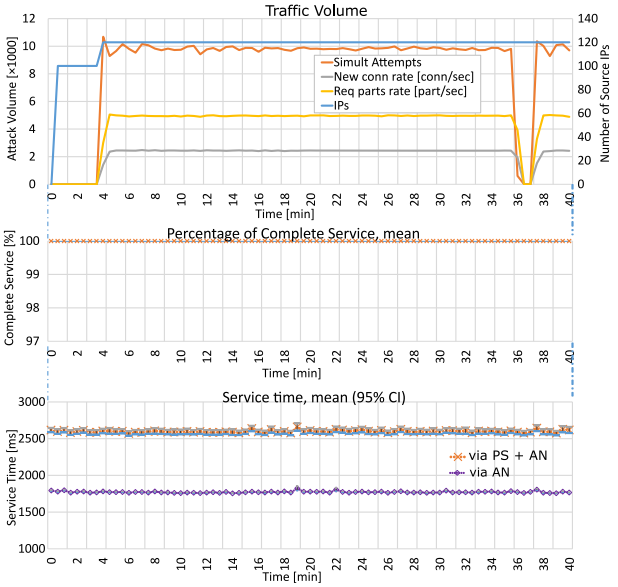

Figure 2. Results of Experiment 1.

We learned with experience that up to 1,000 virtual IP addresses can be set up per single DeterLab node reliably. Any larger value is avoided as it results in unexpected disconnections of attack nodes later during the experiment.

So, in this experiment, a total of 10,000 unique attack sources represent the attack population (emulated on 5 MicroCloud, 2 pc3060, and 3 pc2133 physical nodes). A custom attack tool is developed that can be deployed on thousands of sources and be remotely controlled simply by a single commander node. The custom tool also enables the execution of slow-requesting DDoS attack behavior with tunable attack parameters and remote command and control of attack over the network. So, each physical node runs 1,000 threads of the custom tool. Each single thread creates a local virtual network interface with a unique source IP address. Per source, a legitimate HTTP GET request is constructed and split into small parts of size = 20 Bytes each. Each attacking source IP initiates a single connection attempt at a time to the PS, while per a single attempt there's a single fixed but incomplete request arriving with 1 [sec] time separation between parts. The tool keeps the connection active until the full request is sent (and response is served) or until the PS terminates the connection. A closed connection is then automatically renewed.

The total duration of the attack is 12 hours, which is considered a long-duration attack [4]. Logs are recorded at the PS and MN. Results from three time intervals of the whole experiment are presented in Fig. 3. The attack is switched OFF and ON multiple times immediately after its start and later after 10 hours to observe the possible effect on the PS's performance observed initially. The resulting total rate of new connections per second is nearly 1,650 [conn/sec] (i.e., about 10 connections per minute per source). The total rate of request parts per second is nearly 6,450 [part/sec] (i.e., per source, about 39 parts per minute). The total number of simultaneous attempts at any instant is nearly 6,500 [attempts]. Per source, there are no simultaneous attempts.

Results show a 100% MF_i with a 95% confidence interval.

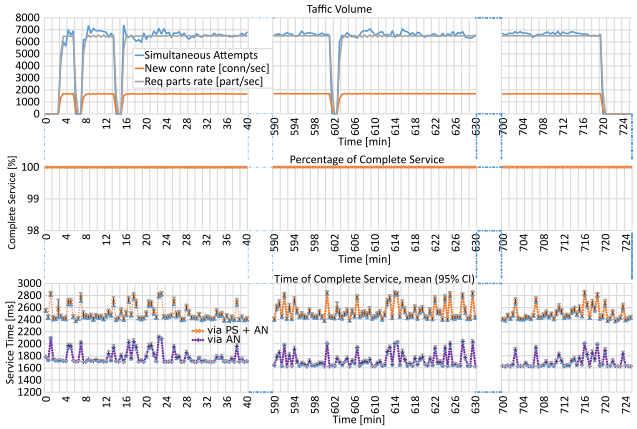

Figure 3. Results of Experiment 2.

