博士論文

**Doctoral Dissertation**

# End-to-End Encryption Enabled Overlay-based Mitigation of HTTP and HTTPS DDoS Attacks: Design and Proof of Concept Implementation

(エンド・ツー・エンド暗号化に対応したオーバーレイに基づく
HTTP および HTTPS DDoS 攻撃緩和手法：
設計と概念実証実装)


by

Mohamad Samir AbdelRahman Eid

モハマド　サミル　アビデラヒマン　エイド

*(37-107400)*

Supervisor

Professor. Hitoshi Aida

Department of Electrical Engineering and Information Systems
The University of Tokyo


This dissertation is submitted for the degree of
Doctor of Philosophy

Graduate School of Engineering　　June 2016

To my family ...

# Declaration

I hereby declare that except where specific reference is made to the work of others, the contents of this dissertation are original and have not been submitted in whole or in part for consideration for any other degree or qualification in this, or any other university. This dissertation is my own work and contains nothing which is the outcome of work done in collaboration with others, except as specified in the text and Acknowledgments.

Mohamad Samir AbdelRahman Eid
June 2016

# Acknowledgments

And say, “My Lord, advance me in knowledge.” (Quran 20:114)

I would like to express my deepest gratitude to my supervisor, Professor Hitoshi Aida. It’s a precious chance to work under his supervision and gain from his endless support. He has always been a great source of inspiration and encouragement.

Many thanks also to my undergraduate teachers in Egypt, especially Professor Salah Yusuf. He has taught me a lot and strongly supported my idea of studying in Japan and especially in such a great institution as The University of Tokyo. May his soul rest in peace.

I also extend my thanks to those who shared every moment with me throughout the studying years. Cheering me when down and celebrating my achievements with love. My mother Soad. My father Samir, may he rest in peace. They sacrificed a lot just to see me successful and happy. My Sister and brothers Iman, Hesham, and Ahmad. My beloved and patient wife Yasmine, and of course my amazing kids Malak and Yusuf. I couldn’t have done it without you all beside me.

I like also to express my gratitude to Professor Kumiko Morimura for treating us so kindly and always being there for us like a true friend. Professor Yoshimasa A. Ono for opening his library for me and giving me valuable advice on technical writing. Professor Yashushi Wakahara for spending his precious time with me discussing my developing research ideas in the beginning. All the student support staff for their great help. Ms. Sayuri Nakayama and Ms. Misako Motooka who both have been so kind and helpful all the time. And all my lab mates, who have been nice to me, many thanks.

Special thanks also to the DeterLab team for their experiment support.

This material is based upon work financially 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. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author and do not necessarily reflect the views of the the mentioned entities.

# Abstract

To date, DDoS attacks against web servers remain among the most common types of cyber-attacks for their simple yet effective nature. Unlike volumetric attacks that utilize the transport or network layer protocols (low-level), high-level DDoS attacks based on HTTP are smaller in volume and harder to detect. Although mitigation at server's premise (locally-based) can be effective against certain attack categories, coping only locally with the increase in attack volumes and sophistication would require large spending that may only be affordable by large-sized enterprises. Conversely, SMEs require affordable, scalable, and practical remotely-based mitigation (i.e., third-party-managed overlay-based).

As overlay-based mitigation of low-level DDoS became well established, recently more frequent attacks are reported that are based on HTTP and its SSL encrypted version (HTTP(S)-DDoS) attacks. Current overlay-based mitigation systems can mitigate HTTP(S)-DDoS attacks, yet they either suffer from traffic decryption, limited identification, or both. Limited identification at the overlay-nodes necessitates traffic decryption to mitigate complex HTTP(S)-DDoS, while true end-to-end encryption limits the behavior identification attributes.

Practicality is a key for wide acceptance of the system by organizations and users. So, in order to effectively mitigate HTTP(S)-DDoS while complying with the encryption requirement, practical enhancement of overlay-based identification of clients is investigated in this dissertation. Focusing particularly on the complex versions of HTTP(S)-DDoS, namely: single-request per-connection, sub-detection-thresholds, slow-requesting HTTP-DDoS, and multi-vector attacks.

To enable the desired enhanced identification, firstly a new overlay-based system is designed which practically introduces a third level of client identification (per-session) in addition to the conventional two-level identification (per-IP and per-connection). Web servers are unmodified and managed locally by their administrators (called Secret Servers or SS), while the third-party-managed overlay-based mitigation system consists of distributed special purpose overlay-nodes of two kinds; Access Nodes (AN) through which alone the client-server communication takes place, and Public Servers (PS) which act as initial preparation points. The new system also assumes no client-side

modification or special downloads.

Then, a novel taxonomy of HTTP(S)-DDoS attacks is introduced organizing possible source behavior strategies from the AN's and PS's perspectives. In addition, enabled by the introduced enhanced identification, a unique reputation and penalty system based on three levels of behavior attributes records is designed. The novel reputation system requires no traffic decryption at the overlay nodes or special software at the client or server. While the PS and AN can be equipped with several degrees of countermeasures (CM) against attacks, each component in the proposed system is equipped with two degrees of mitigation measures. For the PS and AN, a practical client probing (slow-responding) mechanism is introduced to counter certain defense-unaware attacks. In addition, the AN's second degree CM analyzes the enhanced behavior records for only the per-session identification level to detect complex defense-aware attacks. Further, the PS's second degree CM aims to block PS-targeted attacks two steps away from the SS.

Furthermore, a proof of concept prototype of the proposed system is implemented, with non-optimally configured parameters to demonstrate the concept's soundness, and deployed on the DeterLab cyber-security testbed for experimental evaluation. At first, to examine the system's practicality and its transparency to both clients and servers, the prototype system is qualitatively tested for usability with actual commercial websites. Then, considering the goal of DDoS attacks, four metrics are defined for comprehensively measuring the mitigation effectiveness, namely; mitigation factor, cost, time, and collateral damage. Among the experiments conducted with simple and complex HTTP(S)-DDoS attack assumptions that are absent from related works and conventionally hard to detect, the results of seven experiments are presented and discussed, including; brute-force, below detection thresholds, single-request per-connection, slow-requesting HTTP-DDoS, multi-behavior per-shared-IP, and multi-vector attack conditions. For attack traffic, attack tools popular among attackers are utilized (i.e., LOIC and Slowloris) for experiments with limited number of sources (10 to 20 sources), and similarly built custom tools for highly distributed centrally controlled automated attacks (1,000 to 10,000 sources).

The results suggest that utilizing the introduced practical enhanced identification can eliminate the necessity for traffic decryption by overlay-nodes and for inspection of client-server traffic content. With the enabled reputation and penalty system, we can accomplish high mitigation factors of conventionally hard to detect HTTP(S)-DDoS attack categories in relatively short mitigation times, in contrast to conventional overlay-based methods. Especially considering complex attack conditions that are missing in related research such as single-request per-connection sub-detection-threshold HTTPS-DDoS. It suggests that less complex attack categories can be equally mitigated. In addition, results suggest that enhanced identification can achieve low collateral damage in terms of the chance of receiving service and service time for non-attacking clients that share an attack IP. However, the unoptimized implementation of

the prototype system shows a cost in service time even without attack. Also, it shows a temporarily decrease in mitigation factor and rise in cost during mitigation time. Experimentally demonstrating the concept's soundness based on the introduced per-session identification alone opens the way for investigating the inclusion of the three identification levels within various machine learning techniques for adaptive system parameters' tuning and for analyzing the enhanced behavior record patterns.

# Table of Contents

**Abstract** **ix**

**Table of Contents** **xiii**

**List of Figures** **xvii**

**Nomenclature** **xxi**

**1 Introduction** **1**

1.1 Background and Motivation . . . . . 1

1.2 Attack Categories . . . . . 3

1.2.1 Low-level DDoS Attacks . . . . . 3

1.2.2 High-level DDoS Attacks . . . . . 4

1.3 Mitigation Categories . . . . . 6

1.3.1 Locally-based Solutions . . . . . 6

1.3.2 Remotely-based Solutions . . . . . 7

1.4 Problem Statement . . . . . 9

1.5 Research Scope . . . . . 10

1.6 Thesis Organization . . . . . 12

**2 Related Works on Overlay-based HTTP-DDoS Mitigation** **13**

**3 Proposed Method and Prototype Implementation** **19**

3.1 Objectives . . . . . 19

3.2 Proposed Method . . . . . 20
3.2.1 System Overview . . . . . 20
3.2.2 Preparation and Communication Stages . . . . . 21
3.3 High-level Attack Strategies . . . . . 22
3.4 Mitigation . . . . . 25
3.4.1 Detection Concept . . . . . 26
3.4.2 Reputation and Penalty . . . . . 29
3.4.3 Attack Countermeasures . . . . . 35

**4 Evaluation** **41**
4.1 Evaluation Method . . . . . 41
4.1.1 Performance Metrics . . . . . 42
4.1.2 Emulation Platform (Testbed) . . . . . 44
4.1.3 Evaluation Plan . . . . . 45
4.1.4 System Parameters . . . . . 48
4.2 Experiments . . . . . 49
4.2.1 High Rate HTTP-DDoS via AN . . . . . 49
4.2.2 Slow-Requesting HTTP-DDoS via PS . . . . . 51
4.2.3 Distributed, High Rate, Slow-Requesting HTTP-DDoS via PS . 55
4.2.4 Highly Distributed, Low Rate, Slow-Requesting HTTP-DDoS via PS . . . . . 57
4.2.5 Slow-Requesting HTTP-DDoS via AN . . . . . 60
4.2.6 Low Rate HTTPS-DDoS via AN . . . . . 64
4.2.7 Multivector HTTP(S)-DDoS Attack via AN . . . . . 68

**5 Discussion** **73**
5.1 Service Time Cost . . . . . 73
5.2 Scalability . . . . . 74
5.3 Evaluation Method . . . . . 76
5.4 Attack Conditions . . . . . 78
5.5 Encryption and Trust . . . . . 81

5.6 Conventional Solutions . . . . . 82
5.7 Actual Implementation Considerations . . . . . 84

**6 Conclusion** **87**

**References** **89**

**Publications** **97**

# List of Figures


1.1 DDoS attacks architecture. . . . . 2

1.2 Scalability of locally-based methods against attack traffic is limited by the server's local resources and budget, thus not difficult to overwhelm by attackers. . . . . 7

1.3 In comparison to locally-based approaches (Fig. 1.2), shared overlay-based approaches provide high scalability against attacks at relatively low cost. . . . . 8

1.4 Conventional overlay-based solutions decrypt data in transit to mitigate complex HTTP(S)-DDoS attacks and base mitigation only per-IP or per-connection. . . . . 10

1.5 Conventional two-level identification (per-IP and per-connection) provides limited behavior attributes for the mitigation of HTTP(S)-DDoS attacks. . . . . 11

2.1 Architecture of sPoW. . . . . 14

2.2 Basic protocol of CLAD. . . . . 15

2.3 Architecture of OverCourt. . . . . 16

3.1 By practically enhancing the client identification at the overlay-nodes, the objective is to eliminate the necessity for traffic decryption by the overlay-nodes for effective mitigation of complex HTTP(S)-DDoS attacks remotely. . . . . 20

3.2 Enhancing conventional client identification by adding a third level of identification. . . . . 21

3.3 Proposed system overview [1–3]. . . . . 22

3.4 Taxonomy of attack source's possible HTTP(S)-DDoS strategies, from the AN's and PS's perspectives, read from left to right. For example, the set of strategies CB32** refer to single renewed session attack, with multiple fixed connections, through which multiple invalid-looking requests of alternating parameters are sent. . . . . 24

3.5 Top: Definition of three mitigation categories, with respect to; mitigation factor, time, and cost. Bottom: Expected mitigation category for different HTTP(S)-DDoS attack categories. . . . . 26

3.6 Detection attributes stored locally in hash maps by each AN and PS. ANs handle four levels of attributes' values, while PSs handle only two. 27

3.7 Description of stored attributes by the AN. . . . . 27

3.8 Description of stored attributes by the PS. . . . . 28

3.9 Considered AN and PS exceptions. . . . . 30

3.10 Each record vector registers the underlying exceptions during the latest $\beta$ time steps. . . . . 31

3.11 Reputations and penalties hierarchy. In current prototype implementation, $\mathrm{R_G} = 1$. . . . . 33

3.12 AN exceptions handling in the current prototype, given $\mathrm{R_G} = 1$. LHS: Local PID reputation ($\mathrm{R_L}(\hat{E}_P)$) states. RHS: SID penalty ($P(\hat{E}_S, \mathrm{R_L})$) states. . . . . 34

3.13 PS exceptions handling, given $\mathrm{R_G} = 1$. LHS: Local PID reputation ($\mathrm{R_L}(\hat{E}_O, \mathrm{R'_L})$) states. RHS: PID penalty ($P(\hat{E}_P, \mathrm{R_L})$) states. . . . . 35

3.14 Pre-service slow-response probing. . . . . 36

3.15 Function of $P_i$; the lookup table manually set for the duration before, between and after the slow-response probes. . . . . 37

3.16 Example of behavior record analysis countermeasure. . . . . 38

3.17 Pre-probing message reception and aggregation. . . . . 39

4.1 Apache web server benchmarks using ApacheBench for a 200 KB file. . 48

4.2 Setup and physical mapping of Experiment 1. . . . . 49

4.3 Results of Experiment 1. LHS: Start of attack. RHS: Switching attack. 51

4.4 Setup and physical mapping of Experiment 2. . . . . 52

4.5 Results of Experiment 2. Slow-requesting HTTP-DDoS attack starts nearly at t = 4 [min]. . . . . 54

4.6 Setup of Experiment 3. Same physical mapping of Experiment 2. . . . 55
4.7 Results of Experiment 3. Slow-Requesting HTTP-DDoS attack starts nearly at t = 2.5 [min]. . . . 57
4.8 Setup and physical mapping of Experiment 4. . . . 58
4.9 Results of Experiment 4. . . . 59
4.10 Setup and physical mapping of Experiment 5. . . . 60
4.11 Service Measurements of the test web server under direct slow-requesting HTTP-DDoS attack. . . . 62
4.12 Results of Experiment 5. . . . 63
4.13 Setup and physical mapping of Experiment 6. . . . 64
4.14 Service Measurements of the test web server under a very low rate, direct HTTPS-DDoS attack. . . . 66
4.15 Results of Experiment 6. . . . 67
4.16 Setup and physical mapping of Experiment 7. . . . 68
4.17 Results of Experiment 7: Traffic and Service Measurements via MN, M-shared-1, and M-shared-2. . . . 70
4.18 Summary of experiments. . . . 71

5.1 Non-attack service time measurements. . . . 74
5.2 Proposed vs conventional solutions. . . . 84

# Nomenclature

The next list describes several acronyms, terms, and symbols that will be later used within the body of this dissertation.

### Acronyms and Terms

APR Access permission request, page 21

ACK Acknowledge message of the TCP protocol, page 4

AN Access Node, page 20

AP Access Permission, page 21

CD Collateral damage, page 43

CDN Content Delivery Networks, page 13

CID Connection identifier, page 29

CM Countermeasures, page 35

CoS Chance of service completion, page 42

DDoS Distributed Denial of Service, page 1

DNS Domain Name System, page 4

DNSSEC Domain Name System Security Extensions, page 5

DP Default prevention, page 25

DRP Detection response prevention, page 25

DRR Detection response reduction, page 25

EDoS Economic Denial of Sustainability, page 4

FS Failed session, page 28

HOIC How Orbit Ion Canon attack tool, page 4

HTTP Hypertext Transfer Protocol, page 4

HTTP(S)-DDoS Used as a superset reference to HTTP-DDoS and HTTPS-DDoS, including slow-requesting HTTP-DDoS, page 5

HTTPS HTTP secure, page 5

ICMP Internet Control Message Protocol, page 3

IIS Internet Information Services web server, page 6

ISP Internet Service Provider, page 6

LHS Left hand side, page 32

LOIC Low Orbit Ion Canon attack tool, page 4

max Maximum value, page 32

MC Mitigation cost, page 42

MF Mitigation factor, page 42

MITM Man in the middle, page 81

MONLIST Monitor list command of the NTP protocol, page 5

MT Mitigation time, page 42

NTP Network Time Protocol, page 4

OID Origin identifier, page 28

PID Client-server pair identifier, page 28

PS Public Server, page 20

RF Reduction factor of attack traffic, page 42

RHS Right hand side, page 33

RST Reset message of the TCP protocol, page 4

RUDY R-U-Dead-Yet attack tool, page 5

SID Session identifier, page 28

SME Small to Medium sized Enterprise, page 7

SNI Server Name Indication, page 81

SS Secret Server, page 20

SSL Secure Sockets Layer, page 4

ST Service time for a specified file, page 42

SYN Synchronize message of TCP, page 3

TCP Transmission Control Protocol, page 3

TLS Transport Layer Security, page 4

UDP User Datagram Protocol, page 3

**Symbols**

$\alpha$ Time step duration, page 27 [sec]

$\beta$ Number of record past time steps, page 27

$\gamma$ Decimal value of $\hat{E}_S$, page 34

$\hat{E}_O$ Origin-level exceptions record vector of length $\beta$ time steps, page 30

$\hat{E}_P$ Pair-level exceptions record vector of length $\beta$ time steps, page 30

$\hat{E}_S$ Session-level exceptions record vector of length $\beta$ time steps, page 30

A/P Attempts per client-server pair, page 29

A/$\alpha$ Attempts per time step, page 29

$A_i$ Current number of simultaneous attempts, page 29

FA/P Failed attempts per client-server pair, page 29

FA/$\alpha$ Failed attempts per time step, page 29

IA/O Incomplete attempts per origin, page 29

IA/$\alpha$ Incomplete attempts per time step, page 29

$IP_C$ IP address of client C, page 21

$IP_N$ IP address of AN, page 21

m Number of considered consecutive time steps ($m \leq \beta$), page 28

PID/O Client-server pairs per origin (i.e., per source IP), page 29

PortN Port temporarily assigned by the AN for a client, page 21

R'$_{\mathrm{L}}$ Local reputation received from the AN by the PS, page 32

Warn$_{\mathrm{AN}}$ Warning from AN, page 29

$C/\alpha$ Connections per time step, page 28

$C/S$ Connections per session, page 28

$C_i$ Current number of open connections, page 28

$E_*$ Exception number *, page 30

$FS/\alpha$ Failed sessions per time step, page 28

$FS/P$ Failed sessions per pair, page 28

$L_1$ Level 1 of local reputation (normal), page 32

$L_2$ Level 2 of local reputation (suspicious), page 32

$L_3$ Level 3 of local reputation (bad), page 32

$M/\alpha$ Messages per time step, page 29

$M/C$ Messages per connection, page 29

$P_i$ Penalty, page 28

$R_G$ Global reputation of the source IP, page 28

$R_L$ Local reputation per PID, page 28

$S/P$ Sessions per pair, page 28

$S_i$ Current number of open sessions, page 28

$t_1$ Duration before the first probe, page 36

$t_2$ Duration between the first and second probes, page 36

$t_3$ Duration after the second probe, page 36

$T_{a(max)}$ Maximum aggregation time of a request, page 38

$t_{att_{start}}$ Attack starting time, page 42

$t_{p_{max}}$ Time at which an SID reaches max penalty, page 42

$T_{r(max)}$ Maximum request time, page 38

$t_{req_{sent}}$ Time the request was sent, page 42

$t_{resp_{full}}$ Time the response is fully received by client, page 42

$T_S$ SID starting time, page 29

$w_i$ Current time step $i$, page 27

$w_{ref}$ Reference time step clock, page 27

$Warn_{SS}$ Warnings from SS, page 28

$P_{max}$ Penalty's maximum value, page 33

$P_{min}$ Penalty's minimum value, page 33

# Chapter 1

# Introduction

## 1.1 Background and Motivation

The Internet has become the main infrastructure for a global information-based society. Research and educational institutes depend increasingly on the Internet as a platform for collaboration and as a medium for disseminating their research discoveries rapidly. Also enterprises of various sizes offer secure Internet-based transactions, as well as share and exchange information with their divisions, suppliers, partners and customers efficiently and seamlessly. Likewise, governments are increasing their reliance on the Internet to provide information to the public, in addition to various government services online.

