ResearchGate

See discussions, stats, and author profiles for this publication at: https://www.researchgate.net/publication/315497416

Trustworthy DDoS Defense: Design, Proof of Concept Implementation and Testing

Article *in* IEICE Transactions on Information and Systems · August 2017

DOI: 10.1587/transinf.2016ICP0024

CITATION
1

READS
99

2 authors:

Mohamad Samir A. Eid
The University of Tokyo
4 PUBLICATIONS **5** CITATIONS
SEE PROFILE

Hitoshi Aida
33 PUBLICATIONS **91** CITATIONS
SEE PROFILE

All content following this page was uploaded by Mohamad Samir A. Eid on 20 July 2017.

The user has requested enhancement of the downloaded file.

PAPER Special Section on Information and Communication System Security

Trustworthy DDoS Defense: Design, Proof of Concept Implementation and Testing

Mohamad Samir A. EID[†a)], *Nonmember and* **Hitoshi AIDA**[†b)], *Member*

SUMMARY Distributed Denial of Service (DDoS) attacks based on HTTP and HTTPS (i.e., HTTP(S)-DDoS) are increasingly popular among attackers. Overlay-based mitigation solutions attract small and medium-sized enterprises mainly for their low cost and high scalability. However, conventional overlay-based solutions assume content inspection to remotely mitigate HTTP(S)-DDoS attacks, prompting trust concerns.

This paper reports on a new overlay-based method which practically adds a third level of client identification (to conventional per-IP and per-connection). This enhanced identification enables remote mitigation of more complex HTTP(S)-DDoS categories without content inspection. A novel behavior-based reputation and penalty system is designed, then a simplified proof of concept prototype is implemented and deployed on DeterLab. Among several conducted experiments, two are presented in this paper representing a single-vector and a multi-vector complex HTTP(S)-DDoS attack scenarios (utilizing LOIC, Slowloris, and a custom-built attack tool for HTTPS-DDoS). Results show nearly 99.2% reduction in attack traffic and 100% chance of legitimate service. Yet, attack reduction decreases, and cost in service time (of a specified file) rises, temporarily during an approximately 2 minutes mitigation time. Collateral damage to non-attacking clients sharing an attack IP is measured in terms of a temporary extra service time. Only the added identification level was utilized for mitigation, while future work includes incorporating all three levels to mitigate switching and multi-request per connection attack categories.

key words: *Distributed Denial of Service (DDoS) attacks, application level attacks, network security, overlay networks.*

1. Introduction

Distributed Denial of Service (DDoS) attacks remains a top cyber threat to webservers, due to its simple yet effective nature [1, 2]. Attackers attempt to overwhelm servers with requests from distributed sources, so actual users are denied service. It can be driven politically, for money, fame, or as a cover for a simultaneous intrusion attempt. 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 victim. Conversely, 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 [3]. High-level DDoS attack models target one of three resources; network (e.g., download bandwidth, connection queues, etc), processing (e.g., CPU, memory, processing queues, etc) or economic (e.g., engaging per-usage paid-for services).

DDoS mitigation solutions can be either; locally-based (i.e., at server's premise), or remotely-based (i.e., at source, infrastructure, ISP, or an overlay network). Locally-based solutions can be effective, but coping with the increase in attack volumes and sophistication require enormous spending on purchasing, deploying, upgrading, and maintaining such solutions [4]. 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, for example by means of a reverse proxy, since only a high-level message may reach the server. So, the effect of low-level DDoS traffic can be prevented 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 [1], especially HTTP and HTTPS based DDoS (i.e., HTTP(S)-DDoS) attacks [5]. So, the focus of this paper is on overlay-based mitigation of HTTP(S)-DDoS, particularly on its complex versions, namely: single request per connection, multi-behavior per-shared-IP, sub-detection-thresholds, and multi-vector attacks.

Current overlay-based solutions share at least one of two demerits. 1) *Traffic decryption*: To inspect the content for mitigation, the client-server SSL/TLS (henceforth simply called SSL) connections are split, with overlay nodes decrypting then re-encrypting traffic [6–8]. Yet it prompts trust concerns, given the recent rise in awareness about encryption because of multiple factors, including; the continuous rise in cyber-attacks, privacy compliance regulations and consumer concerns [9]. This has caused organizations to evolve their thinking with respect to encryption key control and data residency, as shown in a recent study [10]. Surveying 5,009 IT professionals in 11 countries shows that 76% of financial service organizations are most likely to control encryption keys within their organization rather than at a third-party provider. 2) *Limited identification*: Mitigation per-IP or per-connection (i.e., two-level identification) is a conventional best practice [11]. Yet, this limits the ability to detect complex HTTP(S)-DDoS. For example, mitigation per-IP may result in errors in case of a multi-behavior per-

Manuscript received September 8, 2016.
Manuscript revised February 7, 2017.
Final manuscript received February 7, 2017.

[†]The author is with Department of Electrical Engineering and Information Systems, The University of Tokyo, Bunkyo-ku, Tokyo, 113-8656, Japan.
a) E-mail: eid.msa@acm.org
b) E-mail: aida@ee.t.u-tokyo.ac.jp

DOI: 10.1587/transinf.E0.D.1

Copyright © 2017 The Institute of Electronics, Information and Communication Engineers

shared-IP traffic, and in punishing a group of clients as one client (i.e., collateral damage), even after the attack stops. On the other hand, mitigation per-connection is not effective in case of a single request per connection attack.

To promote trust in the system, we define true-end-to-end encryption of client-server transactions (or non-split SSL) as a main requirement for an overlay-based DDoS-mitigation service. So, the proposed system's goal is to enable effective mitigation of complex HTTP(S)-DDoS attacks far from the server, while complying with the encryption requirement. Section 3.2.4 explains in more detail the metrics we use to define effective mitigation, i.e., mitigation factor, mitigation cost, mitigation time and collateral damage. The concept's core is to add a third-level of client identification to overlay-based mitigation, allowing an overlay-node to group related connections per client. We call this enhanced identification.

An initial version of our approach, which focused on low-level DDoS, was presented in [12]. Distributed special purpose Access Nodes (AN) are implemented, through which alone the client-server communication takes place. In addition to special purpose Public Servers (PS) which act as initial preparation points.

In this paper, with a focus on HTTP(S)-DDoS, a practical third level of identification is utilized enabling a novel behavior-based reputation and penalty system. Then, a proof of concept prototype with simplified mitigation measures and parameters is implemented and deployed on DeterLab [13]. Several experiments are conducted to evaluate the soundness of the concept. Among them, the results of two-long run experiments with a single-vector and a multi-vector complex HTTP(S)-DDoS attack scenarios are presented. Popular attack tools (i.e., LOIC and Slowloris) for HTTP-DDoS and a custom built one for HTTPS-DDoS were used for evaluation.

Results of testing with complex HTTP(S)-DDoS attack categories suggest that utilizing practical enhanced client identification can enable a high mitigation factor as 99.2% in short mitigation time of as 2 minutes. Yet, mitigation factors drop, and a cost of longer time to load the requested resource file (i.e. service time) temporarily during mitigation time. That's far from the server, without assuming knowledge of the client-server content. In addition, results show low collateral damage in terms of the chance of getting service and service time for non-attacking clients that share an attack IP in contrast to conventional overlay-based methods. Only the added third identification level was utilized for mitigation, while work in progress include incorporating all three levels to mitigate switching and multi-request per connection attack types.

The rest of this paper is organized as follows. Section 2, surveys related work on DDoS mitigation. Section 3, presents our proposed concept and prototype implementation. In Sect. 4 and 5, the evaluation results are explained and discussed. Finally, Sect. 6 concludes the work.

2. Related Works

Existing commercial overlay-based solutions can mitigate HTTP(S)-DDoS attacks [6–8, 11]. In addition, a plethora of overlay-based approaches to detect such attacks are proposed in literature. For example, employing software defined networking (SDN) [14], dynamically instantiated replica servers [15], human verification challenges [16], and content caching [17]. However, as discussed in Sect. 1, they either suffer from traffic decryption, limited identification, or both.