From the perspective of ST_i, the AN's performance is apparently not affected by this attack, being directed at the PS. This indicates that the attack traffic is not reflected by the PS on the AN. As for the PS's performance, a mitigation cost is observed. The PS contributes by nearly 1,200 [ms] to the whole (PS + AN) service time. The cost added by the PS resides within 800 to 1,200 [ms] (i.e., in addition to the AN's own added service time), with 95% confidence margin. Section 5.1 further discusses the latency cost.

5. Discussion

5.1. Latency Cost

The proposed method, with the extra connection steps, introduces latency cost even in absence of attack. However, as the benchmarks in section 4.1 suggest, higher request concurrency towards the unprotected server means higher latency too. So, latency variation is expected, even without the proposed system in place. For example, the service time of the requested file rises to 3,995.18 [ms] with only 300 concurrent requests (section 4.1). On the other hand, as experiments suggest, this prototype can mitigate nearly 10,000 concurrent attempts per single PS with a flat latency cost of 2,600 [ms] for the first requested file only, and 1,800 [ms] for subsequent files (i.e., via AN). Also, in the current prototype, each separate function within the AN (i.e., relaying C-SS bytes, managing SID records, etc) is handled by a single thread, so, we plan to investigate the effect on latency of increasing the number of relaying threads per AN.

5.2. Other DDoS Attack Conditions

In this paper, we consider the possibility of HTTP-DDoS attacks towards the PS, with the focus on slow-requesting attack conditions as explained in section 3.2. However, attackers can attempt larger and more distributed attacks than the considered cases. We argue that in contrast to a conventional webserver, the PS expects only one request per client, and performs constant simple tasks regardless of the requested resource. This may explain the difference in impact of request concurrency on the unprotected SS (section 4.1) compared to the PS prototype (sections 4.3 and 4.4).

In addition, the PS expects TCP connections from any source at any time to its fixed port 80, so it may attract low-level attacks such as a TCP SYN flood [30]. Yet, low-level attacks can not directly affect the SS. In the worst case, a massive low-level attack can saturate the link to several of the shared PSs, which then can be sacrificed to stop such attack as discussed in section 5.3.

On the other hand, each AN expects only connections from specified source IPs to specified port numbers at specified times. Even if the attack is via legitimately acquired SIDs, attack traffic is spread over multiple ANs beyond the attacker's choice. Otherwise, low-level attack traffic can only hope to congest the links to the thousands of ANs which simply drop incorrect SID connection attempts.

Furthermore, we consider additional complex (i.e., difficult to detect) high-level attack categories towards the AN in [1], namely; single-request per-connection, multi-behavior per-shared-IP, sub-detection-thresholds, slow-requesting HTTP-DDoS, and multi-vector attacks.

Another model of DDoS attacks is called Economic Denial of Sustainability (EDoS). The concept is based on engaging a paid-for service, such as an on-demand DDoS mitigation service, to cost the SME economically every time the attack is executed. However, the economic model of EDoS doesn't hold in case of an always-on shared solution such as the proposed method.

5.3. Scalability

Scalability against attacks is fully the mitigation solution's responsibility, regardless of the server's capacity to handle traffic, assuming enough over provisioning of overlay nodes. Reported large multi-Gbps attacks are on the low-level [4], so it can't reach to the SS directly anyway in the proposed scheme. Conversely, high-level attacks are inherently much smaller in volume since they can't be amplified, and to evade detection. In experiments 1 and 2, we tested with a significant traffic volume per single overlay-node, for example considering the reported peak high-level attack rate of 268,800 [r/s] [4]. So, for example with the proof of concept prototype PS withstanding more than 5,000 HTTP request parts per second, then we argue that nearly 54 PSs can mitigate the reported peak.

Whereas as an overlay-based solution, scalability with legitimate traffic is achievable by distributing traffic over thousands of nodes, thus reducing the possibility of choke points. Yet, it also depends on the local server's capacity (i.e., shared responsibility), since the proposed method is geared towards end-to-end-encrypted locally-managed content.