With the growth of the Internet, cyber attacks have also evolved fast. Further, traditional operations in vital services, such as banking and medicine are being provided over less expensive and more efficient Internet-based applications. Historically, an attack on a nation's critical services involved crossing physical boundaries. However, the global connectivity of the Internet renders physical boundaries meaningless. Internet based attacks can be launched from across the globe. Therefore, the reliability and security of the Internet not only benefits online businesses, but is also an issue for global security and human safety.

The continuous rise in cyber attacks constitutes a serious threat to Internet users. Such attacks vary in their threat model and sophistication, which depend on the attacker's technical knowledge, among other factors. One of the most common cyber attacks to date, due to its simple yet effective nature, is DDoS attacks [4–6].

DDoS attacks' basic concept is to overwhelm an internet-based service (e.g., twitter.com) with requests from distributed sources, so that actual users are denied service. It can be driven politically, for money, fame, or as a cover for a simultaneous intrusion attempt. Further, DDoS attacks are not difficult to launch against a web server

(i.e., victim) and is likely to remain so with the current nature of the internet. In fact, DDoS attack tools can be downloaded easily from the internet, in addition to non DDoS, network testing, tools that can be used with a bad intention. Over the internet, attackers infect and control their agents remotely, and the messages sent are network packets with little to no cost for the original attacker. It becomes even easier for attackers, with for-rent botnets that can only cost 20 US Dollars to hire for 1 hour [7].

To avoid being traced back and held accountable, attackers tend to format their attack agents into layers. Such agents are compromised firstly by the attacker, before the actual DDoS can take place. As shown in Fig. 1.1, the first layer is composed of handlers; where the attacker compromises one or more machines and through these controls the following layer's machines (i.e., control traffic). The second layer is composed of attack agents (also called zombies, bots, or slaves); where the actual DDoS attack traffic originates (i.e., attack traffic). If the botnet, or part of it, is detected, then the handlers may be exposed in this case but not the original identity of the attacker. Also, more layers of handlers can be used. This research pays attention to mitigating the effect of attack traffic on web servers, while the detection of control or compromise traffic is within the scope of a different research area.

The impact of a DDoS attack may not just mean missing out on the latest sports scores or weather news on the Internet. It may mean losing a bid on an item you want to buy or losing your customers for a day or two while you are under attack. Attacks affect many sectors, including; commercial, industrial, media, and financial institutions. A single attack would sustain for periods ranging from a few minutes to even several days [8]. A recent study, surveying 1,002 IT professionals worldwide, 76%

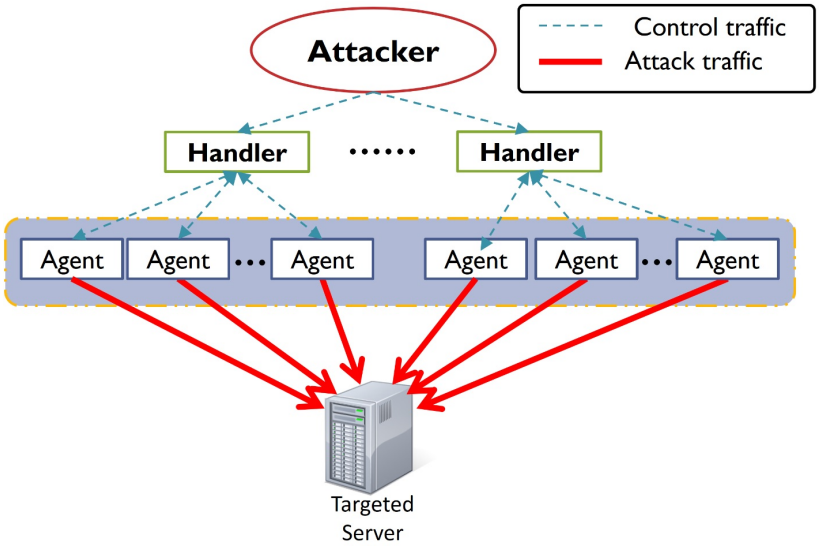



Figure 1.1 DDoS attacks architecture.

reported investing more in DDoS mitigation than a year ago [9]. This suggests the economic feasibility of this research's product.

## 1.2 Attack Categories

DDoS attacks vary in the underlying protocol (high-level protocol based, network based, etc.), rate (brute force "flooding", below detection threshold, etc.), targeted resources (limited memory, bandwidth, or CPU capacity), etc. Different attack categories can have overlapping attack parameter variations. Generally, attacks are either on the low-level (utilizing transport or network layer protocols) or high-level (utilizing application layer protocols). Both categories continue to be on the rise [10].

### 1.2.1 Low-level DDoS Attacks

Low-level DDoS attacks generally rely on traffic volume to overwhelm the victim, with peak rates of 470 Gbps being reported recently [8]. Source IP spoofing is usually used with low-level attacks to hide the attacking agents true identity and in some cases to amplify the attack traffic. Low-level attacks include two major subcategories; protocol exploitation, and flooding attacks.

Protocol exploitation attacks misuse of a certain protocol features to disrupt communications. A popular example is the TCP SYN attack; the exploited feature is the allocation of a space in a connection queue immediately upon receipt of a TCP SYN request. The attacker initiates multiple connections that are never completed, thus filling up the connection queue spaces. A similar version are called state exhaustion attacks. For example, attackers attempt to consume the local limits of the target server's settings by opening multiple connections simultaneously, and keeping them open, thus exhausting the server's configured local limit.

Flooding attacks try to congest the victim' link by sending large volumes of traffic to it. No intelligence is required from the attack agent in this case, in comparison to the protocol exploitation attacks. The effect of such action on the victim system varies from slowing it down or crashing the system to saturating its network bandwidth. Some of the well-known flooding attacks are UDP flood attacks and ICMP flood attacks.

To achieve a large attack volume, attackers often resort to traffic amplification. Amplification of attacks utilize intermediary nodes that are used as attack launchers (called reflectors) to amplify and reflect the attack. The result is also a flood of packets, but the difference is the method of generation. A reflector is any IP host that will return a packet if sent a packet. Spoofing the source address is the key factor

in executing such attacks. So, web servers and routers can be reflectors, returning SYN/ACK or RST in response to SYN or other TCP packets. An attacker sends packets that require responses to the reflectors. The reflected packets can flood the victim's link if the number of reflectors is large enough. Note that the reflectors are readily identified as the source addresses in the flooding packets received by the victim. The operator of a reflector on the other hand, cannot easily locate the agent that is pumping the reflector with the source address spoofed as the victim's.

In summary, low-level DDoS attacks rely on large traffic volume. Therefore, they require not only accurate detection of attack traffic, but also scalable server resources, especially bandwidth to keep the service available.

### 1.2.2 High-level DDoS Attacks

In contrast to low-level DDoS attacks, high-level attacks 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 [8]. High-level DDoS attack models target one of three resources; network (e.g., bandwidth bottlenecks, connection queues, etc), processing (e.g., CPU, memory, processing queues, etc) or economic. The latter is also called EDoS attacks which target the economic sustainability the victim server's paid-for services such as an on-demand DDoS defense service that charges the server per usage.

The source addresses of high-level DDoS requests are of real hosts, since a valid application level request requires a successful connection. Therefore, spoofing and amplification is not valid in this case. Exceptions are UDP-based high-level attacks, where there is no connection established, such as NTP and DNS amplification DDoS attacks.

Methods for high-level DDoS attacks include; HTTP-based (i.e., HTTP-DDoS), SSL-based [1], and amplification (e.g., NTP, DNS, etc.) attacks.

An HTTP-DDoS attack targets a web server with a HTTP GET or POST requests. Attackers often employ botnets to generate large traffic. The web server becomes overwhelmed by attempting to answer each seemingly-legitimate request. As a result, actual legitimate requests can be denied from getting the service. Common HTTP-DDoS attack tools include LOIC and HOIC.

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

[1] Both TLS and SSL are henceforth simply called SSL.

server times out their requests. Note that "slow" here indicates the prolonged time for an HTTP to arrive, while the rate of request parts' arrival can still be high or low. For example, we can see a high-rate slow-requesting HTTP-DDoS attack. Common slow-requesting HTTP-DDoS attack tools include Slowloris and RUDY.

SSL-based attacks either try to abuse the SSL protocol itself, or simply send the HTTP-DDoS attack over an encrypted connections (i.e., HTTPS-DDoS). In the former, sources initiate regular SSL handshakes then repeatedly request encryption key renegotiation to overwhelm the server's resources. On the other hand, HTTPS-DDoS mainly aim to complicate the mitigation of attacks.

Henceforth in this dissertation, the term "HTTP(S)-DDoS" is used as a superset reference to HTTP-DDoS and HTTPS-DDoS, including slow-requesting HTTP-DDoS. This category of attacks can't be amplified. The request must originate from a true source address which can't be spoofed. The fact that the source of the HTTP(S)-DDoS request is a real host implies one of two possibilities, where requests either originate from compromised hosts (bots) or from a coordinated group of people who decide to participate in an attack. However, there is not an easy way to distinguish the good requests from the bad, for example; bots sending requests to a server can actually be good bots (crawlers), also for the case of human-operated agents, they could be just a crowd of people genuinely interested in a certain resource together.

In addition, high-level amplification attacks utilize source address spoofing. For example, the NTP amplification attacks utilize servers that support the MONLIST command as reflectors. A large number of UDP packets are sent by the attacker to the reflector with the source address spoofed with that of the victim. This way, reflectors amplifies the attacker's achievable traffic, and also helps the attacker hide its identity. The MONLIST command can be simply disabled by NTP servers, without impacting their function, to avoid participating in this type of attack. On the other hand, DNS amplification attacks rely on open DNS resolvers as reflectors. A small query by the attacker, also with spoofed source address, can result in a much larger response. Even with DNSSEC, which is designed to make the DNS more secure, such attack model is still viable. Proper configuration of DNS servers can limit their participation in such attacks.

Note that from perspective of the protocol exploited at the reflector, NTP and UDP amplification attacks are high-level attacks. Yet, they can also be classified as low-level attacks from the perspective of the reflected UDP traffic targeting the victim. Either way, such amplification attacks are similar to low-level attacks in their reliance on traffic volume to overwhelm the victim. In contrast to HTTP(S)-DDoS, these can be more easily detectable but the large volume is what makes them difficult to mitigate in case of non-scalable server resources.

To further understand HTTP(S)-DDoS, they can be sub-categorized based on source behavior. Section 3.3 further discusses the classification of possible HTTP(S)-

DDoS attack strategies with the aid of a novel taxonomy. However, for now, it's necessary to notice that each attack strategy can be described as either complex or not complex. So, a low-rate below detection thresholds attack is more complex (i.e., knowledge-based, defense-aware, or smart attack) than a high-rate attack (i.e., defense-unaware, or blind attack). Similarly, a single-request per-connection attack is more complex for methods that rely on per-connection behavior analysis alone than a multi-request per-connection attack. Likewise, an encrypted attack is more complex for an overlay-based mitigation method than an unencrypted one. Section 1.3.2 explains why the focus is on the overlay-based mitigation approach.

## 1.3 Mitigation Categories

To mitigate DDoS attacks, there are two main approaches according to the location of deployment. Mitigation solutions can be either; locally-based (i.e., at the server's premise), or remotely-based (i.e., at source, infrastructure, ISP, or an overlay network).

### 1.3.1 Locally-based Solutions

Locally-based DDoS mitigation solutions are components, or settings, configured within the targeted server's network to reduce the attack's impact. Undesired DDoS traffic is therefore locally mitigated.

For example, to help mitigate the low-level TCP SYN floods, locally configuring SYN-cookies can be done [11]. On the high-level, there are also several basic mitigation configurations that aim to protect the server against some of the basic attacks. Apache, the leading web server to date, offers the mod_evasive module [12]. It provides simple DDoS protection as part of the HTTP server. Additional basic settings for the local server can also help mitigate certain basic DDoS attacks [13]. Other web servers, such as IIS and nginx, etc., also have their basic mitigation modules. The reader is encouraged to read further about them. For example, the mod_evasive Apache module helps mitigate HTTP-DDoS attacks against the server by providing evasive actions during attacks and report abuses to file or by email. The module creates an internal dynamic table of IP addresses and URIs. Any single IP address is denied by the module from requesting the same page more than a number of times per second, making more than a set number of concurrent requests on the same child process per second, or making any requests while temporarily being blacklisted. While mod_evasive can inexpensively help protect the server against some HTTP(S)-DDoS attacks, complex low-rate highly-distributed attacks may successfully evade the module's evasion.

In addition, commercial specialized solutions offered by vendors such as A10 Networks® provide various locally-based mitigation hardware [14]. Prices start from

195,000 US dollars per unit [15]. Add to that the expected costs for deploying, upgrading, maintaining and over provisioning for such solutions. Only large-sized enterprises may afford such expenditures. This makes them not suitable for SMEs which require affordable, scalable, and practical solutions (i.e., the main features of remotely-based solutions).

In summary, locally-based solutions may be effective, but attackers can still easily exceed the server's limited network, or processing, resources available for similar locally-based solutions. Especially in case of volumetric attacks as illustrated in Fig. 1.2. It the attack traffic exceeds the limited local resources, it becomes difficult to provide normal service to users regardless of the detection method's accuracy. Nonetheless, locally-based solutions can still be useful if augmented with the more scalable remotely-based mitigation solutions.

### 1.3.2 Remotely-based Solutions

Remotely-based mitigation solutions are those not located at the server's premise. These include four types, namely; infrastructure-based, source-based, ISP-based, and overlay-based.

Source-based approaches assume cooperation from the source host or ISP [16, 17]. On the other hand, infrastructure-based approaches aim to protect the server far from its bottlenecks by considering modifying or installing additional equipment at the Internet infrastructure, or contiguously modifying routers on the path to the victim [18–21]. However, the general problem with such approaches is the lack of economic (and political) incentives for all the sources to cooperate. Similarly, expecting providers

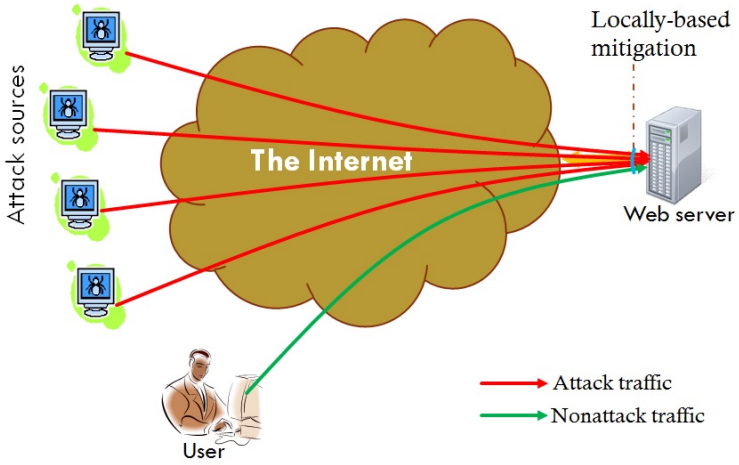



Figure 1.2 Scalability of locally-based methods against attack traffic is limited by the server's local resources and budget, thus not difficult to overwhelm by attackers.

to allow the installment of special filters on their own routers assumes a doubtful degree of collaboration. Besides, those modified routers would have to maintain traffic flow states, introducing additional load on them. Furthermore, it raises concerns about the possibility of abuse. In addition, ISP-based approaches are offered by the victim's ISP, where the distributed attack traffic converges. Offloading the response to ISPs can be useful since they are well situated to respond to relatively large volumes of DDoS traffic [22, 23]. However, ISPs themselves can become the victims of massive DDoS attacks [24, 25].

On the other hand, the overlay-based paradigm eliminates direct client access to servers by deploying a scalable overlay network of mitigation nodes (also called edge servers, edge nodes, or overlay nodes), as illustrated in Fig. 1.3. The overlay-nodes are managed by a specialized third-party mitigation service provider. Ideally, the undesired traffic is remotely identified and filtered out.

Notice that although both terms overlay-based and cloud-based can be used interchangeably in various instances, the term cloud has been recently associated by many people with loss of privacy and trust [26–28]. Also, the term cloud-based mitigation can imply data storage by the cloud nodes, which is strictly avoided in this research as explained further in section 1.4. So, this dissertation refrains from using the latter term, and uses the overlay term instead.

Several commercial overlay-based DDoS mitigation services are available [29–31]. For example, Akamai® offers managed DDoS mitigation solutions [29]. Their distributed edge servers act as a buffer to the origin servers (the victim). Only traffic passing through Akamai's edge servers can connect to the victim's infrastructure. Sim-

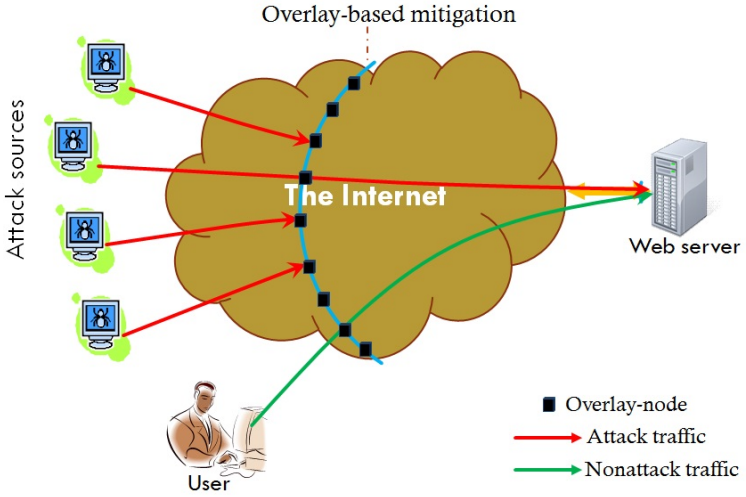



Figure 1.3 In comparison to locally-based approaches (Fig. 1.2), shared overlay-based approaches provide high scalability against attacks at relatively low cost.

ilarly, VeriSign® promises protection far from the servers' location [32]. A monitoring facility inspects the victim's traffic parameters. In case of a detectable attack event, VeriSign works with the victim to divert traffic to a dedicated filtering site.

The capacity of such overlay-based solutions can be elastic enough to scale up as DDoS attacks grow in bandwidth and processing requirements. In addition, the cost of overlay-based services is significantly lower than the locally-based ones, which is a feature especially attractive to SMEs. The cost of a commercial solution can start from as low as 5,000 US Dollars per month [33], dependent on application requirements. Compare this to the starting cost of 195,000 US dollars per single locally-based hardware unit [15]. Also, the costly locally-based solutions require hiring skilled personnel to manage it, as well as keeping it up to date. On the other hand, a third-party-managed security service can reduce such operational and maintenance costs significantly. So, the main advantages of overlay-based approaches, in contrast with locally-based and other remotely-based ones, are; practicality, scalability, and affordability. Especially so for SMEs.

## 1.4 Problem Statement

We learned that overlay-based mitigation solutions against DDoS attacks represent the most practical, scalable, and affordable option for SMEs.

In case of low-level DDoS, overlay-based mitigation is well established [1, 34]. 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, overlay-based mitigation of high-level attacks can be a challenging task. Not surprisingly, recently more frequent high-level attacks are reported [6], especially the most hard to detect HTTP(S)-DDoS attacks [35].