A hybrid solution is proposed in [18]. It employs SDN for a participating server to request from its ISP to block identified undesired traffic. This way the server benefits from the ISP's relatively large resources. So, integration with a locally-based detection method is necessary. Like the work proposed in [19] which provides a framework for classification of HTTP requests given parameters from the multiple layers of the protocol stack. However, the added load from reactive locally-based detection alone may overwhelm the server's resources, thus resulting in service denial. As much as possible, HTTP(S)-DDoS should be detected far from the server.

Aiming at enhancing client identification, the methods in [20, 21] utilize packet header fields as marking fields to distinguish between different clients per-IP. However, assuming modification of core routers hinders deployment. Also, there's no described method for how different client-related connections can be identified for marking by the third-party without content inspection.

Additionally, several methods adopted statistical and machine learning techniques to profile the traffic. For example, the work in [22] adopt Adaptive Selective Verification (ASV) to detect slow-requesting HTTP-DDoS. Similarly, the work in [23] proposed using statistical analysis to build connection scores based on several traffic attributes, including the browsing behavior (e.g., requested page type and popularity). However, similar methods rely on content inspection to detect HTTP-DDoS.

Further, entropy-based approaches can detect DDoS attack connections, such as the work in [24]. However, among the attributes utilized for detection, content inspection is assumed. Similarly, the work in [25] utilizes entropy, but instead of content inspection, detection is based on analyzing packet sizes to separate between attack and normal flows. Yet their definition of a flow is limited, based only per-IP and per-connection.

The work in [26] detect DDoS attacks using an ensemble of adaptive and hybrid neuro-fuzzy systems. Multiple features from TCP flows are extracted and utilized in detection, assuming no payload inspection, showing a high detection rate. However, a source is only identified per-IP or per-connection.

In addition, several mitigation approaches utilize game-theoretic techniques [27–30]. 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.

In the next section, we describe our novel concept which, to the best of our knowledge, is the first to enable enhanced client identification far from the server while complying with the encryption requirement.

3. Proposed Method and Implementation

In conventional overlay-based methods, the goal of mitigating complex HTTP(S)-DDoS is achievable by ignoring the encryption requirement and inspecting the content at overlay-nodes. We propose a new practical overlay-based method that aims at resolving the contradiction between this goal and requirement.

3.1 Proposed System Overview

Our method utilizes a third level of client identification, in addition to the conventional per-IP and per-connection, which we call per-session identification. A protected web server is locally-based (managed by its organization), completely hidden from direct low-level access, and is henceforth called secret server (SS). Conventionally, for a client (C) to receive a web service, one or more connections are established to the server through which actual data communication takes place. In the proposed method, this is called the communication stage. Per-session identification is realized far from the SS by preceding the communication stage with a preparation stage, which is handled by public servers (PS), while access nodes (AN) are in charge of relaying the C-SS actual true-end-to-end-encrypted transactions. More details of both stages and per-session identification are described in the following section.

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.

3.1.1 Preparation and Communication Stages

Refer to Fig. 1 for an example situation where two clients try to communicate with two different web servers. In this example, the PSs are illustrated separately, the same AN is shared, the clients have the same IP (IP_C), and they connect simultaneously. These conditions are just for illustration, while alternative conditions can be easily understood from this example. At first (step 1), each client normally enquirers about the IP address of the desired web server (X or Y). The one or more of the shared PS IP addresses are returned.

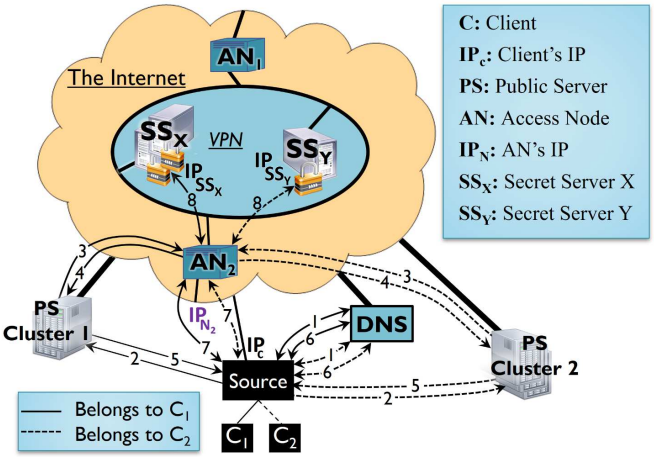

Fig. 1 Architecture of the original proposed framework [12].

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 connects to one of the suitable ANs (step 3) for a new access permission (AP). So, clients have no choice over which AN to use.

To enable per-session identification far from the server without content modification or inspection by the AN, the idea is to separate clients by destination port number. That way, the AN distinguishes between clients based on a unique locally generated session identifier (SID). SID is described by three indicators; IP_C, SS, and a temporary port assigned by the AN (i.e., IP_C:SS:PortN). In this example, the two SIDs are assigned different ports because IP_C is shared. Otherwise, the same port is reusable. In turn, the AN responds to each PS access permission request (APR) with the new SID descriptor (step 4). The AN also sends the client's reputation (described in Sect. 3.2.2) if available. The PS's role towards the requesting client ends in step 5 by sending a HTTP redirection response with the new location of the service (e.g., HTTPS://AN_2.SS_Y.com:PortN/), at the selected AN's assigned port for each client. That marks the end of the preparation stage.

A successful client must make use of the leased SID in a timely fashion. 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 establishing a connection to the designated AN port and passing 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.

So, the extra preparation phase and the role separation of ANs and PSs have multiple merits. First is enabling per-session identification using practical existing protocol features. Per-session identification is crucial for identifying single request per connection attacks and multi-behavior per-shared-IP, since each client's related connections are tied to a specific identifier. This leads to the second merit

which is eliminating the necessity for the AN to read or modify the data in transit for the sake of HTTP(S)-DDoS mitigation (and the ability is also eliminated by having the SS strictly locally manage its wildcard SSL certificate and private encryption key). Section 5.4 discusses the encryption aspect further. In addition, the simple and separate role of PSs enables handling thousands of concurrent requests per PS, accommodating multiple C-SS pairs, and simple replication. It also helps prevent all mitigation-unaware (blind) attacks from affecting the AN (i.e., communication phase), or the SS, effectively mitigating such attacks away from communication paths.

3.2 Mitigation

3.2.1 Detection Concept

The main detection components are distributed over the PS and AN, where each can monitor different behavior attributes. In addition, a supplementary detection component is installed at the SS for the possibility of an undetected attack that identically mimics normal behavior on the system's three identification levels and therefore require direct SS-AN feedback. In this paper, we present only the implemented AN component, and omit the implemented PS component, for space consideration. The supplementary component though is part of our future work, but we discuss it in sections 5.4 and 5.3.

At the AN, each client has three identifiers mapped by the AN to behavior attributes, and one broad top identifier utilized for system-wide attributes of a source address. Certain attributes are per time steps (per α [sec]). This is necessary to monitor behavior of low rate attacks, where attack traffic per source is too small to be measured per 1 second. Also, updating the reputation system per α, instead of per 1 second, helps reduce the AN's load. The system also keeps a time record of behavior attributes values during the past β time steps, which enables observing sub-detection-thresholds attacks. Next, we explain the identifiers and attributes with the aid of Table 1.

The broadest identifier is the origin identifier (OID) which is the source IP address. An OID *category* attribute is used in case of a pre-identified source for exclusion from detection, or for differentiating in detection mode between proxied and direct sources, while R_G is shared locally between different distributed ANs and PSs, and is used for marking suspicious sources for heightened detection (i.e., OIDs with bad R_G are assigned high penalty faster upon misbehavior). In the current prototype implementation, both of the OID attributes are not yet utilized.