5.4. Conventional Solutions

Effective mitigation of HTTP-DDoS attacks can be achieved by utilizing conventional commercial overlay-based solutions. Those assume either a split SSL connection between the client and server, or limited (per-IP and per-connection) identification, or both [7], [8], [9], [12]. However, recent studies suggest that authorizing a third-party mitigation service provider to manage the server's private key or to decrypt the C-S traffic may not be acceptable by large sector of organizations and web users [10], [11].

Furthermore, in academia, we don't know of a proposed overlay-based method for mitigating HTTP-DDoS that assumes non-split end-to-end-encrypted client-server connections. Utilizing enhanced identification, our implemented proof-of-concept prototype shows superior chance of service in contrast to related

methods that inspect the content, yet, with a latency cost. For example, the simulations by [23] with 280 attack sources, show an 80% "Client Success Ratio" (i.e., CoS), while in [24], experimental results with 600 attack sources show nearly 56% reduction in attack traffic (i.e., RF).

6. Conclusion

This paper reported on how our practical DDoS mitigation concept can prevent slow-requesting HTTP-DDoS attacks against the PS component. The new prevention functionality was designed and added to the implemented proof of concept prototype. Through experimentation, the system was tested on DeterLab against the Slowloris attack tool and against our custom built slow-requesting DDoS attack tool. Results suggest that the PS's simple role can act as a second layer of defense with high mitigation factors of both high-rate and low-rate attack traffic per source, even against 10,000 unique attack sources per target PS. However, with a cost of longer time to load the requested resource file (i.e. service time) in comparison to direct access. As part of our future work, we plan to study the effect of additional thread concurrency of prototype nodes on the mitigation factors and cost, and further developing a shared PS-AN client-reputation system.

Acknowledgments

The authors would like to thank; The Japan Student Services Organization (JASSO), The Japanese Ministry of Education, Culture, Sports, Science and Technology (MEXT), The KDDI Foundation, and The Teijin Foundation, for supporting this research. We also thank the DETER team for their experiment support.

References

[1] M. S. A. Eid, and H. Aida, "Trustworthy DDoS Defense: Design, Proof of Concept Implementation and Testing", IEICE Transactions on Information and Systems, Special Section on Information and Communication System Security, Vol. E100-D, No. 8, Aug. 2017. In press.

[2] Kaspersky: DDoS attacks in Q4 2016, Feb 2017. [Online; accessed 20-Apr-2017]

[3] McAfee Labs: Threats Report - March 2016. [Online; accessed 20-Apr-2017]

[4] Incapsula Report: 2015-2016 Annual DDoS Threat Landscape Report, Aug 2016. [Online; accessed 20-Apr-2017]

[5] J. McCallion, "A10 storms the DDoS prevention market with Thunder TPS", 2014. [Online; accessed 9-Apr-2017].

[6] Arbor: Worldwide Infrastructure Security Report - Vol. XI, 2016. [Online; accessed 20-Apr-2017]

[7] CloudFlare, "SSL Options". [Online; accessed 9-Apr-2017].

[8] Akamai, "Kona DDoS defender". [Online; accessed 9-Apr-2017].

[9] VeriSign, "DDoS protection services". [Online; accessed 9-Apr-2017].

[10] R. Goldberg, "Lack of Trust in Internet Privacy and Security May Deter Economic and Other Online Activities", U.S. Dept. of Commerce: National Telecommunications & Information Administration (NTIA), 2016.

[11] Encryption Application: Trends Study - Thales e-Security, Inc., June 2016. [Online; accessed 20-Apr-2017]

[12] AWS Best Practices for DDoS Resiliency, June 2016.

[13] T. Benzel, "The Science of Cyber Security Experimentation: The DETER Project", Proc. 27th Annual Comp. Sec. Appl. Conf, ACM, pp.137-148, 2011.

[14] N. I. Mowla, I. Doh, and K. Chae, "Multi-defense Mechanism against DDoS in SDN Based CDNi", 8th Int. Conf. on Innov. Mob. & Int. Serv. in Ubiq. Comput., pp.447-451, 2014.

[15] J. Li, S. Berg, M. Zhang, P. Reiher, and T. Wei, "Drawbridge: Software-defined DDoS-resistant traffic engineering", ACM SIGCOMM Comp. Comm. Review, vol.44, pp.591-592, Aug 2014.