Conventional overlay-based solutions can mitigate HTTP(S)-DDoS, however they suffer from at least one of the following two demerits:

**1) Traffic decryption:** To inspect the content for mitigation, the client-server SSL connections are split, as illustrated in Fig. 1.4, with overlay nodes decrypting then re-encrypting traffic [29–31]. This means trusting a third-party, in the middle, capable of decrypting what's supposed to be true end-to-end encrypted traffic. This 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 [26, 27]. This has caused organizations to evolve their thinking with respect to encryption key control and data residency, as shown in a recent study [28]. Surveying 5,009 IT professionals in 11 countries shows that 76% of

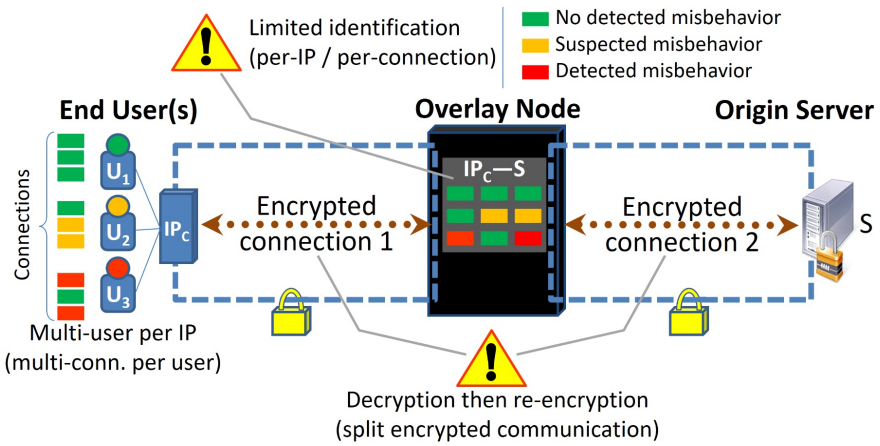



Figure 1.4 Conventional overlay-based solutions decrypt data in transit to mitigate complex HTTP(S)-DDoS attacks and base mitigation only per-IP or per-connection.

financial service organizations are most likely to control encryption keys within their organization rather than a third-party provider.

**2) Limited identification:** Conventional practice of major providers (such as [36]) assume that a client behavior is identifiable based on its source IP address (i.e., per-IP identification) and/or based on behavior of individual connections (i.e., per-connection identification). We call this two-level identification, as illustrated in Fig. 1.5, which provides limited behavior attributes and limits the ability to mitigate complex HTTP(S)-DDoS attacks. For example, 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 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.

## 1.5 Research Scope

It's reported by CISCO that HTTP will occupy 80% of the Internet traffic by 2019. Also, general web-services are increasingly switching to the secure version of HTTP (i.e., HTTPS), not only shops, hospitals and banks. So, for a DDoS mitigation scheme to be widely accepted, it must not contradict with the purpose of HTTPS in particular. In Chapter 2, we review related works which mostly focus on mitigating not so complex categories of HTTP(S)-DDoS such as the multiple-requests per-connection category, and either overlook the true end-to-end encryption requirement, suffer from limited identification, or both.

So, the focus of this research is on overlay-based mitigation of HTTP(S)-DDoS, par-

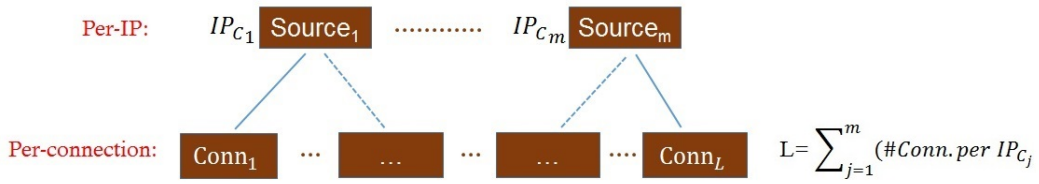



Figure 1.5 Conventional two-level identification (per-IP and per-connection) provides limited behavior attributes for the mitigation of HTTP(S)-DDoS attacks.

ticularly on its more complex versions, namely: single-request per-connection, multi-behavior per-shared-IP, sub-detection-thresholds, slow-requesting HTTP-DDoS, and multi-vector attacks.

To regain the trust of both users and servers, 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 goal of the new system presented in this dissertation is to enable effective mitigation of complex HTTP(S)-DDoS attacks far from the server, while complying with the encryption requirement [2, 3]. Section 4.1.1 explains in more detail the metrics that 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 (i.e., enhanced identification), allowing an overlay-node to group related connections per client. With a focus on HTTP(S)-DDoS, a practical third level of identification is utilized enabling novel behavior-based reputation and penalty system.

Aiming to demonstrate the soundness of the concept, a proof of concept prototype with simplified mitigation measures and parameters is implemented and deployed on DeterLab [37]. Several experiments are conducted to evaluate the soundness of the concept. Among them, the results of seven experiments are discussed.

**The main contributions of this research are:**

- Design the first overlay-based DDoS-mitigation system that enables both enhanced identification and true end-to-end encryption.
- Introduce a new taxonomy of HTTP(S)-DDoS attacks source-strategies.
- Design a novel behavior-based reputation and penalty system.
- Develop a proof of concept prototype to experimentally demonstrate the soundness of the proposed concept.
- The first research to discuss evaluation results against the complex low-rate single-request-per-connection HTTP(S)-DDoS attack categories.

## 1.6 Thesis Organization

The rest of this dissertation is organized as follows. Chapter 2 surveys relevant conventional DDoS mitigation approaches. The chapter concludes by qualitatively summarizing the comparison between the conventional methods. Chapter 3 follows by explaining the proposed method and details of the proof of concept prototype implementation. Chapter 4 presents the experimental evaluation results. Chapter 5 discusses several points on the presented research. Finally, chapter 6 concludes the work and draws lines for possible future research directions.

# Chapter 2

# Related Works on Overlay-based HTTP-DDoS Mitigation

Various overlay-based approaches to mitigate HTTP-DDoS attacks are proposed in literature by researchers over the past decade. Methods vary, employing Software Defined Networking (SDN) [38], dynamically instantiated replica servers [39], content caching [40], pre-service verification [41–45], and credit-based accounting [46, 47].

A common feature of overlay-based approaches is the reliance on content inspection to mitigate HTTP-DDoS. For example, the work in [39] propose a framework where overlay-based mitigation servers are set up as moving targets. In case of an attack, new servers are dynamically instantiated to replace the attacked ones, and affected clients are reassigned to other servers. Although similar methods show a capability to mitigate certain attack categories, it's assumed that the overlay-based replica servers can access the high-level client-sever content.

As a countermeasure to high-level attacks against CDNs, the works in [41,42] utilize a method for pre-service verification. Traffic is prioritized using a self-verifying Proof-of-Work (sPoW) scheme, where sPoW tests the client with a crypto-puzzle of a varying difficulty. For that, clients are required to download a special plug-in. However, the assumed level of users' trust in the system to download the plug-in from a third-party DDoS mitigation provider is questionable, especially with recent studies about trust and privacy [26, 27]. Also, if the sPoW scheme is eventually adopted by multiple competing DDoS mitigation providers, this would imply that a client may end up downloading multiple special plug-ins. Further, the server too is required to install a special plug-in for the system to work, which adds to the server's local load and assumes a certain level of trust from the server's organization, which may prohibit the adoption of the system by most of financial service and healthcare organizations [28]. In addition, based on the system's design, high-level attackers could download the plug-in then start requesting new puzzles without solving them. This way, a single

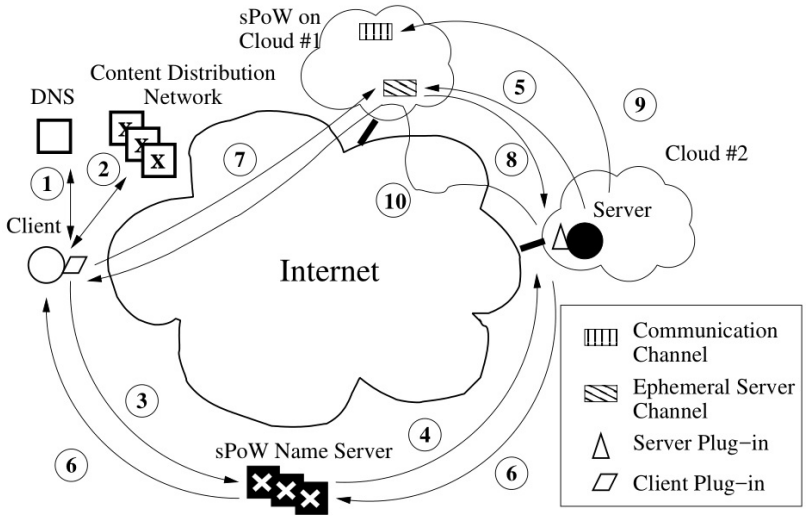



Figure 2.1 Architecture of sPoW [41], [42].

puzzle request from a client plug-in (i.e., step 3 of Fig. 2.1) results in two operations by the server's locally installed plug-in, including the generation of a crypto-puzzle (i.e., steps 5 and 6 of Fig. 2.1). Such asymmetry in workload may be offering attackers a new DDoS attack vector to exploit.

Pre-service verification has been also considered in [43]. The researchers propose a DDoS mitigation system based on cloud computing named CLAD. This system provides mitigation against high-level DDoS attacks, and assumes no modifications to server software. After the client's first DNS query, all its high-level requests are sent though cloud nodes (see Fig. 2.2). The client's request is replied with a graphical Turing test page which acts as a pre-service verification step to ask for authentication. After the client passes the test, the cloud node generates a session key and sends it to the client. The client then uses the key as part of the URL or stores it in a cookie, to be allowed access through the cloud nodes. However, the researchers assume the ability of the cloud nodes to read every request from the client to the server. It's also assumed that the nodes can generate a test web page to the client. Therefore, true end-to-end encryption between the client and the server conflicts with the proposed method.

Also, overlay-based methods have been proposed that utilize client accounting to mitigate DDoS. For example, [46, 47] present an shared overlay-based method named OverCourt to mitigate low-level DDoS with anti-spoofing mechanisms and high-level DDoS. The credit system assumes symmetry of the two-way TCP traffic and assumes no modifications to clients, servers or standard protocols. Good clients are allowed access to the servers through VIP channels [46] (also called normal channels [47]) with high priority, while the rest of clients are required to compete over a limited fraction

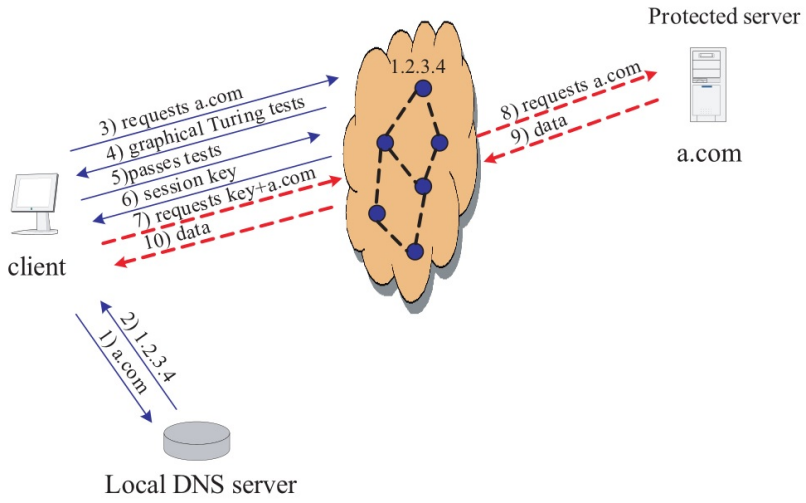



Figure 2.2 Basic protocol of CLAD [43].

of the server's capacity. The protected server is hidden from the Internet and is only accessible from Credit Routers (CR) which account for the credit points of each client, while OverCourt Gateways (OCGs) perform path migration (see Fig. 2.3). Utilizing a credit-based accounting mechanism, clients are classified by the system as good or bad, based on each client's low-level communication patterns history. Then, clients may still launch a low-rate HTTP-DDoS attack. So, the researchers assume that such attack should be fully detected locally, for example by serving graphical Turing tests to clients. So, the integration of similar methods with a locally-based detection method is also possible, such as the work proposed in [48] which provides a framework for local classification of HTTP requests given parameters from the multiple layers of the protocol stack. However, relying alone on reactive local detection may only stop small volumes of attacks and is likely to overload the server's limited resources as attacks grow, thus resulting in service denial. It's desired to automatically mitigate as much of possible of the attack traffic far from the server's local resources, while local detection should only act as a supplementary component for the fraction of attacks unmitigated remotely. Also, excessive reliance on pre-service graphical tests may annoy the web server's users. In addition, the proposed credit system is based only per-IP. So, a source IP that opens multiple connections to a server will have only one credit value regardless of the number of users sharing it. This may give a bad credit to multiple good users as a result of a single bad user, even after the attack stops.

As overlay-based methods provide a scalable platform for mitigation (i.e., detection and response), response is often focused at the overlay-nodes [38–47], while detection operations may be distributed both locally and remotely. A plethora of detection approaches against HTTP-DDoS have been proposed in literature. Several approaches adopted statistical and machine learning techniques to profile the traffic to detect

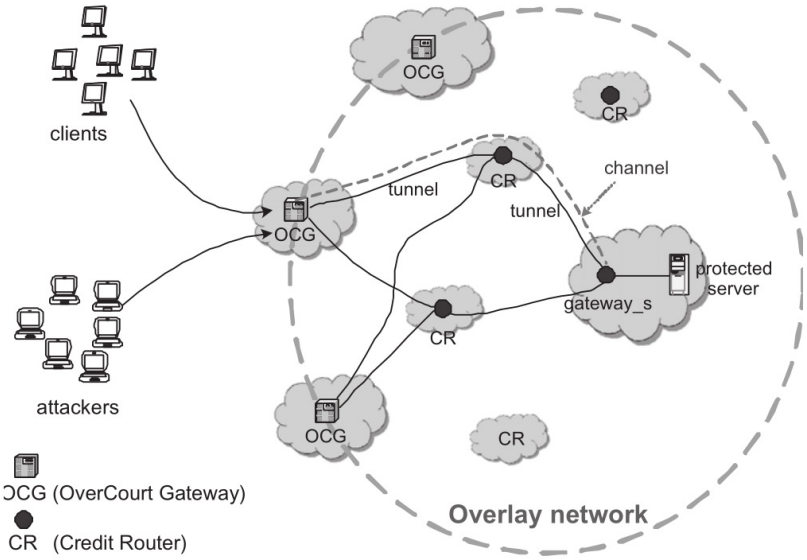



Figure 2.3 Architecture of OverCourt [46].

high-level attacks [49–54]. For example, the approach in [53] adopt Adaptive Selective Verification (ASV) to detect slow-requesting HTTP-DDoS. The behavior of the proposed method depends on the number of pieces of data already processed. An acknowledgment is sent only when the total payload is received. Therefore, inspecting and analyzing the received payload pieces is mandatory. Via simulation with 280 high-level attack sources, their method shows up to 80% chance of service completion for non-attacking users during attack time [53]. Similarly, the work in [54] proposed using statistical analysis to build connection scores based on several traffic attributes, including the browsing behavior (i.e., requested page type, popularity, page request frequency, etc.). Experimental evaluation under 600 attack source IPs show a reduction in attack traffic by 50% within 3 minutes, without reducing non-attack traffic. Although similar approaches show high performance in terms of attack reduction and detection speed, in principle they assume high-level data inspection. So, they are better suited locally to comply with the true end-to-end encryption requirement. Also, with only per-IP and per-connection identification attack categories such as multi-request per-connection can be detected, yet mitigation of the more complex cases of single-request per-connection attack categories is doubtful.

Further, more methods considered game-theoretic techniques to detect zero-day DDoS attacks [55–59]. In general, such approaches model DDoS attacks as a game between each user node (i.e., attackers and legitimate users) and the defender (i.e., the mitigation system). 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). For example, the approach employed in [57] models a DDoS attack as a one-shot, non-cooperative,

zero-sum game between each attacker node and the defender. Attackers strategies are based on four parameters; the number of attack nodes, the distribution of the flow rate of each node, the flow rate mean value, and the standard deviation for the distribution used. The response mechanism is firewall parameters' modification, assuming constant attack conditions. Assuming only local data inspection under the server's authority, such methods can comply with the encryption requirement. However, such methods identify and rate limit users per-IP or per-connection. Consequently, a complex HTTP-DDoS attack, for example, with a distributed low-rate single-request per-connection (i.e., low-bandwidth cost on individual attack sources) can evade such detection methods.

The work in [60] propose detecting high-level DDoS attacks using an ensemble of adaptive and hybrid neuro-fuzzy systems. No packet payload inspection is assumed, and so if deployed remotely it still can comply with true end-to-end. Several features are extracted from TCP connections and utilized in detection, including; the number of connections to the same host during specified time window, the number of connections having SYN errors using the same service during specified time window, and the variance of time difference between two consecutive packets. While evaluations show high detection rates, the source is identified only per-IP or per-connection, thus featuring the same demerit of limited identification.

Further, entropy-based approaches can detect DDoS attack connections [61, 62]. For instance, the work in [61] considers real-time handling of HTTP-DDoS mimicking flash-crowd traffic. Constructing a Real-time Frequency Vector (RFV) in order to characterize traffic as a set of models. Monitored traffic attributes include the average frequency for each website URL and the observed number of HTTP packets received. Therefore, attacks are recognized by examining the entropy of HTTP-DDoS and flash crowds. However, among the attributes utilized for detection, content inspection is assumed. In contrast, the work in [62] bases detection on analyzing packets' sizes, instead of contents. The distribution on the packet sizes is used to distinguish the attack flows from normal traffic. It is based on the assumption that the attack packet size is concentrated in an interval while the non-attack packet size would have a random distribution. Their software simulation results show the method's effectiveness in distinguishing automated attack traffic from legitimate traffic in low packet rate as well as high packet rate. Although similar methods don't assume client-server data inspection, their definition of a flow is limited only based per-IP and per-connection.

Aiming at enhancing client identification, several approaches proposed utilizing packet header fields as marking fields to distinguish between different clients per-IP [63, 64]. Although such approaches can be fitted as overlay-based methods, there's no described method for how different client-related connections can be identified for marking by the third-party without remotely inspecting the traffic content.

In summary, related works show a conflict between protecting web servers against

complex HTTP(S)-DDoS attacks and preserving users trust. Locally-based mitigation approaches benefit from the high-level knowledge of users' browsing behavior and can selectively identify attack-related connections. Yet, with the rise in DDoS attacks volumes, it's easy to exceed the limited locally-based mitigation resources. Although original locally-based methods can be fitted for remote deployment at overlay-based bodes, this would also imply remote traffic decryption by a third-party. Eliminating this conflict is necessary in order to promote the trust of users and organizations in such a crucial third-party managed service such as DDoS mitigation.

It's desired to have a practical, scalable, and affordable overlay-based mitigation solution capable of highly mitigating the HTTP(S)-DDoS traffic remotely, without contradicting with the true end-to-end encryption requirement, while any remaining unmitigated fraction of the attack traffic is detected locally.

# Chapter 3

# Proposed Method and Prototype Implementation

## 3.1 Objectives

Overlay-based mitigation solutions against DDoS offer high scalability and low cost, making them an attractive option for SMEs. However, such solutions suffer from either; traffic decryption, limited identification, or both.

The reason for traffic decryption stems from the need to inspect the client's high-level messages by the overlay-node for effective mitigation of HTTP(S)-DDoS. Such inspection helps mitigation service providers to detect certain attacks far from the origin server. So, the origin server needs to provide its certificate and private key to be managed by the third-party overlay network operator. Therefore, as discussed in section 1.4, the data communication between the user and the origin server become transmitted through a decryption/re-encryption middle point (i.e., the overlay-node), with encrypted connections on both sides. On the other hand, eliminating split encrypted data communication would limit the attributes utilized by an overlay-node to identify clients remotely.

The objective of the new method described in this chapter is to enable effective mitigation of complex HTTP(S)-DDoS attacks far from the server, while complying with the encryption requirement, and without assuming modifications to existing protocols. Section 4.1.1 explains in more detail the metrics used to define effective mitigation, i.e., mitigation factor, mitigation cost, mitigation time and collateral damage.

To practically resolve the contradiction between the goal and the requirement, enhanced identification is introduced (illustrated in Fig. 3.1). The concept is to add a third-level of client identification to overlay-based mitigation, allowing an overlay-node to group related connections per client. Section 3.2 explains the method's details.