Each OID is subdivided into several client-server pair identifiers (PID). The value of S_i is used in the cleanup of inactive PIDs (if $S_i = 0$ for β consecutive time steps). S_i also helps monitor PIDs that over consume the AN's local ports. Section 5.1 discusses port depletion. Whereas $Warn_{SS}$ is incremented once a warning is triggered by the SS's supplementary detection component. A PID's value of R_L is locally computed by the AN as a function of R_G, $Warn_{SS}$, and each pair-level misbehavior (see Sect. 3.2.2). It acts as a multiplier for the AN's sensitivity to misbehavior (i.e., for heightened detection). A failed session (FS) is one that observes a misbehavior for m consecutive time steps (i.e., $m \times \alpha$ [sec], where $m \leq \beta$), or that triggers a high-level warning from the SS. The AN keeps a history of past S/α values to monitor a PID's pattern of SID creation, and same for FS/α. Whereas values of S/P and FS/P allow the AN to instantly monitor the overall percentage of failed sessions for a specific PID.

PIDs are further subdivided by *PortN* into different session identifiers (SID). The SID's attributes enable a unique third identification level (i.e., per-session identification) far from the server while complying with true-end-to-end client-server encryption. For example, per-session behavior of a client that slowly creates a single connection, and disconnects after sending a single request, then repeats, would have a small C_i, but a relatively high C/α records for a single client and possibly with pattern (depending on its rate), and increasing C/S. In contrast with the limited two-level identification, where only the number of connections per IP would be observed, which may be high or low, depending on the unknown number of clients. P_i is a local parameter computed by the AN as a function of R_L and session-level misbehavior (Sect. 3.2.2). The value of T_S is used for local entry cleanup, and to detect a suspicious SID as explained in Sect. 3.2.3.

Finally, a connection identifier (CID) is basically the SID subdivided by source port. For each individual connection, a CID is mapped to M/α and M/C, counting all messages transferred from the client to SS, during the current step of α and throughout the connection, respectively. Analyzing the values of M/α records and M/C, in context with the SID's attributes, can enable the detection of complex attacks. For example, if an SID knowingly keeps the values of C_i and C/α within normal, further analyzing its underlying CIDs to observe patterns in the M/α records and M/C can indicate the likelihood of suspicious behavior. It's important

Table 1 Detection attributes at the AN.

ID	Attributes	Description
OID	Category	Reserved for tagging proxied or special-listed sources
	R_G	Global (system-wide) reputation of the source IP
PID	S_i	Current number of open sessions under this PID
	$Warn_{SS}$	Number of warnings received from the SS for underlying SIDs
	R_L	Local reputation calculated and stored by the AN for this PID
	S/α	Number of sessions created during the current time step α
	S/P	Overall number of sessions created under this PID
	FS/α	Number of failed sessions during α
	FS/P	Overall number of failed sessions under this PID
SID	C_i	Current number of open connections under this SID
	T_S	Session starting time
	P_i	Current penalty level for this SID
	C/α	Number of connections attempted/created during α
	C/S	Overall number of connections attempted/created for this SID
	FC/α	Number of failed connections during α
	FC/S	Overall number of failed connections under this SID
CID	M/α	Number of client's messages over this connection during α
	M/C	Overall number of messages sent over this connection
	T_C	Connection starting time
	Probes	Number of AN probes sent over this connection

to note that a misbehavior on a single CID affects the SID's all other related connections, and its future connections too. For each CID, the starting time and the number of verification probes sent are also stored for control.

3.2.2 Reputation and Penalty

A deviation of attributes from the expected client behavior per time step is henceforth called an "exception". We define two groups of exceptions. The first are pair-level exceptions which are based on per-PID attributes and are recorded in the vector $\hat{E}_P$. The second group are the session-level exceptions which are based on per-connection and per-session attributes and are recorded in the vector $\hat{E}_S$. $\hat{E}_P$ and $\hat{E}_S$, each is of length β bits, are shifted-in per time step by one position with the input as '0' indicating an exceptions-free time step or '1' indicating otherwise. So, for example, an all 1's $\hat{E}_P$ (i.e., $\hat{E}_P = max$) indicates the detection of at least one pair-level exception in all of the recorded past β time steps. The vectors are updated only in case of a new exception incident is triggered, while if none occur for a whole $\beta \times \alpha$ [sec], then the vectors are reset. These vectors are fed into the penalty and reputation update functions, which in turn determine the level of response by the AN.

Table 2 summarizes the exceptions considered so far in the system's concept. Five rate-related ones ($E_{1.1} \sim E_{1.5}$), plus five more for protocol, timing, size and SS warnings ($E_2 \sim E_6$). In the current prototype, only $E_{1.3}$, $E_{1.4}$, E_4 and E_5 are utilized, while future work includes expanding the prototype to utilize all exception records.

Next, with the aid of Fig. 2, we explain the update of R_L states, which is a function of $\hat{E}_P$ only, as R_G and $Warn_{SS}$ are normalized in the current prototype implementation.

In the current implementation, R_L has three possible levels; normal, suspicious, and bad. Each PID enjoys a normal R_L at first and as long as all PID attributes are below thresholds and $\hat{E}_P = 0$. Thresholds are tunable for each PID separately. In the incident of an exception $E_{1.1}$, $E_{1.2}$, E_5 or E_6 is triggered, separately or combined, the incident is recorded in $\hat{E}_P$, and R_L enters a suspicious state. As long as $0 < \hat{E}_P < max$, R_L's state remains unchanged. If $\hat{E}_P = max$ then R_L enters the bad state. Even in the case of a bad R_L, a PID is still granted service. So basically $\hat{E}_P$ controls R_L's transition up and down. Finally the value of $R_L = max\{R_G, R_L\}$, in case R_G is enforced. In the current prototype, the discussed three levels of R_L are represented numerically as below. Next, we explain how these values are used to affect the penalty update.

$$R_L = \begin{cases} 1 & , \quad if\ \hat{E}_P = 0 \\ 2 & , \quad if\ 0 < \hat{E}_P < max \\ 3 & , \quad if\ \hat{E}_P = max \end{cases}$$

The penalty (P_i) is enforced per SID and is a function of R_L and $\hat{E}_S$. The state diagram for P_i is omitted from this paper due to space consideration. By design, P_i has two states; normal initially ($P_i = P_{min}$), and differential ($P_{min} < P_i \leqslant P_{max}$). Normal state remains if $\hat{E}_S$ is all zeros (i.e., no session-level exceptions reported for at least $\beta \times \alpha$ [sec]). In any case of a session exception of the types $E_{1.3}$, $E_{1.4}$, $E_{1.5}$, E_3 or E_4, then $\hat{E}_S$ is updated and the service enters into the differential state. During this state the value of P_i ranges from $[P_{min} + 1, P_{max}]$ and the level of response (Sect. 3.2.3) is directly proportional to the value of P_i. Note that an SID can enter the differential state with $P_i = P_{max}$ without being a failed session yet (described in Sect. 3.2.1), since a bad $\mathrm{R_L}$ leads to $P_i = P_{max}$ even without a persistent record of exceptions or a high-level warning from the SS (which are the conditions for a FS). In case of a FS, a SID is hard limited and its PID's $\mathrm{R_L}$ is set to bad.

Table 2 Considered AN exceptions.

AN Exception	Vector	Event description
$E_{1.1}$: Session rate	$\hat{E}_P$	High rate of new sessions (S/P, S/α)
$E_{1.2}$: Failed session rate	$\hat{E}_P$	High rate of session failure by AN (FS/P, FS/α)
$E_{1.3}$: Connection rate	$\hat{E}_S$	High rate of new connections (C/S, C/α)
$E_{1.4}$: Failed conn. rate	$\hat{E}_S$	High rate of connection failure (FC/S, FC/α)
$E_{1.5}$: Message rate	$\hat{E}_S$	High rate of new messages (M/C, M/α)
E_2: High level	$\hat{E}_S$	Warning message from SS
E_3: Size exception	$\hat{E}_S$	Unexpected request size
E_4: Conn. timing	$\hat{E}_S$	Client doesn't request or doesn't wait for response
E_5: Session timing	$\hat{E}_P$	Client didn't show up at AN (after PS)
E_6: False session	$\hat{E}_P$	Connection attempt to a wrong SID

To utilize $\mathrm{R_L}$ as an exponential multiplier for sensitivity to misbehavior, in the prototype implementation we use the formula for $P(\hat{E}_S, \mathrm{R_L}) = \gamma + R_L^2$, where γ is the decimal value of $\hat{E}_S$. So, $\gamma = \sum_{j=0}^{\beta-1}(E_{S_j} \times 2^j)$. For simplicity, only $\hat{E}_S$'s 3 most significant bits were utilized. In a real implementation, however, $\hat{E}_S$ should be utilized fully. This gives a P_i ranging from $[1, 8]$ for $\mathrm{R_L} = 1$. For a unified range, P_{max} is also set to 8 for $\mathrm{R_L} > 1$. A lookup table is utilized to replace repetitive computation of P_i, which is fitted with the resulting values and indexed by γ and $\mathrm{R_L}$.