[16] Q. Jia, H. Wang, D. Fleck, F. Li, A. Stavrou, and W. Powell, "Catch Me If You Can: A Cloud-Enabled DDoS Defense", 44th Annual IEEE/IFIP International Conf. on Dependable Systems and Networks,pp. 264-275, 2014.

[17] Z. Al-Qudah, B. Al-Duwairi, and O. Al-Khaleel, "DDoS Protection As a Service: Hiding Behind the Giants", Int. J. Comput. Sci. Eng., vol.9, pp.292-300, 2014.

[18] Y. Gilad, A. Herzberg, M. Sudkovitch, and M. Goberman, "CDN-on-Demand: An Affordable DDoS Defense via Untrusted Clouds", In Network & Dist. Sys. Security Symp. (NDSS), 2016.

[19] M. Jog, M. Natu, and S. Shelke, "Distributed and Predictive-Preventive Defense Against DDoS Attacks", Proceedings of the 2015 International Conference on Distributed Computing and Networking, 29:1-29:4, ACM, 2015.

[20] V. Aghaei-Foroushani, and A. N. Zincir-Heywood, "Investigating unique flow marking for tracing back DDoS attacks", IFIP/IEEE Int. Symp. on Integrated Network Management, pp.762-765, 2015.

[21] P. A. R. Kumar, and S. Selvakumar, "Detection of distributed denial of service attacks using an ensemble of adaptive and hybrid neuro-fuzzy systems", Computer Comm., vol.36, pp.303-319, 2013.

[22] L. Zhou, M. Liao, C. Yuan, Z. Sheng, and H. Zhang, "DDOS attack detection using packet size interval", 11th International Conference on Wireless Communications, Networking and Mobile Computing (WiCOM 2015), pp.1-7, 2015.

[23] Y. G. Dantas, V. Nigam, and I. E. Fonseca, "A selective defense for application layer ddos attacks", in IEEE Joint Intelligence and Security Informatics Conference (JISIC), pp. 7582, Sept 2014.

[24] H. Beitollahi and G. Deconinck, "Connectionscore: a statistical technique to resist application-layer ddos attacks", J. of Ambient Intelligence & Humanized Computing, vol.5, no.3, pp.425442, 2014.

[25] W. Zhou, W. Jia, S. Wen, Y. Xiang, and W. Zhou, "Detection and defense of application-layer DDoS attacks in backbone web traffic", Future Gen. Comp. Sys., vol.38, pp.36-46, 2014.

[26] M. Wright, S. Venkatesan, M. Albanese, and M. P. Wellman, "Moving Target Defense Against DDoS Attacks: An Empirical Game-Theoretic Analysis", Proc. of the 2016 ACM Workshop on Moving Target Defense, pp.93-104, 2016.

[27] Y. Liu, D. Feng, Y. Lian, K. Chen, and Y. Zhang, "Optimal Defense Strategies for DDoS Defender Using Bayesian Game Model Information Security Practice and Experience", 9th International Conference, ISPEC 2013, pp.44-59, May 2013.

[28] C. B. Simmons, S. G. Shiva, H. S. Bedi, and V. Shandilya, "ADAPT: A Game Inspired Attack-Defense and Performance Metric Taxonomy" 28th IFIP TC 11 International Conference, pp.344-365, 2013.

[29] T. Spyridopoulos, G. Karanikas, T. Tryfonas, and G. Oikonomou, "A game theoretic defense framework against DoS/DDoS cyber-attacks", Comp. & Security, vol.38, pp.39-50, 2013.

[30] M. S. A. Eid, and H. Aida, "Securely hiding the real servers from DDoS floods", 10th IEEE/IPSJ International Symposium on Applications and the Internet (SAINT), pp.165-168, July 2010.

[31] DETERLab Node Types, http://docs.deterlab.net/core/node-types/. [Online; accessed 9-Apr-2017].

[32] Radware: Common DDoS Attack Tools, Jan 2016. [Online; accessed 9-Apr-2017].