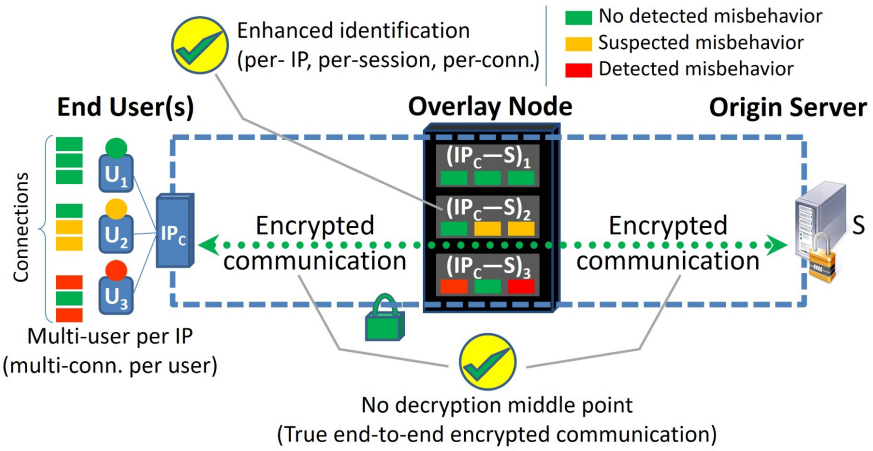



Figure 3.1 By practically enhancing the client identification at the overlay-nodes, the objective is to eliminate the necessity for traffic decryption by the overlay-nodes for effective mitigation of complex HTTP(S)-DDoS attacks remotely.

## 3.2 Proposed Method

### 3.2.1 System Overview

The proposed method utilizes a third level of client identification (i.e., enhanced identification), in addition to the conventional per-IP and per-connection, which we call per-session identification (Fig. 3.2). 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.

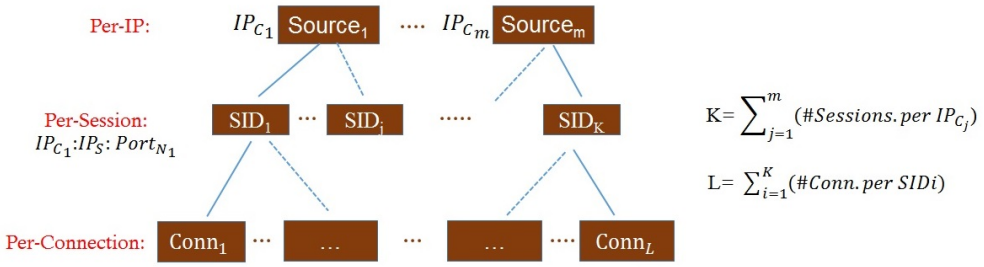



Figure 3.2 Enhancing conventional client identification by adding a third level of identification.

### 3.2.2 Preparation and Communication Stages

Refer to Fig. 3.3 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. 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 (i.e., $SID_1$ = $IP_C$:$SS_X$:$PortN_1$ and $SID_2$ = $IP_C$:$SS_Y$:$PortN_2$). 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. 3.4.2) 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:$PortN_2$/), 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

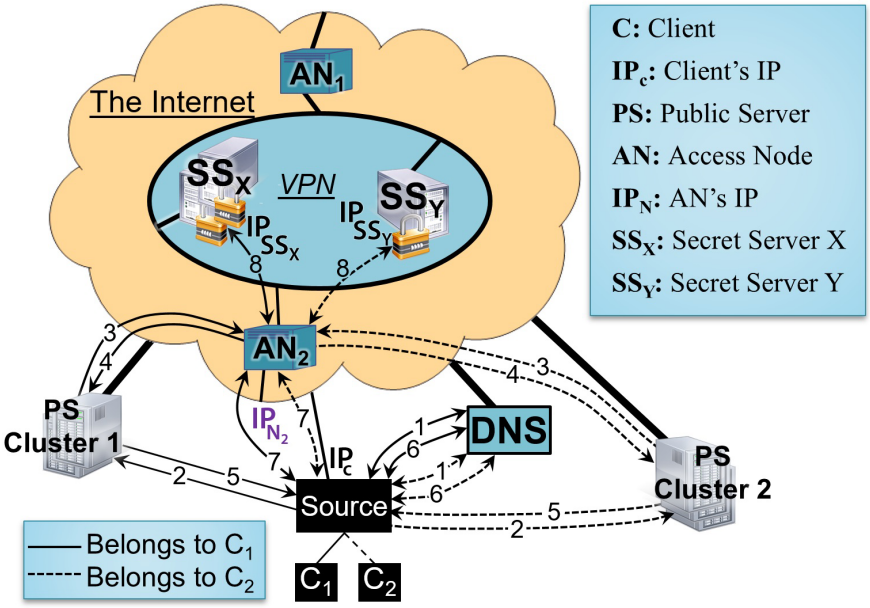



Figure 3.3 Proposed system overview [1–3].

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 5.5 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 (see section 3.3) from affecting the AN (i.e., communication phase), or the SS, effectively mitigating such attacks away from communication paths.

## 3.3 High-level Attack Strategies

As explained in Chapter 1, DDoS attacks are either low-level (e.g., TCP SYN flood) or high-level (e.g., HTTP-DDoS). The work in [1, 34] tackled how the overlay-based PS/AN system can proactively stop low-level DDoS attacks from harming the target

server. Since the SS is only accessible on the high-level, so low-level attacks are by default blocked from reaching the SS, assuming thousands of ANs and PSs in operation. So, the focus in this dissertation is on mitigating the more challenging high-level DDoS attacks. More specifically, the focus is on complex HTTP(S)-DDoS, building on the original method.

Talking about HTTP(S)-DDoS refers to a whole sector of possible DDoS attack categories. All of which need to be taken into consideration, in order to investigate mitigation effectiveness. It's also necessary to describe an attack incident based on each source's behavior strategy, since it's possible to face an attack composed of mixed strategies (also known as a "multivector-DDoS" attack).

From the perspective of client behavior, different HTTP-DDoS attack variations are possible which may vary in practicality and commonness. Each variation is defined by the attacker's decision on various points, such as; connection reacquisition, the requests' rate, their validity, etc.

Existing taxonomies do a good job at classifying DDoS attack types, but they lack the description details which are specific to the method proposed in this dissertation. In our design and experiments, we need a reference where detailed long descriptions of custom attack conditions can be referred to in a single code name. So for simplicity, Fig. 3.4 organizes possible HTTP(S)-DDoS behavior strategies in a compact (non-hierarchical) multidimensional taxonomy, from the AN's and PS's perspectives. This is especially useful also as the number of evaluation experiments grow, where this taxonomy makes it easy to track the evaluation progress.

In Fig. 3.4, the four rightmost dimensions are commonly perceptible by both the PS and AN (as well as other conventional overlay-based mitigation schemes). Thus they are not unique to the proposed method, but important.

From right to left, the first one is how a client behaves after sending a request; does he/she close the connection immediately (fast close), after a delay (delayed close), after any response without reading it (blind wait), or after the successfully waiting for and reading the full response (smart wait). The second point is the rate of creation of; requests, connections, and AN sessions or PS attempts. Thirdly is the request's composition, whether it is a valid, invalid, or incomplete one. The fourth is the requesting behavior per connection, whether it is a single or multiple requests per connection, and of a fixed or alternating parameters.

In addition to the four points above, three more are enabled by the proposed scheme. One unique dimension from the PS's point of view, plus two more unique points from the AN's view. No other content-indifferent overlay-based solution in academia, or in practice, features these additional three unique perspectives.

The AN is capable of identifying behavior further on the "per-session" and (client-server) "per-pair". Per-session, the source behavior has four variations; whether to

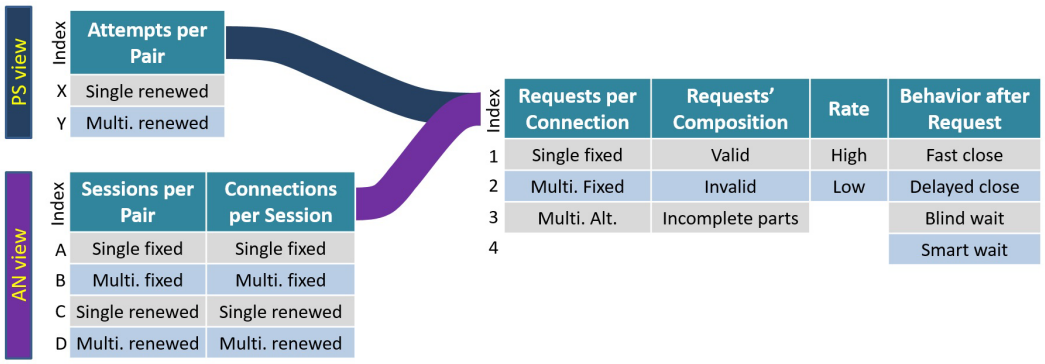



Figure 3.4 Taxonomy of attack source's possible HTTP(S)-DDoS strategies, from the AN's and PS's perspectives, read from left to right. For example, the set of strategies CB32** refer to single renewed session attack, with multiple fixed connections, through which multiple invalid-looking requests of alternating parameters are sent.

open a single or multiple connections at a time, and whether to fix or renew them throughout the attack. Per-pair, the source behavior also has four variations; whether to open a single or multiple sessions at a time, and whether to fix or renew them throughout the attack.

On the other hand, the PS can identify behavior per pair (i.e., has no per-session perception). However, the extra insight featured by the PS arises from the fact that a "client" is expected to send only a single request to the PS before switching to the AN for actual communication. So each request at the PS is counted as a separate "client". This is augmented by the PS's knowledge about which clients actually followed the redirection to the specified AN successfully, and of those which ones behaved well later on, achieved by AN feedback.

Note that the order of the taxonomy's dimensions, and the underlying behaviors, is interchangeable with no effect on the proposed method. Also this taxonomy is readily expandable by considering additional points, such as the type of resource targeted by attack, tool used, etc. However, the interest here is spreading out different thinkable behavior vectors in order to take as a reference in design, development and evaluation phases. For better understanding, below is a demonstration of an example on how to read Fig. 3.4.

For example, consider an attacker with the strategy CB322*. The source opens a single session to one AN, that's not fixed (i.e., renewed if terminated by either side, or after a certain time). Over this session, multiple fixed connections (i.e., the opened connections stay on until closed by the destination, and are never renewed) are opened. Through each connection, multiple requests are sent with invalid-looking requests of alternating parameters. The rates of requests, connections, and sessions

are set below the AN's detection thresholds (i.e., low rate), for example to evade detection. The asterisk indicates an unspecified behavior. In this case, the behavior after a sent request is not specified, i.e. could be either; fast connection termination, delayed termination, waiting for a response and discarding it, or waiting and following its parameters.

For each strategy, there are multiple possibilities of parameters' settings. In this dissertation, the terms scenario, or tactic, are used interchangeably in reference to a strategy with specific settings.

## 3.4 Mitigation

Mitigation of a DDoS attack can be in the form of either the; *prevention*, or *reduction* of the attack's impact on a targeted service. But how to measure the performance of the mitigation system? Will figures of achieved traffic filtration alone suffice to indicate the mitigation effectiveness? Four metrics are considered within the proposed method, for both the PS and AN components.

The first metric used is the mitigation factor (MF) which describes the amount of reduction in attack traffic and the chance of receiving the service. But MF alone with no respect to speed or cost would not be enough. So, additional metrics are utilized for describing the mitigation effectiveness. Mitigation time (MT) is defined as the duration of the mitigation phase, measured from the actual starting time of an attack. As for mitigation cost (MC), it is defined as the increase in service time for a specified file, resulting from the mitigation operations or mitigation measures. Finally, collateral damage (CD) is the inverse effect on the chance of receiving a service for a non-attacking client sharing the same source IP address with an attacking client. More detailed explanation of the considered metrics can be found in Section 4.1.1.

Based on these metrics, Fig. 3.5 (Top) defines three mitigation categories; default prevention (DP), detection-response prevention (DRP), and detection-response reduction (DRR). DP is where full mitigation is achieved (i.e., MF = 1) with zero mitigation time and cost. It's the kind of an obviously undesired behavior. An example for that would be the strategy *****1 (a client not waiting for a response), where DP is expected from a well-scaled secure service provider. On the other hand, DRP has a non-zero mitigation time, and cost for achieving full mitigation. The reducible type (DRR), which is the most difficult to detect, also features a non-zero mitigation time and cost, in addition to a sub-optimum mitigation factor (i.e., MF $<$ 1).

It's desired to maximize the DP and DRP categories, in addition to enabling an acceptable service especially in case of DRR conditions, while maintaining the objectives discussed earlier in section 3.1. For each strategy, it's desired to not only achieve a high mitigation factor, but also have the mitigation time and cost at their

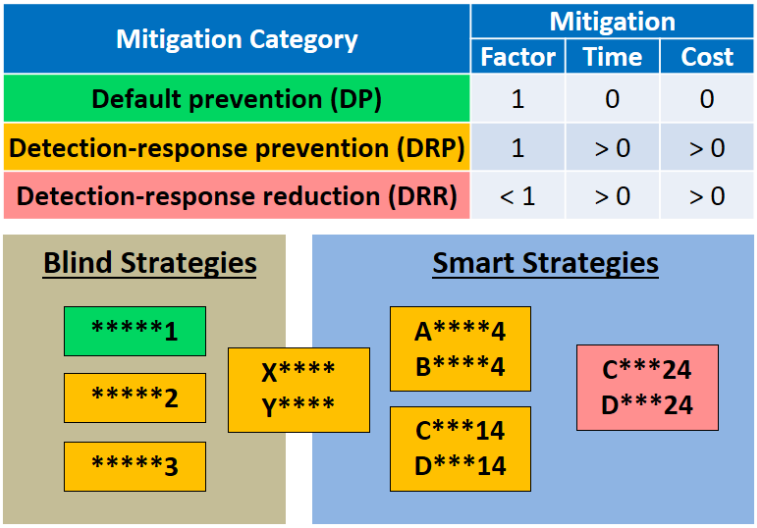



Figure 3.5 Top: Definition of three mitigation categories, with respect to; mitigation factor, time, and cost. Bottom: Expected mitigation category for different HTTP(S)-DDoS attack categories.

lowest.

In Fig. 3.5 (Bottom), the behavioral strategies discussed in section 3.3 are sub-categorized according to behavioral intelligence, into either; smart or blind, where different attack strategies are expected to be DP, DRP, or DRR. The term "smart" strategies is used to describe the ones where the attacker is aware of the defense (i.e., strategies ending in *****4). Any other strategy is referred to as a "blind".

Some smart attack strategies are expected to be preventable after detection (i.e., DRP), while others are only reducible (i.e., DRR). For example, attacks A****4 are expected to be preventable over time because the source utilizes a fixed session for attack, while attacks C***24 may not be completely mitigated because of their session renewal and low rate. Yet they are still reducible over time. Such expectations are put to test through experiments on the implemented proof of concept prototype of the proposed scheme.

### 3.4.1 Detection Concept

The main detection components are distributed over the PS and AN, where each can monitor different behavior attributes.

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. All recorded parameters are time stamped. Every PS and AN component has a reference clock ($w_{ref}$) which is incremented every time step (of duration $\alpha$ [sec]). So the value of the current time step ($w_i$) for an identifier is updated when necessary with the latest $w_{ref}$ value every time this identifier's parameters are accessed. Certain attributes are per time steps (per $\alpha$ [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 $\alpha$, 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 $\beta$ time steps, which enables observing sub-detection-thresholds attacks. Next, we explain the identifiers and attributes with the aid of figures 3.6, 3.7, and 3.8.

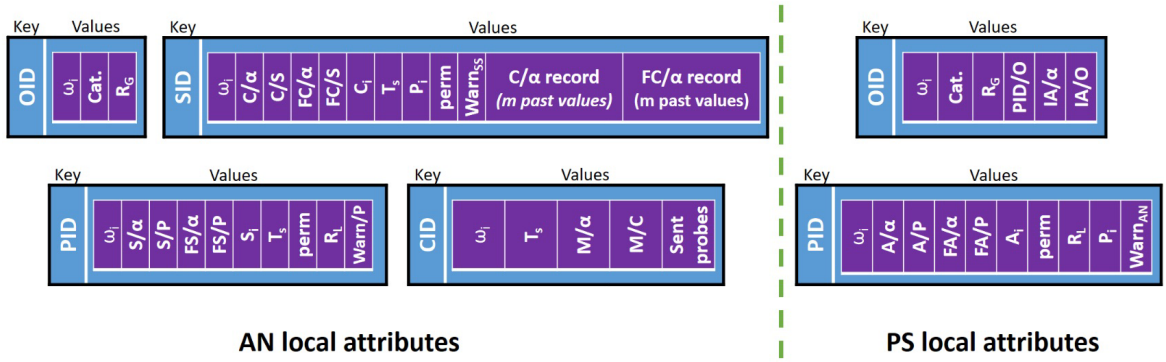



Figure 3.6 Detection attributes stored locally in hash maps by each AN and PS. ANs handle four levels of attributes' values, while PSs handle only two.

| 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 |
| | $S/\alpha$ | Number of sessions created during the current time step α |
| | $S/P$ | Overall number of sessions created under this PID |
| | $FS/\alpha$ | 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 |
| | $C/\alpha$ | Number of connections attempted/created during α |
| | $C/S$ | Overall number of connections attempted/created for this SID |
| | $FC/\alpha$ | Number of failed connections during α |
| | $FC/S$ | Overall number of failed connections under this SID |
| CID | $M/\alpha$ | 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 |

Figure 3.7 Description of stored attributes by 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/O | Overall number of pairs under this OID |
| | IA/$\alpha$ | Number of incomplete attempts per current time step α |
| | IA/O | Overall number of incomplete attempts per OID |
| PID | *A/α* | Number of attempts during α |
| | *A/P* | Overall number attempts for this PID |
| | *FA/α* | Number of failed attempts during α |
| | *FA/P* | Overall number of failed attempts under this OID |
| | *Ai* | Current number of simultaneous attempts for this PID |
| | $R_L$ | Local reputation calculated and stored by the PS for this PID |
| | $P_i$ | Current penalty level for this PID |
| | $Warn_{AN}$ | Number of warnings from AN |

Figure 3.8 Description of stored attributes by the PS.

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 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 $\beta$ consecutive time steps). $S_i$ also helps monitor PIDs that over consume the AN's local ports. Section 5.2 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. 3.4.2). 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 $S/\alpha$ values to monitor a PID's pattern of SID creation, and same for $FS/\alpha$. 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 $C/\alpha$ 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 (Section 3.4.2). The value of $T_S$ is used for local entry cleanup, and to detect a suspicious SID as explained in Sect. 3.4.3.

Finally, a connection identifier (CID) is basically the SID subdivided by source port. For each individual connection, a CID is mapped to $M/\alpha$ and $M/C$, counting all messages transferred from the client to SS, during the current step of $\alpha$ and throughout the connection, respectively. Analyzing the values of $M/\alpha$ 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 $C/\alpha$ within normal, further analyzing its underlying CIDs to observe patterns in the $M/\alpha$ records and $M/C$ can indicate the likelihood of suspicious behavior. It's important 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.

As for the PS, only two identifiers are considered. Consequently, attributes stored by the PS are much fewer than the AN. Per OID, in addition to the category and $\mathrm{R_G}$, the number of pairs per OID (PID/O), incomplete attempts per time step (IA/$\alpha$), and the overall number of incomplete attempts per OID (IA/O), are stored. An attempt that can't indicate the targeted SS is considered incomplete. For example, in case of a malformed or incomplete request. Per PID, the PS records parameters similar to the AN's PID map but in terms of attempts instead of sessions (i.e., A/$\alpha$, A/P, FA/$\alpha$, FA/P, and $\mathrm{A_i}$). Note that here the penalty is enforced per PID, unlike the AN. The flag $\mathrm{Warn_{AN}}$ is raised by the PS in case of a warning message received form the AN.

Additional feedback from the SS is also necessary in certain attacks. For example a very low rate invalid request attack is very similar to normal behavior on the low level and therefore require a high-level feedback from the SS directly, or indirectly, to the AN. Only the current prototype implementation components at the PS and AN are presented in this chapter, while SS feedback and other requirements that should be considered in a real implementation are discussed in chapter 5.