3.2.3 Attack Countermeasures

Attack countermeasures can be either proactive or reactive. In either case, attack traffic prevention or reduction is part of the AN and PS, far from the SS. In the current prototype, two attack countermeasures are in place.

Firstly, the client is probed with early fake single-byte slow response packets before proceeding with actual service. At least 1 probe is sent, to proactively prevent part of the blind attacks that doesn't wait for the SS's reply as a normal client would do. Conforming with existing standards, we utilize the constant bytes at the beginning of any HTTP (i.e., 'H', 'T', 'T', 'P', and '/') or HTTPS (i.e., 22_{10} and 3_{10}) first response. Since the system aims at HTTPS more, so we set the maximum number of probes to 2 for both protocols.

For any CID, the level of its SID's P_i determine the number and timing of sent probes. In the current proof of concept prototype, a lookup table is used for the duration before (t_1), between (t_2) and after the probes (t_3) is shown in Table 3. The table is manually populated based on ex-

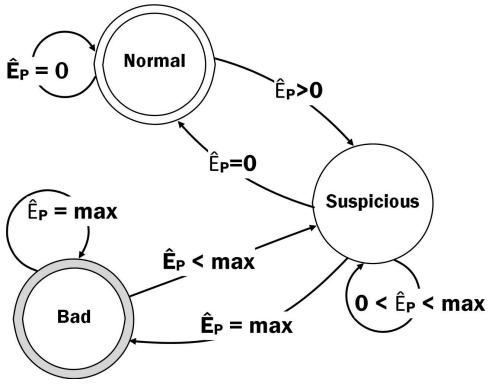

Fig. 2 Local PID reputation ($R_L(\hat{E}_P)$) states.

perience to exponentially throttle down attacking SIDs. It also helps prevent blind attacks that don't wait long enough for the probing phase to finish. As part of our future work, we plan to implement a separate queue for each level of P_i, so that for each penalty level is served with a lower priority than the one below. Therefore, the actual throttling automatically increases in case of a high demand by low penalty SIDs.

Secondly, the AN analyzes the current and past values of C/α and FC/α. If the analyzed values show a suspicious pattern, the related SID is placed in "jail" for a single time step while observing its latest C/α and FC/α. During jail time, all the jailed SID's new connection attempts are accepted, then immediately closed while counted as one attempt by the AN. In addition, R_L is exceptionally changed to bad for related PID with a reset timer of $\beta \times \alpha$ [sec]. Studying the behavior of popular attack tools commonly used by attackers (i.e., Slowloris and LOIC) suggest the effectiveness of this method as automated tools are trapped in jail by re-attempting at a nearly constant rate. So, to demonstrate the concept, a simple analysis algorithm is employed. An SID is sent to jail if $\frac{C/S}{w_i - T_s} \approx C/\alpha_i$, $\sum_{j=1}^{m} \frac{C/\alpha_{i-j}}{m} \approx C/\alpha_i$, or $\sum_{j=1}^{m} \frac{FC/\alpha_{i-j}}{m} \approx FC/\alpha_i$, where α_i is the current time step and m is the length of considered past behavior record. In the latest prototype implementation, m is arbitrarily set to 4 to demonstrate the concept.

3.2.4 Mitigation Metrics

We describe the four metrics considered in this paper to measure the effectiveness of mitigation, namely; mitigation factor (MF), mitigation cost (MC), mitigation time (MT) and collateral damage (CD).

MT can also be called the mitigation phase, which is defined as the interval within $[t_{att_{start}}, t_{p_{max}}]$ (i.e., MT = $t_{p_{max}}$ - $t_{att_{start}}$), where $t_{p_{max}}$ is when attack SIDs reach max penalty, and $t_{att_{start}}$ is the attack's actual starting time. Yet, the value of MT alone can't describe the mitigation effectiveness.

We define MC 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.

Table 3 Function of P_i; the lookup table used for the duration before, between and after the probes.

Penalty	# probes	t_1 [ms]	t_2 [ms]	t_3 [ms]
1	1	0	500	-
2	1	500	500	-
3	1	500	1000	-
4	2	500	1000	500
5	2	500	1000	1000
6	2	1000	1500	1500
7	2	2000	3000	2000
8	2	3000	5000	3000

The mitigation factor (MF_i) is defined as the attack traffic reduction factor (RF_i) multiplied by the chance of service completion (CoS_i) under attack for each time step i, where:

$$RF_i = 1 - \frac{\widehat{Att_i}}{Att_{\text{pub}}} \quad , \quad CoS_i = \frac{Resp_i}{Req_i}$$

Att_{pub} is the average achievable high-level attack traffic volume (i.e., requests per second) on the C-AN side before mitigation. While, for each time step i, $\widehat{Att_i}$ is the high-level attack traffic volume on the AN-SS side, Req_i is the number of non-attack requests sent, and $Resp_i$ is the corresponding number of service responses completed.

Finally, for each time step i, the collateral damage is defined as $CD_i = 1 - CoS_i$. Section 5.2 discusses these metrics further.

In Sect. 4, we present two of the several experiments we conducted so far to evaluate the soundness of the proposed concept by subjecting the current prototype version to complex HTTP(S)-DDoS attack strategies, some of which has not been considered before in related research on DDoS mitigation.

4. Evaluation

Enhanced identification, offered by the proposed concept, aims to enable effective mitigation (i.e., high MF, low MC, low MT, and low CD) of complex HTTP(S)-DDoS attacks far from the server, while complying with the encryption requirement. To evaluate this, the simplified proof of concept prototype is tested under several attack conditions within DeterLab [13] which permits root access to its nodes and testing with disruptive traffic. Among the conducted experiments, two experiments are reported in this paper representing a single-vector and a multi-vector HTTP(S)-DDoS attack.

Three types of parameters define the experiments' setup; prototype implementation, testbed specifications, and attack parameters. For the current prototype, we test with non-optimally set values. The value of β is set to 8 in the

experiments as it's neither too large nor too small. Whereas the threshold on the number of new connections per SID per time step ($C/\alpha\big|_{Thd}$) is set to 20, assuming a small α and that browsers normally set a small limit on simultaneous connections per server [31]. Other thresholds are not enforced, and R_G is normalized. We discuss the prototype parameters in Sect. 5.2.

4.1 Experiment 1: Single Vector Attack

A too large value of $m \times \alpha$ is undesired from a MT perspective, but also a large enough α is needed to detect low rate attacks. In this experiment, we set the value of α to 40 [sec] from experience. All nodes are automatically assigned by DeterLab from the bpc3000, pc3000, and pc3060 physical node types [32]. A single SS, AN, and PS are used, each on a dedicated pc3000 machine. Additionally, 100 measurement sources, each with a unique source IP, are emulated on a pc3000 based measurement node (MN), generating 1 HTTP GET request per IP per 30 seconds (for a 200 KB file). 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 node's OS is Linux V2.6.32, while the SS is running the open-source Apache server unmodified (version 2.2.14).

For attack, a total of 20 attack sources (emulated on 13 pc3000 and 7 pc3060 machines) first manually pass the PS's preparation stage, then target the SS through the AN using the LOIC blind attack tool (version 1.1.1.25), which is popular among attackers [33, 34]. Each one of the 20 attack sources acquires a single SID, runs 20 concurrent threads, and repeatedly attempts to send a single request per connection, repeated per thread with a random delay before connection termination (matching: single request per connection attack category).

Before hiding the SS, two measurements are conducted on it directly (i.e., without mitigation). First, the server's local limit of requests per second is measured, which can be a function of multiple factors, including: the requested resource, server's configuration, and hardware. For the requested 200 KB file, the non-attacked server showed a local limit of 54.6 [r/s] (i.e., requests per second) in average and 19.6 [ms] service time when tested with ApacheBench (version 2.3).

Additionally, the attack is directed against the SS directly. The generated attack's HTTP request rate measured a total of 47.27 [r/s] in average. As expected, the attack rate is close to the server's local limit but don't exceed it. Generally, once an attack traffic approaches the server's local limit, deterioration in CoS and ST is observed. This direct attack resulted in a decrease in CoS to 66.5% and rise in ST to 8477 [ms] in average with 95% CI. We could increase this limit by tuning the server's configuration, yet, our focus is on evaluating mitigation of attack far from the server. So, that limit is unchanged in all experiments.

4.1.1 Start of Attack

Now the mitigation system is in place. As shown in Fig. 3, attack traffic starts at point i. Between points i (t = 5.3 [min]) and j (t = 7.3 [min]) is the mitigation phase of the AN, during which the peak achievable attack rate on the public side reaches 6149 [r/s], while Att_{pub} is approximately 4935 [r/s]. The AN's effect is seen in terms of CoS and the reduction in attack traffic reaching to the SS. For every time step i, the RF_i is observed above 99.2% during the mitigation phase and afterwards. Before point i, RF is undefined. The AN also shows a 100% CoS_i with a temporary drop to 99% (i.e., MF_i above 98.5%), far from the server, with zero knowledge of the requested high-level content. Yet, the cost is observed as increase in ST_i for the requested file, with temporary spikes during the initial 2 minutes mitigation phase.

Notice also the cost of the current iteration of the prototype, even before attack, seen in ST (nearly 1 [sec]) for the 200 KB file. An optimized implementation is expected to show a smaller value.

4.1.2 Switching Attack

The experiment lasts for 24 hours, which is considered a long duration attack [3]. Near the end, the HTTP-DDoS traffic is switched OFF and ON repeatedly as shown in Fig 4 to observe the effect on mitigation. Switching intervals shorter than α (points k and l) show no effect on CoS_i and RF_i. On the other hand, longer switching intervals (points m and n) caused temporary drops in the CoS_i. Yet, mitigation time is relatively shortened, in comparison to Fig. 3, while the AN's effect is nearly the same after point o and after point j.

The high reduction in attack traffic on the SS side (Fig. 4's top) suggests that the observed temporary decrease in CoS_i, and ST_i fluctuations, are mainly due to the ON-OFF tactic's impact on the prototype AN implementation. Not on the SS itself. So, only the users being served through the attacked AN may notice the short temporary drop during MT and subsequent fluctuation, while everyone else can still enjoy normal service through other ANs.

Notice that for LOIC, the AN's response reduces the ratio of attack traffic SS-side to C-side nearly to zero, but the volume of connections per second remain unaffected. Such observation is utilized to improve the detection in the following experiment (i.e., utilizing the second countermeasure described in Sect. 3.2.3).

In this experiment, the attack is not knowledge-based, unwarily exceeding the detection threshold. In the next experiment, we test with mixed complex attack categories, and also attempt to evaluate collateral damage.

4.2 Experiment 2: Multi-Vector Attack

This time, an updated version of the AN prototype is used.

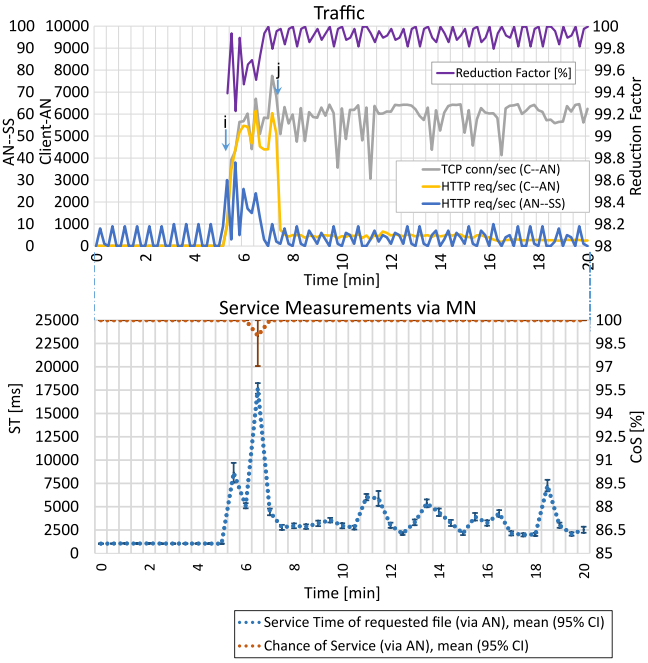

Fig. 3 Experiment 1: Start of attack.

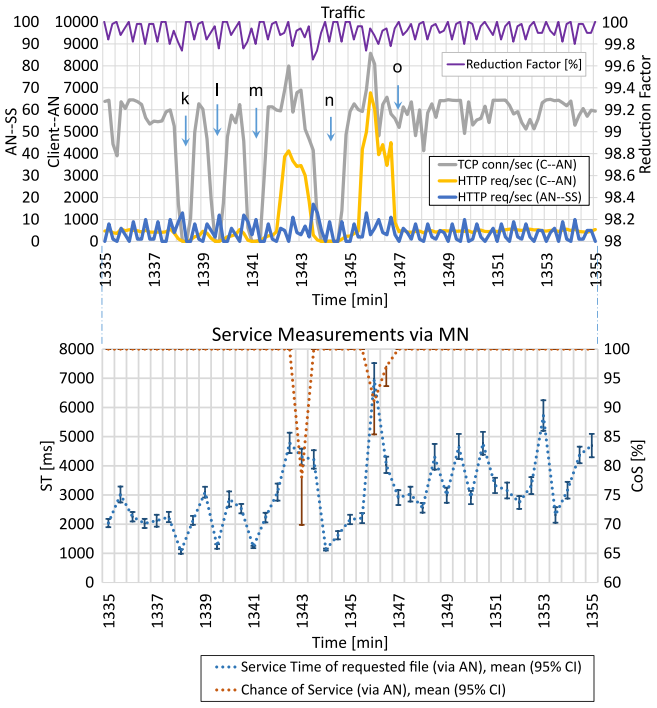

Fig. 4 Experiment 1: Fast repeated OFF-ON attack strategy.

It includes the second countermeasure described in Sect. 3.2.3, and a reduced $P(0, 3)$ (in the P_i lookup table described at the end of Sect. 3.2.2) from 8 to 6, for a less aggressive initial response against new SIDs.

In this experiment the SS is targeted via the AN with a double-vector HTTP(S)-DDoS attack; combining the high-rate slow-requesting HTTP-DDoS and sub-threshold HTTPS-DDoS strategies (matching: single request per connection, multi-behavior per-shared-IP, sub-detection-thresholds, and multi-vector attack categories). The goal is to observe the AN's effect on such traffic far from the server and evaluate the MF, MC, MT, and CD. We test with a lower $\alpha = 30$ [sec].

Slow-requesting HTTP-DDoS is an attack that consumes the limit on connections to the server by keeping it waiting for the rest of the slowly arriving request. For that, we utilize Slowloris which is commonly used by attackers [33, 34]. Originally, the tool only blindly focuses on the PS without reaching any further. However, in the worst case it can be adapted to reach to the AN, which we consider here. Separation between Slowloris requests is set to 1 second. The resulting rate per source is high, because the Slowloris tool aggressively attempts to open multiple concurrent connections. In addition, to emulate defense-aware attack behavior, assuming attacker's knowledge of the preset threshold, we utilize a sub-threshold HTTPS-DDoS which in this case outputs a single valid request per connection, via a renewed single connection per SID, repeated nearly every 7 seconds (i.e., rate per source IP = 9 [conn/min], with 1 [req/conn]).