### 3.4.2 Reputation and Penalty

Different identifiers (PIDs, SIDs, etc.) can have different penalties or reputations, depending on the perceived behavior. A penalty is a numerical value that corresponds to a degree of countermeasures enforced by the PS and AN components. The reputation is also a numerical value, but which acts as a multiplier for the penalty. For example, if two clients misbehave, the one with the worse reputation reaches to the maximum penalty faster.

The AN and PS systems update the penalty and/or reputation only in the case of a

| Index | AN Exception | Record | Incident Description |
|---|---|---|---|
| $E_{1.1}$ | Session rate | $\hat{E}_P$ | High rate of new session creation attempts (S/P, S/α) |
| $E_{1.2}$ | Failed session rate | $\hat{E}_P$ | High rate of sessions termination by AN (FS/P, FS/α) |
| $E_{1.3}$ | Connection rate | $\hat{E}_S$ | High rate of new connection establishment attempts (C/S, C/α) |
| $E_{1.4}$ | Failed conn. rate | $\hat{E}_S$ | High rate of connection terminations (FC/S, FC/α) |
| $E_{1.5}$ | Message rate | $\hat{E}_S$ | High rate of new high level messages (M/C, M/α) |
| $E_2$ | High level warning | $\hat{E}_S$ | Warning message from SS |
| $E_3$ | Size exception | $\hat{E}_S$ | Unexpected request size |
| $E_4$ | Connection timing | $\hat{E}_S$ | Client didn't request, or didn't wait for response |
| $E_5$ | Session timing | $\hat{E}_P$ | Client didn't show up at AN (i.e., after preparation phase) |

| Index | PS Exception | Record | Incident Description |
|---|---|---|---|
| $E_{1.1}$ | Attempt rate | $\hat{E}_P$ | High rate of new preparation attempts (A/α, A/P) |
| $E_{1.2}$ | Failed attempt rate | $\hat{E}_P$ | High rate of unsuccessful attempts (FA/α, FA/P) |
| $E_2$ | Multiple request | $\hat{E}_P$ | User sends more than one valid request to PS per single attempt |
| $E_3$ | AN warning | $\hat{E}_P$ | Warning received from AN |
| $E_4$ | Attempt timing | $\hat{E}_P$ | User requests but fails the probing test |
| $E_5$ | Failed request | $\hat{E}_O$ | User connects but sends nothing or an invalid request |

Figure 3.9 Considered AN and PS exceptions.

behavior violation incident (henceforth called exception). The thresholds, separating between what's an exception and what's not, are mapped per PID, so their values are tunable for each server-client pair separately.

Figure 3.9 summarizes the exceptions considered in the current proof of concept prototype implementation for both the AN and PS. The AN system defines two levels of such exceptions; pair level (registered in the record vector $\hat{E}_P$) and session level (registered in the record vector $\hat{E}_S$). $\hat{E}_P$ data is stored per PID to gather information on the underlying sessions' exceptions. $\hat{E}_S$ data, however, is stored per SID and gathers information on the underlying connections' exceptions. On the other hand, the concept of a session is not defined by the PS system. Instead, in addition to the pair level exception, it defines another level of exception, namely the origin level exception (registered in the record vector $\hat{E}_O$).

From the AN's perspective, currently only 9 exceptions are considered; five rate-related ones ($E_{1.1} \sim E_{1.5}$), plus four more that relate to high-level warnings, size, and timing related exceptions ($E_2 \sim E_5$). A session rate exception ($E_{1.1}$) is registered in the AN's $\hat{E}_P$ if the rate of new sessions creation exceeds the set threshold. Similarly, if the rate of failed sessions exceed the threshold, then a failed session rate exception ($E_{1.2}$) is registered also in $\hat{E}_P$. The third, and last, exception that registers in $\hat{E}_P$ is the session timing exception ($E_5$) where the client shows up at the PS but never makes it to the actual communication phase (i.e., doesn't show up at the AN).

In case of a higher-than-threshold rate of client connection establishment (or attempts) per time step, and overall, then the connection rate exception ($E_{1.3}$) is registered in the AN's $\hat{E}_S$. Likewise, a failed connection ($E_{1.4}$) or a message rate exception are registered in $\hat{E}_S$, according to their respective thresholds. In addition, every time a warning signal is received by the AN from the SS, then a high-level warning exception ($E_2$) is registered. In addition, size exception ($E_3$) is reserved for the case of an unexpected size first message, but not currently implemented. Furthermore, if the client

connects to the AN but sends nothing, or fails the probing test, then a connection timing exception ($E_4$) is registered in $\hat{E}_S$ in this case. In the current proof of concept prototype, only $E_{1.3}$, $E_{1.4}$, $E_4$ and $E_5$ are utilized.

From the PS's perspective, 6 exceptions in total are considered; two of which are rate-related, while the rest relate to client timing, AN feedback, and preparation correctness. An attempts rate exception ($E_{1.1}$) is registered to the PS's $\hat{E}_P$ in the case of a PID committing to the preparation phase with a rate higher than the threshold. Also, if the rate of unsuccessful attempts is high, then a failed attempts rate exception ($E_{1.2}$) is registered in $\hat{E}_P$. This is the PS, and by design a connection from any client is not expected to deliver more than one request. If more than one request is sent by the client, then the PS system registers a multiple-requests exception ($E_2$) in $\hat{E}_P$. In addition, even if the client requests only once as normal and has no warnings feedback from the AN, it still has to pass the probing test by the PS. If the client fails the test, then an attempt timing exception ($E_4$) is registered in $\hat{E}_P$. Finally, if a client connects but sends nothing, or sends and invalid message to the PS, then a failed-request exception ($E_5$) is registered in the PS's $\hat{E}_O$.

Record vectors $\hat{E}_S$, $\hat{E}_P$, and $\hat{E}_O$, each is of length $\beta$ bits storing exceptions' data occurring in the past $\beta$ time steps. Figure 3.10 illustrates with example how record

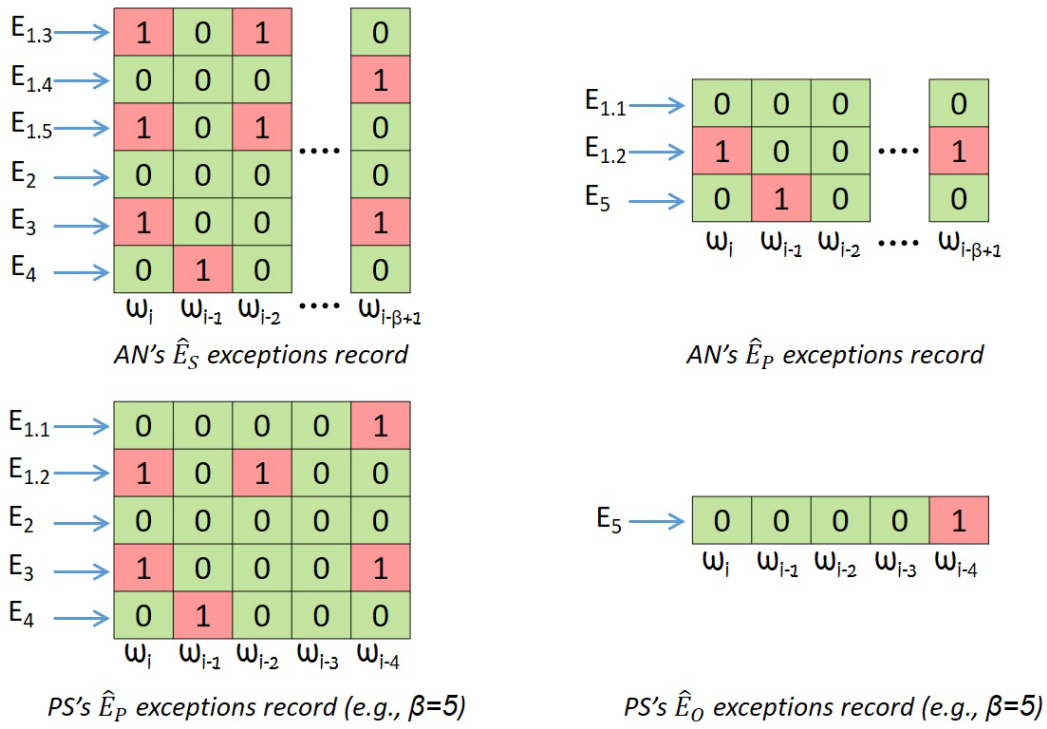



Figure 3.10 Each record vector registers the underlying exceptions during the latest $\beta$ time steps.

vectors are updated by the AN and PS systems. Records are not updated every time an exception incident is triggered. Instead, record update takes place only once per time step. So, if more than one exception take place during the same time step $w_i$, then only a single registration in the related vector is performed. A value of '0' indicates an exceptions-free time step while '1' indicates otherwise. Only $\hat{E}_O$ is single-dimensional, so obtaining its value is straightforward. For example, an all '1's $\hat{E}_O$ (i.e., $\hat{E}_O = max$) indicates the detection of at least one origin-level exception in all of the recorded past time steps. As for the other two ($\hat{E}_P$ and $\hat{E}_O$), they are multi-dimensional, requiring some pre-processing before being able to utilize their values. Considering each dimension separately is ideal for detection. However, for now, the implementation is simplified by combining all dimensions into one by passing them through a bitwise logic OR function.

These records are fed into the penalty and reputation update functions, which in turn determine the level of response by the AN and PS systems. The number of exceptions are not used directly to update the penalty value. Instead, only the exception record vectors $\hat{E}_S$, $\hat{E}_P$, and $\hat{E}_O$ are utilized, in addition to the reputations' values. Figure 3.11 explains the general relation between the recorded exceptions and the penalty and reputation for both the AN and PS systems. So for the AN, the local reputation is a function of $\hat{E}_P$ and $\mathrm{R_G}$, and is enforced on the PID level (i.e., each PID has its own value, which affects all the underlying SIDs). As for the AN's penalty, it's a function of $\hat{E}_S$ and $\mathrm{R_L}$, and is enforced on the SID level. On the other hand, the PS system evaluates its local reputation as a function of the received reputation from the AN ($\mathrm{R'_L}$), $\mathrm{R_G}$, and $\hat{E}_O$. While the PS's penalty is a function of $\hat{E}_P$ and $\mathrm{R_L}$. Note that, unlike the AN, the PS enforces the penalty on the PID level.

To recap, each of the PS and AN has its own locally enforced penalty that is computed as a function of the reputations, local and global, in addition to the recorded behavior exceptions. In a future iteration of the developed prototype, the value of $\mathrm{R_G}$ is processed centrally and shared by the mitigation service provider, and is a function of the aggregated information gathered form different, distributed, ANs and PSs. As for the current implementation, $\mathrm{R_G}$ is not enforced (i.e., its value is fixed to 1).

To understand the different states of reputation and penalty, their update and transitions, see figures 3.12 and 3.13.

Figure 3.12's LHS explains the possible state transitions of the AN's local reputation as a function of $\hat{E}_P$ only (i.e., $\mathrm{R_L}(\hat{E}_P)$), since $\mathrm{R_G} = 1$. In the current implementation, $\mathrm{R_L}(\hat{E}_P)$ has three possible levels; normal ($L_1$), suspicious ($L_2$), and bad ($L_3$). At the start of any PID, $\hat{E}_P = 0$ and $\mathrm{R_L}(0)$=$L_1$, as long as no pair-level exceptions take place. In the case of an exception $E_{1.1}$, $E_{1.2}$ or $E_5$ is triggered, separately or combined, the exception is registered in $\hat{E}_P$ (i.e., $\hat{E}_P \neq 0$), and the new state of $\mathrm{R_L}(\hat{E}_P)$ is suspicious ($L_2$). As long as $0 < \hat{E}_P < max$, the middle state remains unchanged. If $\hat{E}_P = max$ then $\mathrm{R_L}(\hat{E}_P)$ enters the bad state. Even in the case of a bad reputation, a

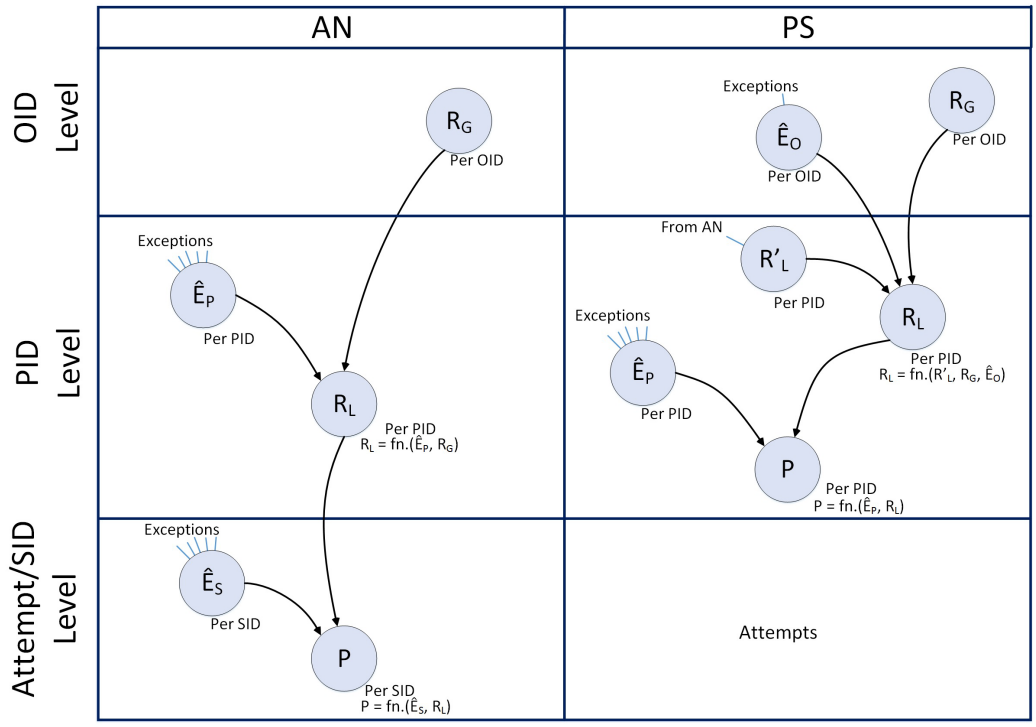



Figure 3.11 Reputations and penalties hierarchy. In current prototype implementation, $\mathrm{R_G} = 1$.

PID is still granted service. So basically $\hat{E}_P$ controls the reputation's state transition up and down. Finally the value of $R_L = max\{\mathrm{R_G}, \mathrm{R_L}(\hat{E}_P)\}$, in case $\mathrm{R_G}$ is enforced.

In the current prototype, the discussed three levels of $\mathrm{R_L}$ are represented numerically as below. Next, we explain how these values are used to affect the penalty update.

$$\mathrm{R_L}(\hat{E}_P) = \begin{cases} 1 & , \quad if\ \hat{E}_P = 0 \\ 2 & , \quad if\ 0 < \hat{E}_P < max \\ 3 & , \quad if\ \hat{E}_P = max \end{cases} \tag{3.1}$$

The AN's penalty $P(\hat{E}_S, \mathrm{R_L})$ is enforced per SID and is a function of $\hat{E}_S$ and $R_L$. Figure 3.12' RHS explains the penalty $P$'s possible state transitions. 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 (Section 3.4.3) is directly proportional to the value of $P_i$. Note that an SID can enter the differential

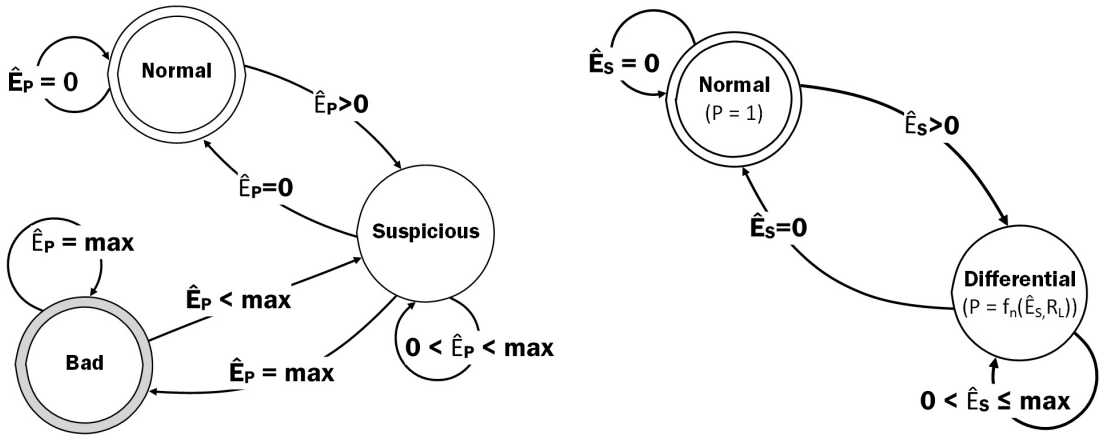



Figure 3.12 AN exceptions handling in the current prototype, given $\mathrm{R_G} = 1$. LHS: Local PID reputation ($\mathrm{R_L}(\hat{E}_P)$) states. RHS: SID penalty ($P(\hat{E}_S, \mathrm{R_L})$) states.

state with $P_i = P_{max}$ without being a failed session yet, 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.

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 $\gamma$ 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 $\gamma$ and $\mathrm{R_L}$.

Figure 3.13 (LHS) explains the possible state transitions of the PS's local reputation as a function of $\hat{E}_O$ and R'$_\mathrm{L}$ (i.e., $\mathrm{R_L}(\hat{E}_P, \mathrm{R'_L})$), given $\mathrm{R_G} = 1$. In the current implementation, $\mathrm{R_L}(\hat{E}_P, \mathrm{R'_L})$ has two levels; normal ($L_1$) and suspicious ($L_2$). At the start of any PID, $\hat{E}_O = 0$ and $\mathrm{R_L}(0, 1){=}L_1$ (assuming R'$_\mathrm{L} = 1$), as long as no origin-level exceptions take place. In the case of an exception $E_5$ is triggered, it is registered in $\hat{E}_O$ (i.e., $\hat{E}_O \neq 0$), and the new state of $\mathrm{R_L}(\hat{E}_P, \mathrm{R'_L})$ is suspicious ($L_2$). Same transition takes place in case of an R'$_\mathrm{L} > 1$. If $\hat{E}_O \neq 0$ or R'$_\mathrm{L} \neq 1$, then the suspicious state remains unchanged. Service is still granted to PIDs with suspicious reputation. So basically $\hat{E}_O$ and R'$_\mathrm{L}$ control the reputation's state transition up and down. Finally the value of $R_L = max\{\mathrm{R_G}, \mathrm{R_L}(\hat{E}_P, \mathrm{R'_L})\}$, in case $\mathrm{R_G}$ is enforced.