The utilized node types bvx2200, bpc2133, and pc2133 were automatically selected by DeterLab [32]. A single SS, AN, and PS are used in this experiment, each on a dedicated bvx2200 machine. For attack, 1,000 HTTPS-DDoS custom attack sources are emulated on 10 machines (5 bvx2200 and 5 bpc2133), in addition to 10 Slowloris sources emulated on 10 additional machines (8 pc2133 and 2 bpc3000). In addition, 100 measurement sources are emulated on the MN (bvx2200), which are modified to use HTTPS, instead of HTTP. To observe the AN's collateral damage on non-attacking users sharing an attack IP, 2 additional measurement sources are run, one measurement source sharing the IP address of a Slowloris attack source (labeled M-shared-1) and another sharing that of a HTTPS-DDoS attacking source (labeled M-shared-2). All nodes' OS is Linux V3.13, while the SS is running an unmodified Apache server (version 2.4.7).

Before hiding the SS, it showed a measured local limit of 54.7 [r/s] (mean) when tested with ApacheBench version 2.3, with average service time per single connection of 18.2 [ms], for a requested file of 200 KB in size. Additionally, the SS showed 0.0% CoS under this direct multi-vector attack.

Then, the experiment is run for 120 [min], of which the first 40 [min] are shown in Fig. 5. The attack traffic starts at t = 8 [min] and continues for 13 minutes before the first controlled switching off/on operation.

In the first 1.5 minutes of the attack, a peak of 1,601 HTTP(S) [r/s] is observed, with average of 1424.6 [r/s]. During this interval, we observe a decrease in RF_i to 92.8%, with an unaffected CoS_i of 100% (i.e., $MF_i = RF_i$) for the MN, M-shared-1, and M-shared-2. Before t = 8 [min], RF is undefined. Inspecting the measurements of the MN, M-shared-1, and M-shared-2, we notice spikes in ST_i during

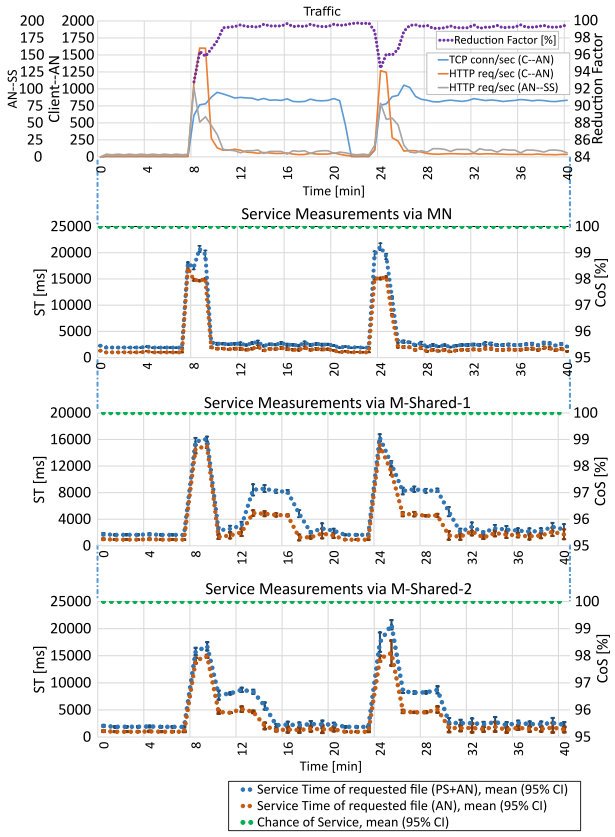

Fig. 5 Experiment 2: Multi-vector attack combining high-rate slow-requesting HTTP-DDoS and low-rate HTTPS-DDoS.

the mitigation phases, coinciding with the attack start and later restarts.

However, during MT, the value of ST_i rises to nearly 17 seconds for obtaining a complete service of the requested file via the AN. Afterwards, the effect of the AN can be seen in terms of around 99.2% RF_i, and back to pre-attack ST_i levels. This suggests that the jail concept (Sect. 3.2.3) and enhanced identification can be effective in remotely isolating automated attack categories, even without knowledge of the traffic content, even with sub-threshold attack rate and mixed behavior.

Later, at t= 23.5 [min], the coordinated attack was restarted after 2 minutes of planned inactivity. The attack tools restart their attack by abandoning their old SIDs and acquiring new SIDs then attacking though the new ones. The AN temporarily showed a 94.4% RF_i with no effect on CoS_i, and as earlier the ST_i deteriorated briefly. Throughout the rest of the 2 hours long attack, the prototype AN showed the same effect.

Further, measurements from the attack nodes' IPs (i.e., from M-shared-1, and M-shared-2) show an additional service time increase next to the spikes. This increase indicates a change in the attack IPs' reputations to the bad level (as explained in Sect. 3.2.3). The increase results in extra probing time only for these IPs, until the reputation automatically returns to normal after β time steps from the jail start. Also, ST_i via PS rises because the AN feeds back the PID's R_L to it. Non-attacking IPs (of the MN) didn't experience this extra service time. However, such short increase is only a few minutes long, until the reputation is later normalized, or until the attack IPs restart the attack.

As attacks evolve, it's anticipated that an even more sophisticated attack may adapt by constantly switching its traffic. We discuss this in Sect. 5.3.

5. Discussion

5.1 Scalability

As an overlay-based solution, scalability with legitimate traffic is achievable by distributing traffic over thousands of nodes, thus eliminating 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.

Whereas 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 [3], so it can't reach to the SS anyway in the proposed scheme. We discuss low-level attacks in Sect. 5.3. 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, attack traffic of 6,149 [r/s] and 1,601 [r/s] are utilized, respectively. This is a significant volume per overlay-node, considering the reported peak high-level attack rate of 268,800 [r/s] [3]. So, with our suboptimal proof of concept prototype implementation of the AN withstanding 6,149 HTTP(S) [r/s], then we argue that nearly 44 ANs can mitigate the reported peak.

Also, to avoid depleting the finite ports pool per AN, different OIDs reuse the same AN ports. So, for k OIDs, l ANs, and a pool of n ports per AN, then the number of SIDs that this system can accommodate is up to $l \times k \times n$ SIDs. For example, a system with 1,000 ANs and a pool of 50,000 ports per AN, the system can theoretically accommodate up to 50 million SID's per OID.

5.2 Evaluation

Evaluations are conducted on the DeterLab testbed within specified environments. So, there are numerous variable factors that may have an impact on the results. Factors include; the system's parameters, prototype implementation, nodes' specifications, network conditions, attack and legitimate behavior, and server configuration. However, our goal is to evaluate with specific metrics the effect of the enhanced identification enabled by the proposed concept on attacks that are conventionally hard to detect without data inspection. We emulated four such attack categories, namely; single request per connection, multi-behavior per-shared-IP, sub-detection-threshold, and multi-vector attacks.

Variations in strategies within such categories may be endless. Yet some strategies are more likely than others, due do multiple factors, including; attack tool popularity, attack sophistication, and victim vulnerability. The attack in the first experiment utilized a popular tool among attackers (i.e., LOIC), which generates a high rate single request per connection. In addition, the second experiment utilized another popular attack tool among attackers (i.e., Slowloris) [33,34], simultaneously with a custom tool that we developed to emulate multi-behavior per-shared-IP, sub-detection-threshold, and multi-vector HTTP(S)-DDoS attacks. Next section discusses other attack possibilities.

Also, among the experiments' factors are the arbitrarily set parameters of the implemented prototype, such as, α and $C/\alpha\big|_{Thd}$. For example, a small α can help reduce the mitigation time, yet, a large one is more suitable for slower and switching attacks. Also, a fixed $C/\alpha\big|_{Thd}$ may lead to false detection under different attack conditions, say, if attackers learn to control their traffic below it, while some non-attacking clients may occasionally exceed it. Therefore, the conducted experiments consider both above and sub-threshold attacks. However, real-time adaptation to attack variations would require automatic tuning of the system's parameters and learning about client usage record patterns for each server. For that, this work is extendable to incorporate existing machine learning based DDoS mitigation methods which originally only employ two-level identification (assuming overlay-based deployment and compliance with the encryption requirement).