The PS's penalty $P(\hat{E}_P, \mathrm{R_L})$ is enforced also per PID and is a function of $\hat{E}_P$ and $R_L$. Figure 3.13's RHS explains the PS's possible penalty state transitions, where it has only two states; normal ($P(0, 1) = P_{min}$) and differential ($P_{min} < P(\hat{E}_P, \mathrm{R_L}) \leqslant P_{max}$). Normal state is where the $\hat{E}_P$ record vector is all zeros (i.e., no pair-level exceptions

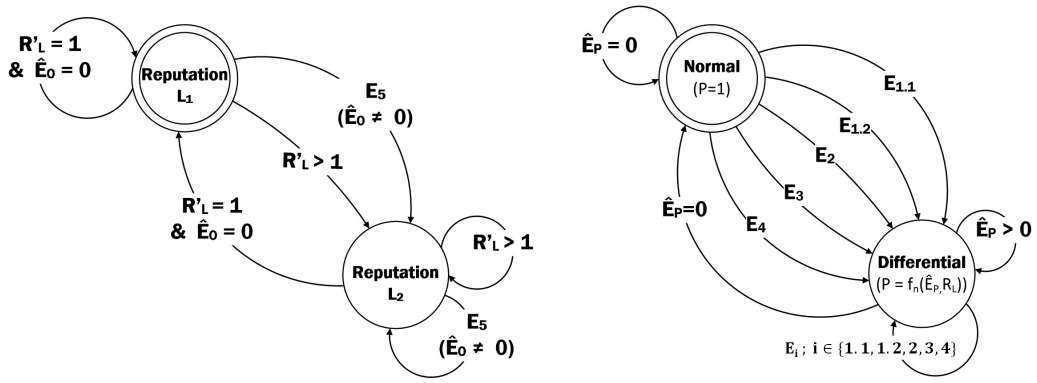



Figure 3.13 PS exceptions handling, given $\mathrm{R_G} = 1$. LHS: Local PID reputation ($\mathrm{R_L}(\hat{E}_O, \mathrm{R'_L})$) states. RHS: PID penalty ($P(\hat{E}_P, \mathrm{R_L})$) states.

have been detected in any of the recorded past time steps), and where $\mathrm{R_L} = 1$. In any case of a pair exception of the type $E_{1.1}$, $E_{1.2}$, $E_2$, $E_3$ or $E_4$, then $\hat{E}_P$ is updated (i.e., $\hat{E}_P \neq 0$) and the service enters into the differential state. During this state the penalty's value ranges from $[P_{min} + 1, P_{max}]$ and determines the PS's response level (section 3.4.3).

### 3.4.3 Attack Countermeasures

Attack Countermeasures (CM) can be either; proactive (i.e.; DP) or reactive (i.e.; DRP or DRR). In either case, the prevention or reduction is part of the AN and PS, far from the SS.

In the proposed scheme, both the AN and PS, by default, are equipped with a new overlay-based pre-service verification method that requires no traffic decryption, uses standard unmodified protocols, and requires no special downloads by the client or installations at the server. It acts as a first degree countermeasure, by probing each client connection with early 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 don'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 every HTTP (i.e., 'H', 'T', 'T', 'P', and '/') or HTTPS (i.e., $22_{10}$ and $3_{10}$) first response. This way, the probing actions are transparent to both the clients and servers. Since the system aims at HTTPS more, so we set the maximum number of probes to 2 for both protocols. For example, if two HTTP probes are sent in response to the client's first message, the first single-byte probe would contain 'H' and the second probe is 'T'. As illustrated in Fig. 3.14, if the client acknowledges the PS's probes without disconnecting or sending a second message, the PS connects to the AN as normal. Otherwise, the AN is not affected. On the other hand, the AN sends the probes and if the client acknowledges

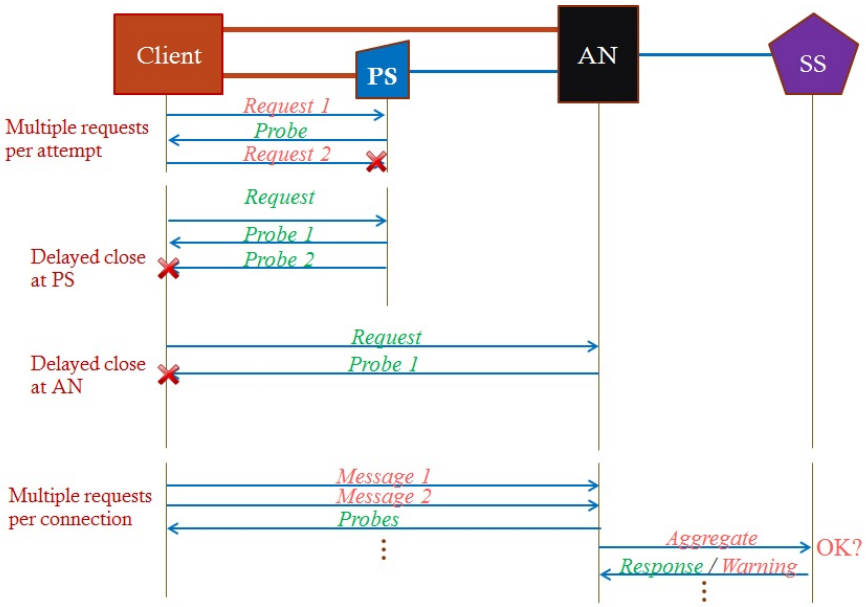



Figure 3.14 Pre-service slow-response probing.

them without disconnecting then the AN transfers the client's message(s) to the SS, then omits the initial two bytes from the server's response when it arrives since they are already delivered to the client during probing. At the AN, the client may send multiple messages, which are aggregated by the AN and transferred to the SS in bulk. This would help simplify supplementary local detection if the series of high-level messages can be analyzed locally at once before committing actual resources, instead of one by one.

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 Figure 3.15. This type of response helps prevent part of the blind attacks (i.e.; *****1), and enables effective mitigation of part of the blind strategies (i.e.; *****2 and *****3) that don't wait long enough for the probing phase to finish. This is true because any blind attack tool can't get beyond the preparation phase, even if it waits for a response. So the SS's resources are conserved.

Additionally, second degree countermeasures are in place at the AN and PS to counter the remaining strategies that evade the first degree countermeasure.

As for the AN's second degree countermeasure, the AN analyzes the current and past values of $C/\alpha$ and $FC/\alpha$. If the analyzed values show a suspicious pattern, the related SID is placed in "jail" for a single time step while observing its latest $C/\alpha$ and $FC/\alpha$. During jail time, all the jailed SID's new connection attempts are accepted, then immediately closed by the AN while registered as one attempt. In addition, $R_L$ is

| Penalty | No. of 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 |

Figure 3.15 Function of $P_i$; the lookup table manually set for the duration before, between and after the slow-response probes.

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. For example, as shown in Fig. 3.16, although $SID_2$ is behaving below the set detection threshold (from the perspective of C/$\alpha$), its past record of C/$\alpha$ attribute would suggest an abnormal behavior. Note that C/$\alpha$ is a unique attribute for the proposed method, since it's difficult to remotely identify which new connection attempts are related using the conventional two-level identification, especially without traffic decryption. So, a simple analysis algorithm is employed in the current prototype. An SID is sent to jail if $\frac{C/S}{w_i - T_s} \approx \mathrm{C}/\alpha_i$, $\sum_{j=1}^{m} \frac{\mathrm{C}/\alpha_{i-j}}{m} \approx \mathrm{C}/\alpha_i$, or $\sum_{j=1}^{m} \frac{\mathrm{FC}/\alpha_{i-j}}{m} \approx \mathrm{FC}/\alpha_i$, where $\alpha_i$ is the current time step and $m$ is the length of considered past behavior record where $m \leq \beta$. In the latest prototype implementation, assuming that a normal client is not likely to generate the same number of connection attempts per time step for 4 consecutive times, $m$ is set to 4 to demonstrate the concept.

The PS is also equipped with its own second degree countermeasure against slow-requesting HTTP-DDoS, which is the worst case scenario for the PS. The reason is that in case of an attack on the PS, other attack categories are less effective. For example, a single-request per-connection attack release the PS's resources faster as it repeats the connection step 2, while a multiple-request per-connection attack are detectable promptly by the PS (since only 1 request per connection is expected). So, in the worst case the attacker knows this is the likely most effective HTTP-DDoS attack category against the PS. 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.

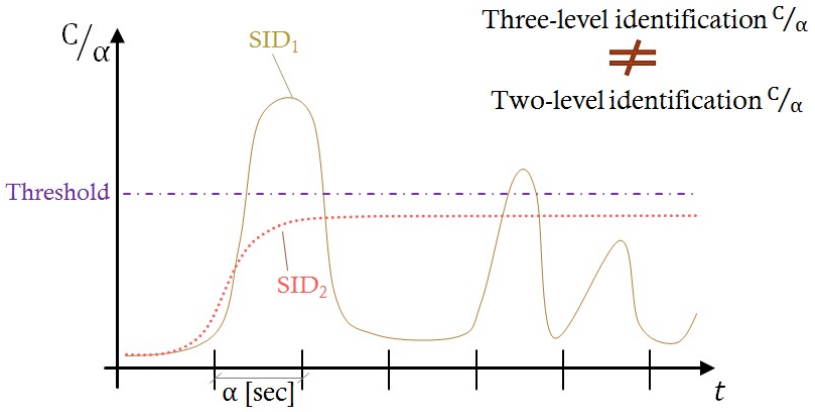



Figure 3.16 Example of behavior record analysis countermeasure.

From the PS's perspective, the request must be collected as a whole before contacting the AN. Therefore an invalid request counts as a failed or incomplete attempt, further contributing to the client's penalty. So, a slow-requesting HTTP-DDoS attack towards the PS is not expected to affect the AN. But, 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.

In the current proof of concept prototype, a simplified countermeasure against slow-requesting HTTP-DDoS targeting the PS and the system is included. Assume a client that connects at a time reference $t = T_C$, as in Fig. 3.17. 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 $T_n = T_1$.

The case with the AN is different, where probing starts after $T_{a(max)}$ regardless of the request's content, since no packet payload inspection at the AN is assumed. So, the AN's $T_{a(max)}$ should be increased for SIDs with $\text{Warn}_{\text{SS}}$ flags, and their reputations and penalties updated by traffic attributes gathered during this duration.

It's also possible to equip the system with additional degrees of countermeasures. For example, by utilizing records of $S/\alpha$ and $FS/\alpha$. In addition, a supplementary local detection component at the SS premise would also be required as discussed in section 5.7 for complete mitigation. The load on this detection is inversely proportional with the mitigation factor offered by the overlay nodes. The aim of this overlay-based

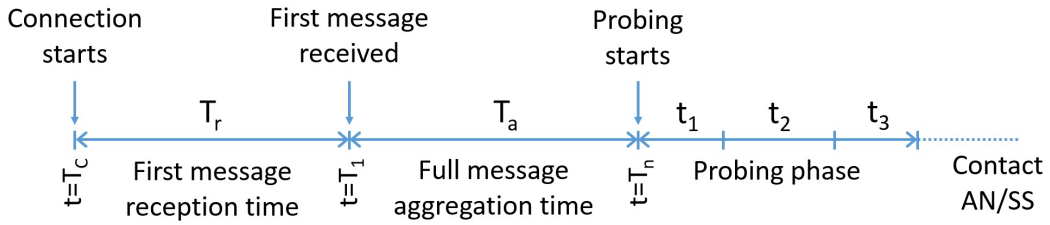



Figure 3.17 Pre-probing message reception and aggregation.

method is to enable high mitigation factors by the AN and PS, thus reducing by the same factor the load on supplementary detection.

Several experiments are conducted to investigate the mitigation effectiveness enabled by the added enhanced identification and attack countermeasures under complex HTTP(S)-DDoS attack strategies. Seven of which are discussed in the next chapter.

# Chapter 4

# Evaluation

## 4.1 Evaluation Method

To demonstrate the soundness of the proposed concept, and to show the potential from investing in it, an evaluation method is required that describes performance under complex attack conditions. Yet, conducting live experiments with DDoS traffic is too destructive and difficult to take place openly on the Internet. Therefore, the usual evaluation methods in related research on DDoS mitigation are either of three options, namely; theoretically, by software simulation, or by emulation (within a controlled environment).

Theoretically proving the efficiency of a proposed mitigation scheme can't describe how DDoS traffic affects hardware and network protocols. For that, experimentation is required with generated attack traffic. Although simulation tools can mimic experimental setups, yet software simulators can't produce realistic results either when it comes to DDoS evaluation. Consider for example the popular ns-2 simulator [65]. In it, Internet forwarding devices (i.e., switches and routers) are modeled only at a high level. The ranges of parameters in commercial forwarding devices are not incorporated [66]. Another popular simulator such as OPNET [67] is more detailed than ns-2. OPNET have detailed models of routers, switches, protocols, etc., based on vendor specifications. However, even these detailed models' parameters such as buffer sizes and forwarding rates are hard to tune to mimic real router behavior [68].

On the other hand, using real routers such as PC, Click, Cisco, etc., provide realistic forwarding behavior. So, testing a DDoS defense under emulated attack conditions on a scaled-down network environment presents a more realistic evaluation option. This requires the development of a prototype of the proposed concept to be deployed on actual network nodes. Therefore, the choice from the start of this research is to proceed with the development and emulation option.

### 4.1.1 Performance Metrics

Performance metrics provide a quantitative description of the mitigation effectiveness. Evaluation based only on the amount of attack bandwidth reduction as a metric is more suitable for low-level volumetric attacks. It can't describe the impact of a high-level attack on the server, or the speed of the mitigation system. The reason is that high-level DDoS attacks with low bandwidth, even after being reduced by a mitigation service, can still result in significant impact on the service.

Therefore, this research adopts a comprehensive set of metrics for evaluation. We describe the four metrics considered in this research to measure the effectiveness of mitigation, namely; mitigation time (MT) mitigation cost (MC), mitigation factor (MF), and collateral damage (CD).

MT, which is the time to mitigate the attack, i.e., mitigation phase, 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. The smaller the value of MT the better. However, even a desirably small duration of MT doesn't describe the attack's impact on the service during and after that duration.

Therefore, 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 corresponding 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 $i$. Ideally, MT and MC are at their lowest possible values.

In addition, the mitigation factor ($MF_i$) is defined as the product of the attack traffic reduction factor ($RF_i$) multiplied by the chance of service completion ($CoS_i$) under attack for each time step $i$, so that:

$$MF_i = RF_i \times CoS_i \tag{4.1}$$

$RF_i$ is the amount of reduction in high-level attack traffic achieved by the enforced countermeasures within a specific time window, and it's calculated as follows:

$$RF_i = 1 - \frac{\widehat{Att_i}}{Att_{\text{pub}}} \tag{4.2}$$

$CoS_i$ comprehensively measures the chance of getting a complete service through the PS and AN. The service is complete if a sent request fully returns the desired

resource from the SS. It's defined as:

$$CoS_i = \frac{Resp_i}{Req_i} \tag{4.3}$$

Where; $Att_{\text{pub}}$ 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.

In case of the PS-targeting attack, there is no attack traffic at the AN-SS side (i.e., $\widehat{Att_i} = 0$). So, $MF_i$ for the PS simply becomes;

$$MF_i = 1 \times CoS_i \tag{4.4}$$

Finally, for each time step $i$, the collateral damage (CD) metric aims to measure the selectivity of the system's response in case of a multi-behavior source IP. It's defined as the inverse effect on CoS for a non-attacking client sharing the same source IP address with an attacking client, and is computed as follows;

$$CD_i = 1 - CoS_i \tag{4.5}$$

This set of metrics serve as a comparison basis for the effectiveness of different mitigation methods. Therefore, an effective mitigation method is expected to show; low MT, low MC, high MF, and low CD. Section 5.3 discusses the considered metrics further.

To obtain these values, traffic measurements and implemented nodes' log data are gathered during each experiment. Traffic measurements and log data are recorded in four locations. The first two locations are via custom tool that we specifically developed for measurements.

The first location is at a measurements node (MN), which emulates 100 different measurement sources (each with a unique source IP address), each generating a single request per time step. The 100 HTTP requests (or HTTPS, depending on the experiment) for a specified resource file per time step are spread throughout the time step. For example, for a 30 [sec] time step, the measurements rate would be roughly 3.33 requests per second. For each request, its success is registered if its full life-cycle is completed (i.e., starting from the preparation phase till the whole response is delivered to the client from the SS). Otherwise, it registers as a failure. In addition, the preparation time (at the PS) is measured as well as the whole service time for the requested file. So, at each time step we have three sets of 100 raw data values (i.e.,

service success set, the whole service time set via PS + AN, and the actual communication time set via AN only) collected by the measurement tool. Eventually, from the recorded measurement raw data sets, all the mitigation performance metrics are obtained for each time step. Evaluation data are plotted versus time in terms of the average point and a 95% confidence interval (CI).

The second location is also via a custom developed measurement tool, like the MN, but this time is deployed on one of the attack sources, sharing its IP, to observe the collateral damage as observed by a non-attacking user sharing an attack IP.

Additionally, the two remaining locations are at the AN and PS. The traffic volume at both is recorded in terms of number of TCP connection attempts and HTTP (or HTTPS) requests. This way, it's possible to tell, at any specific time step, what was the actual whole rate of the DDoS attack, in terms of low-level requests (i.e., TCP connection attempts) and high-level requests. From the collected data, we can also observe the service time (to measure MC), the time taken for mitigation (i.e., MT), as well as the impact of the proposed response methods on the attackers' achievable rate (i.e., the MF).

### 4.1.2 Emulation Platform (Testbed)

For emulation, a researcher has two options. Either construct a dedicated testbed in the lab or utilize a shared testbed. The latter option is more favored for openness to other researchers, scale, as well as the traffic interaction with cross-traffic from other testbed users. Two popular shared testbeds exist today that are mentioned in the DDoS mitigation literature; PlanetLab [69] and DeterLab [37].

Planetlab's system counts on participating organizations to contribute machines (currently more than 1353 nodes). Users gain shared access to those nodes via virtual machine software that achieves user isolation. Planetlab's nodes therefore can be organized into overlays. Also, traffic between nodes travels across the Internet and experiences realistic delays, drops and interaction with cross-traffic.

However, the choice of which shared testbed to utilize is also governed by the nature of the experiments to be conducted. On PlanetLab, researchers can install packages on the testbed's nodes, but their choice of operating system (OS) and the granted privileges on nodes are restricted. Also, launching disruptive attacks, which is the nature of most DDoS attacks, is strictly prohibited on the Planetlab testbed.

On the other hand, with DeterLab, the researcher can gain exclusive root access to a desired number of nodes, load the desired OS, and have root privileges. Also, massive DDoS attacks (i.e., "Risky Experiments", as labeled by the DeterLab team), are acceptable on the Deterlab testbed. Experiment traffic also experiences interaction with cross-traffic from other active experiments. In addition, outside connectivity of

the test traffic with the public internet is an option. So, DeterLab is the chosen platform to be incorporated in this research for the evaluation of the proof of concept prototype. Mainly due to the disruptive nature of the traffic generated, the testbed's flexibility, and to simplify repeatability.

### 4.1.3 Evaluation Plan

As explained in Chapter 1, the goal of this research is to enable effective overlay-based mitigation of complex HTTP(S)-DDoS attacks, while complying with the true-end-to-end encryption requirement. Section 4.1.1 explained the metrics to measure effective mitigation. In this section, we describe the plan to evaluate this goal.

But what's meant by complex HTTP(S)-DDoS attacks? As discussed in section 3.3, there can be multiple perspectives that define an HTTP(S)-DDoS attack category. Also, variations in attack conditions within even the same attack category may be endless. Arguably however, not all possible attack conditions are equal in terms of commonness (i.e., per recent surveys), damage (i.e., effect on service availability), and complexity (i.e., detection difficulty). A complex attack is one that's conventionally difficult to detect. For example, a single-request-per-connection type of attack is intuitively harder to detect by conventional methods than a multi-request-per-connection attack. Similarly, a low-rate of attack requests per source (i.e., below detection thresholds) is also intuitively harder to detect than a high-rate of attack requests per source. Same can be said about a knowledge-based attack (i.e., attacker knows the mitigation configurations) versus an otherwise blind attack.

So, several assumptions are made for the planned evaluation. Firstly, attackers can learn about the PS's redirection and eventually adapt the attack tools to obey the preparation stage and then attack through the AN. Likewise, attacks below the AN's set detection thresholds are to be anticipated. Secondly, effective mitigation against a knowledge-based HTTP(S)-DDoS attack, especially with simplified mitigation measures, also suggests effective mitigation against a less complex attack category. For example, if the implemented prototype AN shows a high MF and low MC against a single-request-per-connection type of attack, that would suggest similar results against a multi-request-per-connection attack. Thirdly, all attacks against a web server behind the proposed method utilize HTTP(S)-DDoS methods. Stopping other possible types of cyber-attacks that may target the PS or AN are beyond the scope of this research.

Therefore, the proof of concept prototype is tested under several HTTP(S)-DDoS attack conditions to evaluate the soundness of the concept. Among the conducted experiments throughout the course of this research, seven experiments are discussed in section 4.2 considering complex attack conditions, some conditions of which are missing from related research on DDoS mitigation.

Four experiments against the AN are discussed. The first addresses the high-rate HTTP-DDoS attack. Each attack source is sending HTTP GET requests at its maximum achievable rate. Although such category may not be hard to detect, it's important to measure the MC of mitigating such a brute force category of attack before proceeding to the more complex categories. In the following experiment against the AN, we assume that the attacker learns that the AN doesn't inspect the content. So, the attacker manually adapts an off the shelf slow-requesting HTTP-DDoS attack tool to attack via the AN, yet with attack rate higher than the detection threshold. Then, the case is investigated where the attacker adapts further by increasing the number of attack sources, switching to encrypted HTTP-DDoS (i.e., HTTPS-DDoS), sending a single request per connection to evade conventional per-connection detection methods, and significantly reducing the attack rate per source to below the AN's set detection thresholds (i.e., low-rate, single-request per-connection, sub-detection-threshold HTTPS-DDoS attack). Therefore, assuming possible attacker's knowledge of the mitigation details. Finally, we examine the case where the attacker combines two complex HTTP(S)-DDoS attack categories simultaneously, which is also known as a multi-vector DDoS attack.

In addition, three experiments with attacks on the PS are discussed. 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-request per-connection attacks are detectable promptly by the PS (since only 1 request per connection is expected). Same thing with malformed requests. So, the three PS-targeted experiments in section 4.2 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 about other attack conditions in section 5.4. The first experiment is conducted with 20 high-rate attack sources. Then we increased the number of attack sources to 2,000 in the following experiment, with similar high-rate attack conditions per source. Further, the third experiment expands the slow-requesting attack population to 10,000 unique attack sources with low-rate per source this time. The assumption is that the failed attacker may attempt to spread the requests over larger number of sources to evade detection.

Although attacks that are several hours long were a common occurrence in earlier years [7, 70], recently the majority (93%) of the reported HTTP(S)-DDoS attacks are shorter than 1 hour in duration [8]. Therefore, all the attacks considered in the experiments reported in this dissertation are considered long-duration attacks.

In section 4.2, experiments are not ordered in terms of the type or conditions of attack. They are rather sorted in the chronological order of their execution (i.e., from older to the most recent).

The experiments' topologies are a scaled down approximate version of the real situation. Also, virtualization is utilized in some of the experiments to multiply the

number of attack sources given a finite number of machines. Therefore, spreading out the attack population. Downscaling a topology while keeping reasonable fidelity is an open research problem [66]. Section 5.3 discusses the evaluation decisions further, as well as anticipated additional attack categories and their variations.

Several tools are used for generating the attack traffic. For the high-rate HTTP-DDoS, a blind attack tool called LOIC [71] is utilized because of its commonness [72]. For the low-rate HTTP(S)-DDoS controlled attack conditions, a new custom DDoS tool is developed and utilized. As for the slow-requesting HTTP-DDoS, three attack source types are utilized. First, the publicly available tool commonly used by attackers, called Slowloris [73]. In addition, a new custom slow-requesting attack tool was also designed and developed that generates the same category of attack, but with more attack parameters' control. Finally, a custom tool was built that utilizes the original Slowloris code to adapt to the proposed mitigation scheme. Experiment specific attack conditions are described within their respective experiments.

In addition, we qualitatively test the prototype's transparency (i.e., operation without special requirements form clients or servers). So, before testing the prototype system under DDoS attacks, several non-attack tests with actual commercial websites were conducted to observe the transparency of the proposed method, using commonly used web browsers. The prototype is set up in the lab with access to the Internet and real browsing by the researcher. The problem with this lab setup is that the browser prompts the user with a warning message about the servers' certificate. Yet, such warning would not exist in the real deployment, since the web server should issue a wildcard certificate to work seamlessly with all ANs.

### Test Web server Benchmarks

Before deploying the mitigation system, direct benchmarking measurements on the Apache web server to be used as SS are conducted (i.e., before hiding the SS). 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, and with no request concurrency, the non-attacked server showed a local limit of 54.6 [r/s] (i.e., requests per second) in average. In addition, a 19.6 [ms] service time is observed when tested with ApacheBench (version 2.3). Testing with with different concurrency settings, the non-attacked server showed nearly the same local limit of requests per second. But as the concurrency increases, so does the measured service time as shown in Figure 4.1. The utilized unmodified Apache server versions 2.2.14, 2.2.22 and 2.4.7 showed the same benchmarks.

Note that this limit can be raised by tuning the server's configuration. However, the focus of this research is on evaluating mitigation of attack far from the server. So, that limit is unchanged in all experiments.

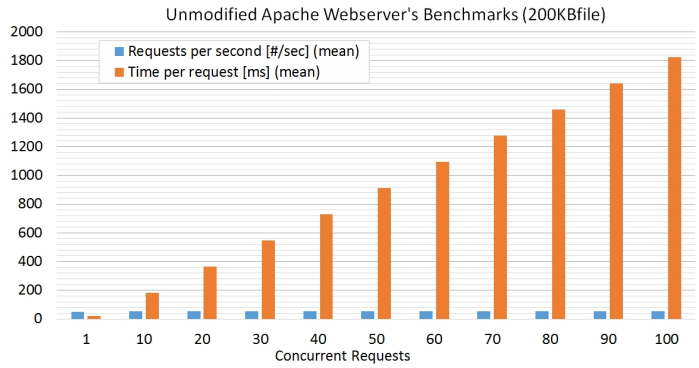



Figure 4.1 Apache web server benchmarks using ApacheBench for a 200 KB file.

### 4.1.4 System Parameters

Three types of parameters define each experiment's setup; prototype implementation, testbed specifications, and attack parameters. For the current proof of concept prototype implementation, we test with manually set non-optimal values as follows. Selecting a large value of $\beta$ would result in richer behavior records while a small value simplifies the system. From experience, the value of $\beta$ is set to 8 in all experiments, being neither too large nor too small. Also, during prototype development, we tested with several values of $\alpha$ between 10 to 60 seconds. We learned that selecting a large value of $\alpha$ results in a longer mitigation time while a small value may limit the collected attributes. Accordingly, in the discussed experiments the value of $\alpha$ is within 30 to 40 seconds (as described in each experiment). In addition, the set range of $\alpha$ is considered when setting the per-$\alpha$ thresholds. Also, it's assumed that browsers normally set a small limit on simultaneous connections per server [74]. So, the AN's default threshold values are set as follows; $S/\alpha\big|_{Thd} = 20$, $C/\alpha\big|_{Thd} = 20$, $M/\alpha\big|_{Thd} = 10$, $FS/\alpha\big|_{Thd} = 2$, and $FC/\alpha\big|_{Thd} = 2$, while other AN's thresholds are not enforced. The PS's default threshold values are set as follows; $A/\alpha\big|_{Thd} = 20$, and $FA/\alpha\big|_{Thd} = 2$, while other PS's thresholds are not enforced. Such parameters are tunable for better results. Also $R_G$ is normalized in the current prototype version. The values of $T_{r(max)}$ and $T_{a(max)}$ are set to 2 and 3 seconds, respectively. As for $\hat{E}$, only the 3 most significant bits are considered, and different exceptions are combined. Testbed specifications, attack parameters are described in their respective experiments. We discuss further about the prototype parameters in Sect. 5.3.

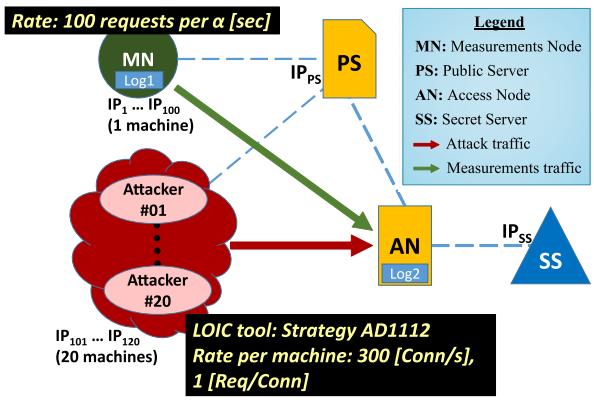
In addition, general Linux kernel parameters are modified from their defaults to accommodate the large number of concurrent threads and connection attempts. As follows; *net.ipv4.ip_local_port_range=1024 65535*, *net.core.somaxconn=16384*, *fs.file-max=1000000* and *net.ipv4.tcp_max_syn_backlog=16384*.

## 4.2 Experiments

### 4.2.1 High Rate HTTP-DDoS via AN

This experiment is designed to investigate the effectiveness of the mitigation system against the brute force attack category. Attack sources send HTTP GET requests at their maximum achievable rate. It's also labeled a high-rate HTTP-DDoS attack as the attacker is assumed to be unwarily requesting at a rate per source that's higher than the mitigation system's detection thresholds. So, the attack category itself is not complex (i.e., not hard to detect). However, this experiment is conducted considering the possibility of this attack category, before moving on to the more complex categories. Mitigation effectiveness is measured in terms of the metrics discussed in section 4.1.1, i.e., MF, MC, and MT.

Figure 4.2 shows the experiment's setup and the physical mapping of nodes on the DeterLab testbed. All nodes are automatically assigned by DeterLab from the bpc3000, pc3000, and pc3060 physical node types [75]. 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 100 HTTP GET requests per 30 seconds for a 200 KB file from the 100 different source IPs (as described in section 4.1.1). 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). A too large value of $m \times \alpha$ is undesired from a MT perspective, but also a large enough $\alpha$ is needed to detect low rate attacks. In this experiment, we set the value of $\alpha$ to 40 [sec] from experience.




| Node | Physical Mapping | | CPU | RAM |
|---|---|---|---|---|
| AN | bpc3000 | | Dual Intel Xeon (3 GHz) | 2 GB |
| SS, PS, MN | pc3000 | | | |
| Attack Nodes (20 machines) | ×13 | pc3000 | | |
| | ×7 | pc3060 | | |

Figure 4.2 Setup and physical mapping of Experiment 1.

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 a popular tool among attackers [72]. 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 the single-request-per-connection attack category, or strategy AD1112).

Before hiding the SS, 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 (Section 4.1.3) 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.

### Start of Attack

Now the mitigation system is in place. As shown in the LHS of Fig 4.3, 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 nearly 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 (up to nearly 18 seconds). That cost later resides between 2 to 7 seconds.

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.

### Switching Attack

The experiment lasts for 24 hours, which is considered a long-duration attack [8]. We also consider the case where the attacker may decide to switch the attack ON and OFF to trick the detection mechanism. Or to try to keep the AN within the detection phase as much as possible. So, near the end, the HTTP-DDoS traffic is switched OFF and ON repeatedly as shown in Fig 4.3's RHS to observe the effect on mitigation. Switching intervals shorter than $\alpha$ (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

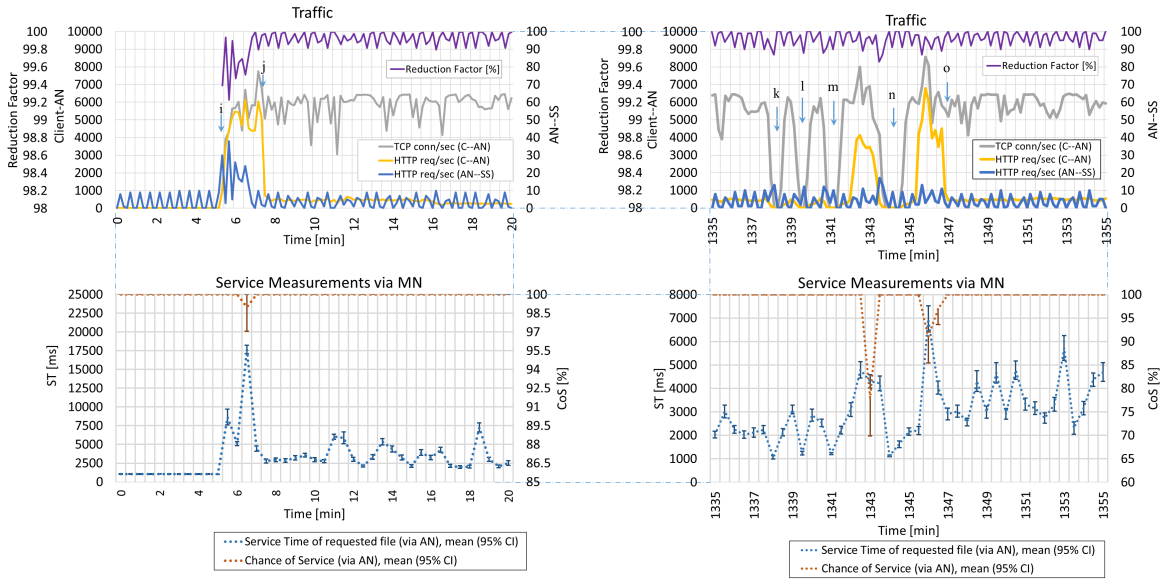



Figure 4.3 Results of Experiment 1. LHS: Start of attack. RHS: Switching attack.

drops in the $CoS_i$. Yet, mitigation time is relatively shortened, in comparison to Fig. 4.3's LHS, 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 (top of Fig. 4.3's RHS) 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 non-optimized 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, despite of the high reduction factor by the AN after the mitigation phase, we notice that the new TCP connection attempts per second on the C-AN side remains high. This shows how such automated attack source reacts on the low-level to the AN's high-level countermeasure. This volume of low-level traffic doesn't reach the SS anyway, and is utilized in mitigation by updating the prototype to achieve higher mitigation factors and lower mitigation costs.

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

### 4.2.2 Slow-Requesting HTTP-DDoS via PS

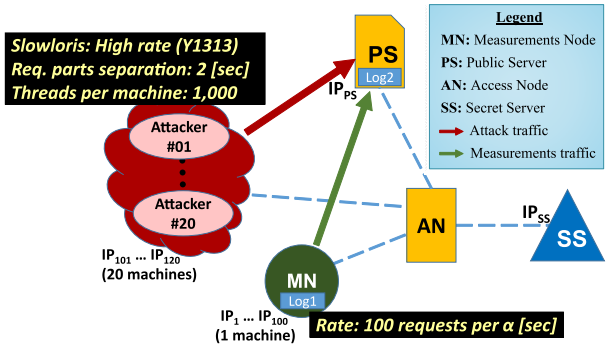
This experiment investigates the effectiveness of the mitigation system against an anticipated complex HTTP-DDoS attack category against the PS: slow-requesting

HTTP-DDoS. Mitigation effectiveness is measured in terms of the MF and MC mitigation metrics discussed in section 4.1.1. A slow-requesting 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.

The assumption is that the attacker knows the likely most effective HTTP-DDoS attack category against the PS. That's because single-request per-connection attacks release the PS's resources faster, while multiple-request per-connection attacks are detectable promptly by the PS due to its nature. It's also assumed that the attacker tries to cause damage to the service availability by wisely focusing the finite attack resources on either the AN or the PS. So, the attack sources in this experiment focus only on the PS, and don't proceed to attack the AN, while the cases of attacks that proceed to the AN are considered in separate experiments.

The setup is as shown in figure 4.4. A single SS, AN, and PS are used, each on a dedicated bpc2133 physical testbed node [75]. 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 (as described in section 4.1.1). 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). The non-attack benchmarks for the utilized Apache server (section 4.1.3) are the same for this version as well. The value of $\alpha$ is set to 30 [sec] which is neither too large nor too small. In a real implementation, the value of $\alpha$ should be adaptively tuned.




| Node | Physical Mapping | | CPU | RAM |
|---|---|---|---|---|
| AN, PS | bpc2133 | | Quad core Intel Xeon X3210 (2.13 GHz) | 4 GB |
| SS | bpc2133 | | Quad core Intel Xeon X3210 (2.13 GHz) | |
| MN | bvx2200 | | Dual Core AMD Opteron (2.2 GHz) | |
| Attack Nodes (20 machines) | ×7 | bpc2133 | Quad core Intel Xeon X3210 (2.13 GHz) | |
| | ×4 | bvx2200 | Dual Core AMD Opteron (2.2 GHz) | |
| | ×9 | bpc3000 | Dual 3 GHz Intel Xeon processors | 2 GB |

Figure 4.4 Setup and physical mapping of Experiment 2.

In this experiment, the attacker is unwarily exceeding the detection threshold. Attack traffic is distributed over 20 unique attack sources that are deployed on 7 bpc2133, 4 bvx2200, and 9 bpc3000 physical testbed nodes [75] 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 web servers [72]. 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. That's the default value of the Slowloris tool used without modification. For each attack thread, the sleep time between consecutive request parts is set to 2 seconds. That's the 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. This matches the strategy code Y1313.

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*

Note that such attack conducted directly towards the SS, utilizing only 10 Slowloris attack sources, results in a 0.0% CoS for non-attack clients (see section 4.2.5).

Now with the mitigation in place, the attack lasts for 200 minutes, of which the first 40 minutes are shown in Fig. 4.5. The four indicators plotted at the top of Fig. 4.5 are recorded by the target PS once every $\alpha = 30$ seconds. They describe the attack volume. The first indicator 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 about 2 parts before being terminated by the PS.

For all HTTP-DDoS attacks against the PS, attack traffic reaching the SS equals

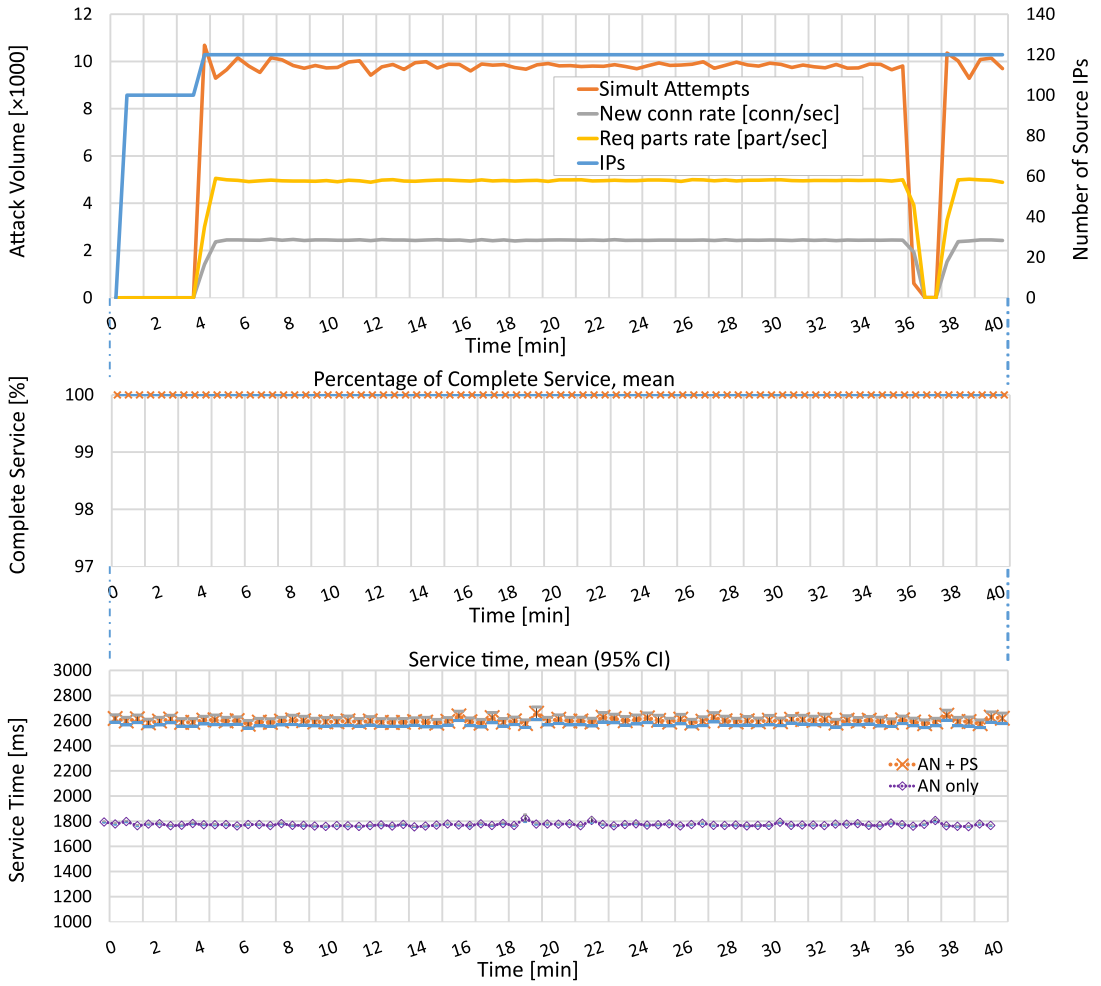



Figure 4.5 Results of Experiment 2. Slow-requesting HTTP-DDoS attack starts nearly at t = 4 [min].

zero. Therefore, $RF_i = 1$ and $MF_i = RF_i \times CoS_i = CoS_i$. So, the mitigation effectiveness of the PS is measured in terms of $CoS_i$ and $ST_i$.