In addition, the metrics in Sect. 3.2.4 which aim to describe the mitigation from different angles are to be used with caution. For example, the definition of mitigation time can describe the time taken to mitigate attacking traffic, however it assumes knowledge of $t_{att_{start}}$, which may not always be the case. The exact start time of an attack may be difficult to determine in case of a very low rate attack, which makes it difficult to calculate the mitigation time and similarly the value of Att_{pub}. In related works, the CoS is used as a metric in [22] and called "Client Success Ratio", while in [23] the used metric is equivalent to our defined RF whereas researchers in [19] consider service time in their metrics. Similarly, the metrics used in [29] are "bandwidth utilization by attacker" (equivalent to RF) and "ratio of lost legitimate users to the total number of legitimate users" (equivalent to CoS). Furthermore, the definition of collateral damage should also include the service time as a factor. As in experiment 2, the service time for the shared IPs appears similar to the non-attacking sources, except with a short increase in service time, which is a kind of collateral damage (with 100% CoS).

5.3 Attack Strategies

DDoS mitigation has been an endless battle where attack strategies evolve just as mitigation solutions do. There's a chance that an evolved HTTP(S)-DDoS attack sends multiple requests per connection, or multiple SIDs per PID (e.g., the switching behavior seen in Fig. 3∼5). For that, additional attributes must be utilized in detection, including M/α and S/α. The attacker's options are limited; if the attacker reduces M/α, C/α and C/S to evade detection, then intuitively the rate of S/α and S/P would rise. So, a PID with small C/α, small C/S, and small M/α, but large S/α would indicate the likelihood of suspicious behavior. Also, analyzing with a suitable machine learning algorithm the recorded values of S/α for the past β time steps can enable detecting a switching attack behavior with high mitigation factors in short mitigation times, far from the server, while complying with the encryption requirement. So, we argue that the AN's effect presented in Sect. 4 can be extended to the other attack types once their respective attributes are considered. Considering all the attributes in mitigation is a future work.

Furthermore, the mitigation system must consider the case where the attack traffic may not be detected initially by the AN. For example, a new attack strategy that appears identical to legitimate behavior may be hard to detect given the described levels of behavior attributes in Sect. 3.2.1 alone, even with automatic mitigation parameters tuning. Then, the undetected traffic would require a supplementary detection component.

One way to implement supplementary detection is to have the SS periodically transfer its logs to the mitigation provider (e.g., using Syslog messages). Then the mitigation provider analyzes the logged traffic features such as the request categories, sizes, errors, frequencies, and arrival times. However, the logs alone may still not be enough to detect a careful flash-crowd mimicking attack. Also, logs such as requests' categories vary from server to server, and so content-indifferent log analysis may be difficult to implement.

Instead, supplementary detection is better suited at the SS's premise, which has two functions: 1) detection of high-level behavior anomalies, 2) warning the mitigation system's components over the private network. For detection, simply the local SS can serve a human verification test to clients, where failed ones are reported to the mitigation system. However excessive human verification can be annoying for good clients. To reduce the need for such verification tests, previously proposed concepts for statistical anomaly detection can be integrated within our system to profile the features of normal behavior for each server based on analysis of request types and browsing patterns expected by the server. However, the locally-installed third-party supplementary detection component may raise trust concerns. In the next section, we discuss the trust perspective.

Upon receiving a warning, the PSs and ANs can respond in different ways. One way is to step up the countermeasures, for example by temporarily raising the penalties of the concerned clients. Another way to deal with hard to detect attacks, and surges in legitimate traffic, is by incorporating an existing approach that employs scheduling policies to prioritize how clients receive service. Such approaches originally assume traffic decryption, limited identification, or both. For example, the concept considered in [35] can be integrated in our system, benefiting from enhanced iden-

tification, so that the service of requests is scheduled based on reputation and penalty levels which are function of combined remotely-based behavior attributes and SS warnings.

In this paper, we focus on the increasingly popular HTTP(S)-DDoS attacks. Yet, we discuss the possibility of a low-level DDoS attack. The PS expects connections from almost any source at any time to its fixed port 80, so it may attract low-level attacks such as a TCP SYN flood. However, the specialized PSs expect only a single request per client, and can handle larger request rates than a conventional web server. In addition, it can be sacrificed to stop a multi-Gbps massive attack of such kind far from the communication stage. 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, where any incomplete attempts, or high rate of C/α, are easily detectable by the AN and updates R_G accordingly. Otherwise, low-level attack traffic can only hope to congest the links to the thousands of ANs which simply drop incorrect SID connection attempts.

Another model of DDoS attacks is called Economic Denial of Sustainability (EDoS). The concept is based on engaging a paid-for service, such as the DDoS mitigation service, to cost the SME economically. However, we assume the mitigation service to be always-on, and shared, therefore its economic sustainability can't be denied.

5.4 Encryption and Trust

In the proposed concept, the third-party-managed mitigation service is trustworthy as it cannot access the C-SS content in transit, which is guaranteed by non-split SSL connections between the C and SS ends. Notice that conventional Content Delivery Networks (CDN) may also offer non-split SSL as an option utilizing Server Name Indication (SNI), which we discuss its limitations in Sect. 5.5.

However, initially the client and PS exchange a single request and response, which are unencrypted. So, only that first transaction is readable by the mitigation provider and any man in the middle (MITM). But consider the goal of SSL encryption, which is about guaranteeing the integrity, authenticity and confidentiality of the client-server transactions. Since the cryptography information, including the server's private key, is strictly only managed locally by the SS, so the PS (or MITM) 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. Further, we argue that the information leakage from the unencrypted first request to the PS is not significant, since it's already known to the mitigation provider that C is intending to communicate with SS. Notice that in the SSL handshake, the server's certificate is not encrypted anyway, which can also leak the same information for any MITM.

In addition, the SS is locally-managed which means that its administrators already trust the server related software and hardware. However, the assumed additional local supplementary detection component may raise trust concerns as it can access unencrypted information. Trust is indeed a complicated problem. For instance, trust concerns have recently convinced the European Parliament to push for systematic replacement of existing proprietary software in all the EU institutions by open-source software, and adding 'open-source' as a mandatory selection criterion in public IT procurement [36]. To promote trust in our proposed method, the source code of the local component must be made publicly accessible for independent auditing and verification. This has two major merits. First, it helps prevent backdoors where SMEs can fully manage it and be independently assured that it operates as specified. Second, it opens the way for multiple proprietary remotely-based mitigation providers to compete and cooperate, using the open-source component and openly improving it. This way, the business of DDoS mitigation is no more dominated by major proprietary vendors. Thus, promoting affordable, practical, and trustworthy DDoS mitigation solutions for SMEs to choose from and easily switch between them.

One final point about encryption. Since we already trust major third-party certificate authorities (CA) to verify the authenticity of the certificate-owning server, would it be widely acceptable to expand this trust further to allow CAs to inspect traffic data for the sake of DDoS-mitigation? Recent studies about encryption and trust suggest otherwise [9, 10]. Also, making no new trust assumptions about the DDoS mitigation providers allows for multiple new innovative entities to enter the business of overlay-based DDoS mitigation and be widely accepted by users (SMEs and clients).

5.5 Conventional Solutions

Effective mitigation of HTTP(S)-DDoS attacks is achievable by utilizing existing Content Delivery Networks (CDN) which increase the content's availability. However, recent studies suggest that authorizing a third-party CDN to manage the server's private key may not be acceptable by large sector of organizations and web users [9, 10].

Instead, CDNs and overlay-based solutions in general, can also offer non-split SSL as an option utilizing Server Name Indication (SNI). However, CDN with SNI suffers from limited identification (i.e., only per-IP and per-connection). Without enhanced identification, if a source IP opens x connections per unit time for example, determining whether x is normal or not depends on the number of clients sharing that IP, which is unknown by the remotely-based CDN nodes. Same with persistent connections. Also, future attacking connections cannot be remotely profiled if only a single request per connection is sent. Further, if a source IP is blocked for slowly creating new attack connections, conventionally it's hard to exempt new non-attack connections either simultaneously or even promptly after the attack stops. So, although CDNs with SNI can be effective against massive low-level DDoS attacks, yet,

we argue that CDNs with SNI can't effectively mitigate complex HTTP(S)-DDoS attack types such as; single request per connection, multi-behavior per-shared-IP, and sub-detection-thresholds. This means that for CDNs with SNI and non-split SSL, more attack categories would need to be detected locally, and stopped inaccurately (only per-IP or per-connection). On the other hand, our proposed method aims at increasing the scope of attack categories that can be effectively mitigated far from the server, thus reducing the burden on the server's supplementary detection component.

Furthermore, in academia, we don't know of a proposed overlay-based method for mitigating HTTP(S)-DDoS that assumes non-split end-to-end-encrypted client-server connections. Utilizing enhanced identification, the implemented proof-of-concept prototype shows mitigation factors above 99.2%, yet, with temporary drop to nearly 80% during the 2 minutes mitigation time. We claim this as a high mitigation factor and low mitigation time, in contrast to related methods that inspect the content. For example, the simulations by [22] with 280 attacker sources, shows an 80% "Client Success Ratio" (i.e., CoS), while in [23], experimental results with 600 attacker sources show nearly 56% reduction in attack traffic (i.e., RF) in 2 minutes. On the other hand, commercial solutions such as [37] claim a 5 to 15 minutes "time to mitigate", which is not measured from the actual attack starting time but from the time the attack traffic is redirected to the mitigation centers after it has been detected (i.e., an additional detection time should be added). So, it's not equivalent to our defined MT, but rather longer.

As discussed in Sect. 2, DDoS mitigation approaches are many. Each has its strong and weak points. Table 4 summarizes the comparison between the proposed framework to conventional locally-based and overlay-based approaches from five perspectives. Source and infrastructure based approaches are excluded out of practicality.

Conventional overlay-based solutions can mitigate HTTP(S)-DDoS attacks but these assume either a split SSL connection between the client and server, or limited (per-IP and per-connection) identification, or both [6–8, 11]. On the other hand, a locally-based solution is intuitively in position that enables both non-split encryption requirement (since private key is only managed locally) and enhanced identification (e.g., utilizing HTTP cookies). In contrast, our method enables a content-indifferent enhanced identification, far from the server, with no need (or ability) to inspect or modify the client-server communications for DDoS mitigation. Scalability and expense wise, the proposed method inherits the general merits of overlay-based solutions which are superior to locally-based solutions. In short, our method is most suitable for trust-aware SMEs that strictly locally manage encryption parameters and their servers but need a remotely-based mitigation service for its high scalability and low cost.

6. Conclusion and Future Work

Protecting web servers against HTTP(S)-DDoS attacks should not contradict with the recent rise in awareness about traffic encryption due to trust concerns. This paper presents a new overlay-based DDoS mitigation concept that affirms this awareness. The implemented prototype, equipped with simplified detection measures and attack countermeasures to demonstrate the concept's soundness, is tested under several simple and complex HTTP(S)-DDoS conditions on the DeterLab testbed. Results of the conducted experiments suggest that utilizing practical enhanced client identification enables high mitigation factors of sub-threshold and single request per connection attack categories, in short mitigation time, far from the server, without assuming knowledge of the client-server content. In addition, non-attacking clients that share an attack IP experience low collateral damage by the proposed concept in contrast to conventional overlay-based methods. Future work includes utilizing the identification levels unconsidered yet in the prototype to mitigate switching and multi-request per connection attack categories. We also plan to experiment with local feedback between system components.

Table 4 Proposed vs conventional solutions.

	Proposed Method	Conventional Solutions	
		Locally-based	Overlay-based
Connection Encryption (Client-Server)	Full (end-to-end)	Full (end-to-end)	Split (decrypted)
Client Identification	Enhanced	Enhanced	Limited
Capital + Operating Expense	Low	High	Low
Scalability against Attack Traffic	Scalable	No	Scalable
Scalability with Legitimate Traffic	Shared responsibility	Server's responsibility	Shared responsibility

Acknowledgments

This work is supported in part by; The Japan Student Services Organization (JASSO), The Japanese Ministry of Education, Culture, Sports, Science and Technology (MEXT), The KDDI Foundation, and The Teijin Foundation. We also thank the DeterLab operations team at the University of South California for their testbed support.

References

[1] Kaspersky: DDoS attacks in Q4 2016, securelist.com/analysis/quarterly-malware-reports/77412/ddos-attacks-in-q4-2016/, Feb 2017.
[2] McAfee Labs: Threats Report - March 2016, www.mcafee.com/us/resources/reports/rp-quarterly-threats-mar-2016.pdf.
[3] Incapsula Report: 2015-2016 Annual DDoS Threat Landscape Report, www.incapsula.com/blog/2015-16-ddos-threat-landscape-report.html, Aug 2016.
[4] J. McCallion, "A10 storms the DDoS prevention market with Thunder TPS", www.itpro.co.uk/security/21393/a10-storms-the-ddos-prevention-market-with-thunder-tps, 2014. [Online; accessed 14-March-2016].
[5] Arbor: Worldwide Infrastructure Security Report - Vol. XI, www.arbornetworks.com/images/documents/WISR2016_EN_Web.pdf, 2016.
[6] CloudFlare, "SSL Options", support.cloudflare.com/hc/en-us/articles/200170416-What-do-the-SSL-options-mean-.

[Online; accessed 7-Feb-2017].
[7] Akamai, "Kona DDoS defender", www.akamai.com/us/en/multimedia/documents/product-brief/akamai-kona-site-defender-product-brief.pdf. [Online; accessed 7-Feb-2017].
[8] VeriSign, "DDoS protection services", www.verisign.com/internet-defense-network. [Online; accessed 7-Feb-2017].
[9] 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.
[10] Encryption Application: Trends Study - Thales e-Security, Inc., www.thales-esecurity.com/knowledge-base/analyst-reports/encryption-application-trends-study-2016, June 2016.
[11] AWS Best Practices for DDoS Resiliency, d0.awsstatic.com/whitepapers/Security/DDoS_White_Paper.pdf, June 2016.
[12] 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.
[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] 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.
[16] 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.
[17] 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.
[18] 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.
[19] A. Krizhanovsky, "Tempesta: A framework for HTTP DDoS attacks mitigation", Proc. 24th Annual International Conf. on Comp. Science and Software Eng., IBM Corp., pp.148-162, 2014.
[20] 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.
[21] 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.
[22] 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.
[23] 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.
[24] 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.
[25] 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.
[26] 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.
[27] 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.
[28] 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.
[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] 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.
[31] Hypertext Transfer Protocol – HTTP/1.1: Connections (RFC2616, Sec.8).
[32] DETERLab Node Types, http://docs.deterlab.net/core/node-types/. [Online; accessed 7-Feb-2017].
[33] Arbor: Worldwide Infrastructure Security Report - Vol. X, 2015.
[34] Radware: Common DDoS Attack Tools, https://security.radware.com/ddos-knowledge-center/ddos-attack-types/common-ddos-attack-tools/, Jan 2016. [Online; accessed 7-Feb-2017].
[35] S. Ranjan, R. Swaminathan, M. Uysal, A. Nucci, and E. Knightly, "DDoS-Shield: DDoS-Resilient Scheduling to Counter Application Layer Attacks", IEEE/ACM Transactions on Networking, vol.17, pp. 26-39, 2009.
[36] European Parliament resolution P8_TA(2015)0388 on the electronic mass surveillance of EU citizens, Oct 2015.
[37] LEVEL 3® DDoS MITIGATION, http://www.level3.com/-/media/files/brochures/en_secur_br_ddos_mitigation.pdf. [Online; accessed 7-Feb-2017].

Mohamad Samir A. Eid received the Bachelor's degree from Misr University for Science & Technology in 2005, and his Master's degree in Electrical Engineering from The University of Tokyo in 2010. He is now with the Dept. of Electrical Engineering & Information Systems, The University of Tokyo, Japan.

Hitoshi Aida was born in 1957. He graduated from The University of Tokyo in 1980, and received Doctor of Engineering in 1985. After joining the University of Tokyo, he stayed two years at SRI International in California as an international fellow from 1988 to 1990. He has been a professor of the University of Tokyo since 1999. His current research interests include computer networks, distributed processing, wireless network applications, and disaster resilient networks. He is a senior member of IEEE and a member of ACM, IEICE, IPSJ, JSSST, JSAI and IEIEJ.

View publication stats