During attack, the PS is show a 100% $CoS_i$ with a nearly flat mitigation cost (i.e., no increase in $ST_i$) via both the PS and the AN, which is higher than the non-attack ST benchmarks. Note that the MT metric is undefined in this experiment. 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.

So far, a large attack rate per source IP is utilized against the prototype PS. However, the observed high MF and low MC of the tested PS against high rate alone

is only a single angle to evaluate the PS from. Therefore, an additional experiment is presented next, with a more distributed attack population and higher attack intensity.

### 4.2.3 Distributed, High Rate, Slow-Requesting HTTP-DDoS via PS

This experiment extends the investigation started in experiment 2 (section 4.2.2) of the effectiveness of the mitigation system against the complex slow-requesting HTTP-DDoS attack category towards the PS. So, this experiment inherits multiple features from the previous experiment, namely; the general attacker assumptions, the physical testbed node mapping, the measurements setup, link speeds and software versions (see section 4.2.2). But to emulate a more realistic attack condition, a wider attack population is assumed this time, in comparison to experiment 2. The number of attack sources is increased 100 times, with also high-rate attack conditions per source. Mitigation effectiveness is measured in terms of the MF and MC mitigation metrics discussed in section 4.1.1.

Figure 4.6 shows the setup of the experiment. A total of 2,000 unique attack sources are emulated on 20 physical testbed nodes. They simultaneously target a single PS with the generated attack traffic. The OS installed on the attack nodes is Linux version 3.2.0-64. Measurement data is collected also by the 100 virtual sources, emulated on the dedicated MN, sending 100 requests every 30 seconds from 100 unique source IPs. The resource requested is a single 200 KB file served by SS.

For slow-requesting HTTP-DDoS attack, Slowloris is a popular tool. However, to control so many attack sources at once, a custom attack tool is developed that can be deployed on thousands of sources and be remotely controlled by a single commander node. The custom tool also enables the execution of slow-requesting DDoS attack

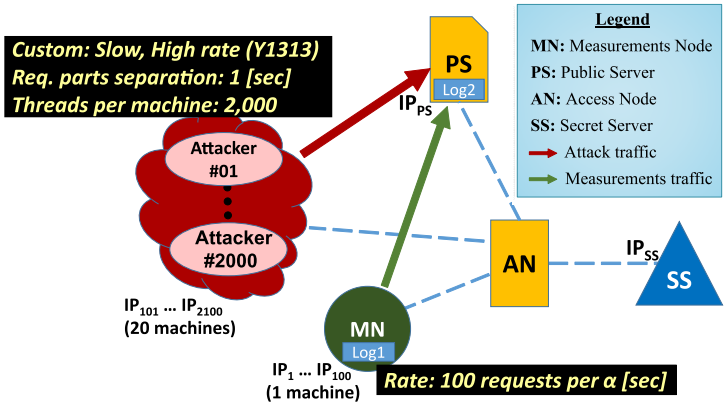



Figure 4.6 Setup of Experiment 3. Same physical mapping of Experiment 2.

behavior with tunable attack parameters.

In this experiment, the attack sources are also exceeding the detection threshold. Similar to experiment 2, the PS and the measurement node collect data every $\alpha = 30$ seconds. As observed in figure 4.7, the total rate of new connections per second equals 5000 (i.e., 2.5 per each source). This is achieved by running 20 concurrent threads per source IP (i.e., 40,000 attack threads in total). A legitimate HTTP GET request is constructed and split into small chunks of size = 20 Bytes each. One chunk is sent at a time to the PS with a 1 second separation, resulting in a high-rate slow-requesting HTTP-DDoS type of attack (strategy Y1313). That results in a total of about 20,000 simultaneous active attack connections at any given moment of time, as recorded by the PS, generating an attack rate of about 20,000 request parts per second.

At $t = 2.5$ minutes, the number of source IPs jump from 100 (measurement sources) to 2,100. This is exactly the start of attack just after the attack initiation command is received by all the attack sources simultaneously. Here also the number of IPs values are represented on the secondary axis (Fig. 4.7). For attacks against the PS, $RF_i = 1$ (i.e., $MF_i = CoS_i$). Despite of the large volume of simultaneous attempts at the single PS, we observe a $MF_i$ that's higher than 98% with a 95% confidence interval. This experiment went on for 300 minutes with no change in performance.

From the perspective of $ST_i$, the AN's performance is not affected by this type of attack, being directed at the PS (i.e., attack traffic is not reflected by the PS on the AN). As for the PS's performance, a mitigation cost can be seen, especially in the minutes following the start of attack. Without attack, the PS contributes by nearly 1 second to the whole (PS + AN) service time. During attack, the cost added by the PS resides within 1 to 3.5 seconds, with 95% confidence margin. This increase affects only the preparation phase, while following client-SS transactions are not affected. This suggests that for this category of attacks, the PS serving the preparation stage acts as a second line of defense before the AN. This is achievable by deploying thousands of PSs in the real case, which is a practical assumption.

For example, consider the case of an attacker that could gather 100,000 online distributed sources, a figure not reported before in a high-level attack, generating nearly 250,000 new connection attempts and 1,000,000 request parts per second. The results from this experiment suggest that only 50 PSs would be enough to highly mitigate such hypothetical attack, while simultaneously serving other clients. In addition, distributing the well-behaving clients over the geographically distributed ANs, where actual communication takes place trouble free and payload-inspection free.

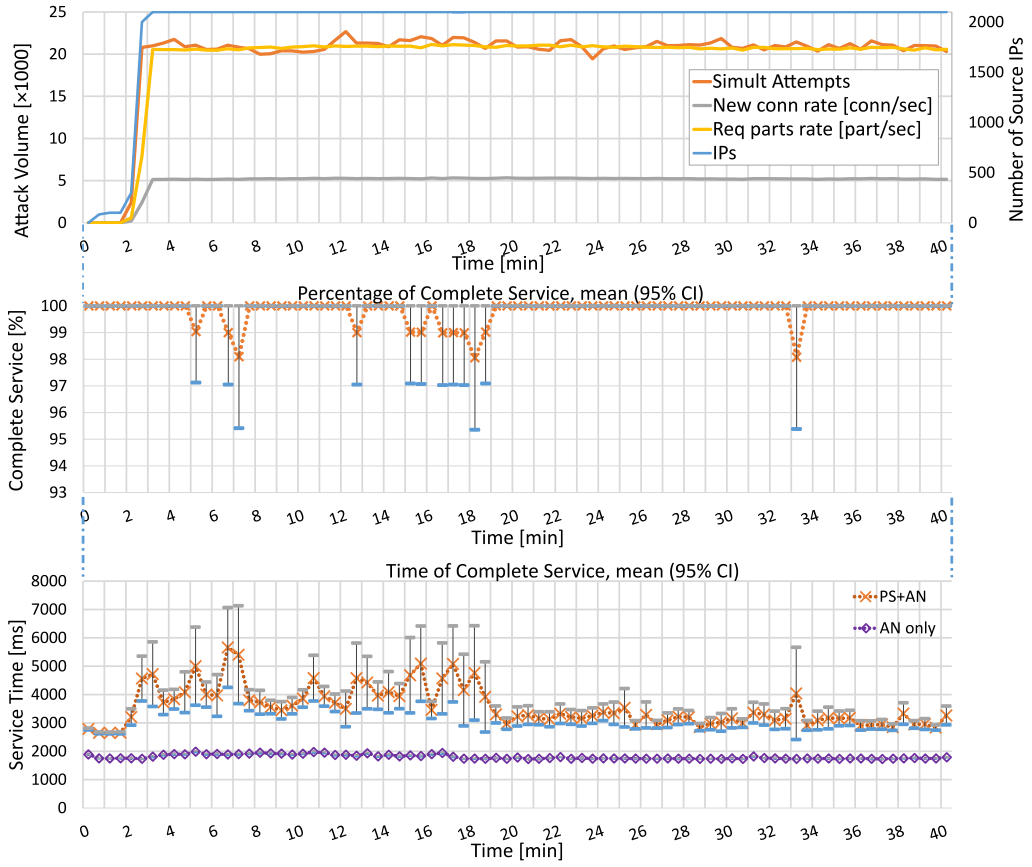
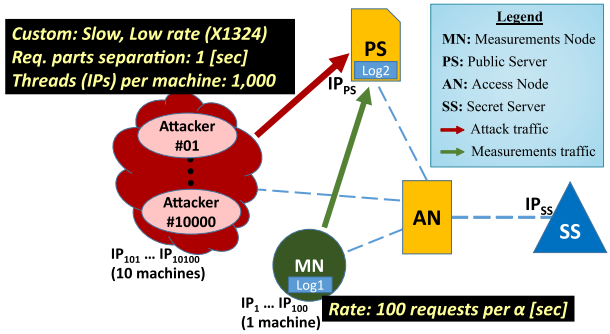



Figure 4.7 Results of Experiment 3. Slow-Requesting HTTP-DDoS attack starts nearly at t = 2.5 [min].

### 4.2.4 Highly Distributed, Low Rate, Slow-Requesting HTTP-DDoS via PS

The results of experiment 4.2.3 were encouraging to 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 500 times, in comparison to experiment 2, this time with low-rate attack conditions per source. The assumption is that failed attacker may attempt to distribute the attack traffic requests over larger number of sources to evade detection. Mitigation effectiveness is measured in terms of the MF and MC mitigation metrics discussed in section 4.1.1.

Figure 4.8 shows the experiment's setup and the physical mapping of nodes on the DeterLab testbed. All nodes are automatically assigned by DeterLab from the MicroCloud, pc2133, pc3060 physical node types [75]. The AN and SS are each running on a dedicated MicroCloud node. Additionally, the PS is running on pc2133 node. In




| Node | Physical Mapping | | CPU | RAM |
|---|---|---|---|---|
| AN, SS | MicroCloud | | One Intel Xeon (2.4 GHz) | 16 GB |
| PS | pc2133 | | One Intel Xeon (2.13 GHz) | 4 GB |
| MN | pc3060 | | Dual Intel Xeon (3GHz) | 2 GB |
| Attack Nodes (10 machines) | ×5 | MicroCloud | One Intel Xeon (2.4 GHz) | 16 GB |
| | ×2 | pc3060 | Dual Intel Xeon (3GHz) | 2 GB |
| | ×3 | pc2133 | One Intel Xeon (2.13 GHz) | 4 GB |

Figure 4.8 Setup and physical mapping of Experiment 4.

addition, 100 measurement sources, each with a unique source IP, are emulated on a pc3060 based measurement node (MN), generating 100 HTTP GET requests per 30 seconds for a 200 KB file from the 100 different source IPs (as described in section 4.1.1). 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 V3.13.0-62, while the SS is running the open-source Apache server unmodified (version 2.2.22). The value of $\alpha$ is unchanged from its 30 [sec] previous setting.

In addition, 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.

As observed in figure 4.9, the resulting total rate of new connections per second is nearly 1,650 [conn/sec] (i.e., about 10 connections per minute per source): a low-rate slow-requesting HTTP-DDoS type of attack. 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, i.e., matching the strategy code X1324.

The total duration of the attack is 12 hours, while logs are recorded at the PS and MN. Results from three time intervals of the whole experiment are presented in Fig. 4.9. The attack is switched OFF and ON multiple times immediately after its start and later after 10 hours to observe the possible effect of such behavior on the PS's performance observed initially.

The attack starts at t = 3 [min], then is switched OFF for 1 minute then back ON at t = 6 [min], 14 [min], and 602 [min], respectively. The attack is finally terminated from the source at t = 720.5 [min]. Results show a 100% $MF_i$ with a 95% confidence interval. 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.

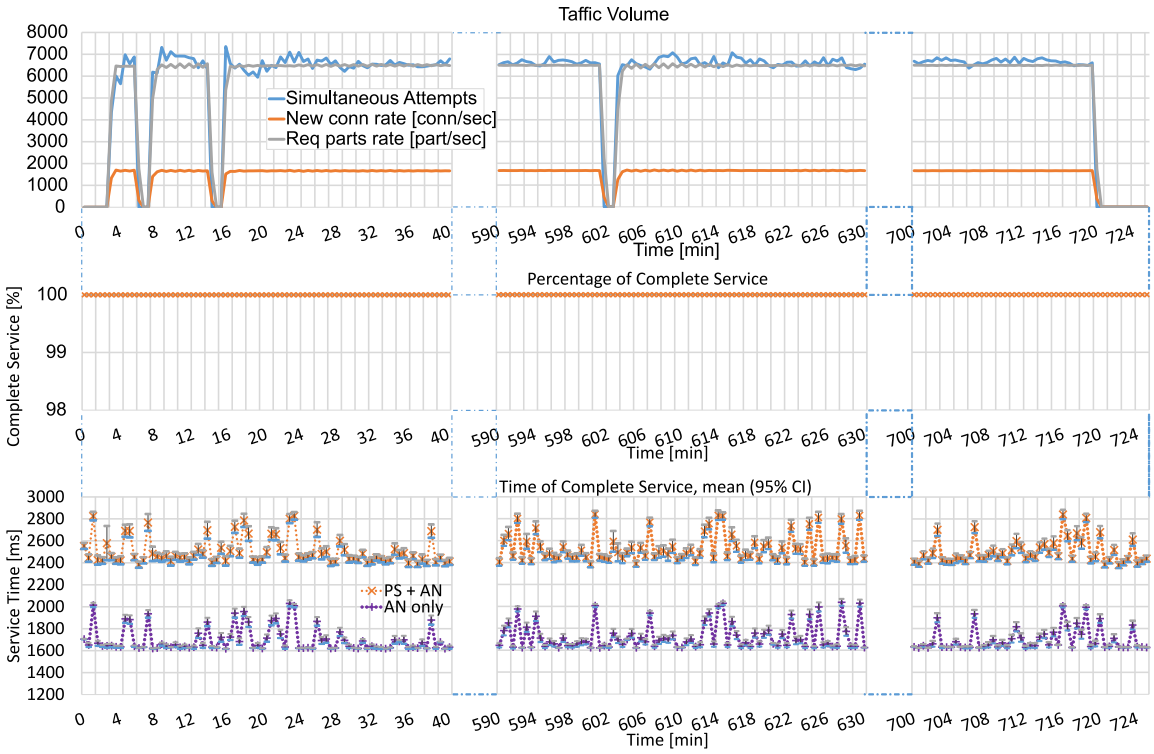



Figure 4.9 Results of Experiment 4.

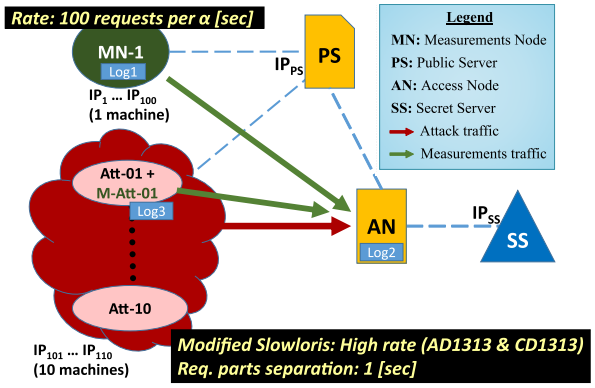
### 4.2.5 Slow-Requesting HTTP-DDoS via AN

The objective of this experiment is to study the effectiveness of the proposed method, with the simplified prototype, against this complex category of HTTP-DDoS attacks, in terms of the MF, MT, MC and CD metrics. In this experiment, it's assumed that the attacker is aware of the defense setup and so proceeds with the PS normally then attacks via the AN. Also, the attacker knows that the AN doesn't inspect the content and therefore attempts the slow-requesting HTTP-DDoS attack category.

Figure 4.10 shows the experiment's setup and the physical mapping of nodes on the DeterLab testbed. All nodes are automatically assigned by DeterLab from the MicroCloud and dl380g3 physical node types [75]. The AN, PS and SS are each running on a dedicated dl380g3 node. Also, all testbed link speeds of 100Mbps are used. The operating system used for all nodes is Linux V3.13.0-74, while the SS is running the open-source Apache server unmodified (version 2.4.7). The value of $\alpha$ is unchanged from its 30 [sec] previous setting.

In addition, the 100 measurement sources, each with a unique source IP, are emulated on a MicroCloud based measurement node (MN). Every 30 seconds, a request for a 200 KB file is sent form each source IP over HTTPS, generating 100 requests per 30 seconds from the 100 different source IPs. So, we have 100 data points collected every 30 seconds as raw data in terms of service time, transfer speed, and connection success. The plotted measurements data in Figures 4.11 and 4.12 represent the average values of each 100 points with a 95% confidence margin.

In addition to the data recorded at the AN and the MN, an additional measurement source is run on one of the attacking nodes (i.e., sharing the same source IP address of the attacker). The reason is to see the impact of an attacking user on a non-attacking user sharing the same source IP. This is in attempt to observe the possible collateral damage, introduced by the proposed method, on users sharing an IP with an attacker.




| Node | Physical Mapping | | CPU | RAM |
|---|---|---|---|---|
| AN, PS | dl380g3 | | Xeon E5-2420 v2 2.5 GHz | 16GB |
| SS | | | | |
| MN | MicroCloud | | Xeon E3 LP 2.4 GHz | 16GB |
| Attack Nodes (10 machines) | ×4 | MicroCloud | | |
| | ×6 | dl380g3 | Xeon E5-2420 v2 2.5 GHz | 16GB |

Figure 4.10 Setup and physical mapping of Experiment 5.

The additional measurements point is setup at one of the attack machines, i.e., Att-01, and is henceforth referred to as M-Shared-1. M-Shared-1 performs the same the procedure of the MN, except that it has a single source IP, and sends 10 HTTPS requests every 60 seconds for the 200 KB file.

Now, for an attacker to target the SS through an AN, there are some things to consider. Readily available, slow-requesting DDoS attack tools can be used for testing. However, for such tools to be compatible with the proposed method, a modification is required. Specifically, these tools need to perform 2 extra steps before any attack traffic is sent towards the AN. Otherwise, the AN will drop that traffic by default. The first step is sending a valid request to the PS and successfully following the resulting redirection in time, using the correct SID, at the correct AN. The second step is sending a valid request through the assigned AN, also in time, and waiting for the service to complete without exceptions. So that the AN confirms the registered SID. After both steps, then the actual attack tool can commence its malicious operation.

So, to generate the test slow-requesting HTTP-DDoS attack traffic, we have developed a custom tool that performs the initial preparation steps successfully then executes the Slowloris code towards the correct AN and SID. The Slowloris tool can't be used with virtual IP addresses, and so each attack node on the testbed is with only a single source IP address. Total of 10 different attack sources (emulated on 4 MicroCloud and 6 dl380g3 testbed nodes) target a single AN. Each attack node first completes the initial preparation and then the Slowloris code is executed.

Then, the Slowloris code itself starts multiple threads, which are set to restart every 2 minutes. For each attack thread, the TCP connection reacquisition rate is set to 2 seconds, while the sleep time between consecutive request parts is 1 seconds. So, each thread initiates its connections to the AN and renews them separately, while per a single connection there's a part of single but incomplete fixed request arriving at a high rate. This matches the strategy code AD1313. Further, throughout the experiment's duration, the attack was switched off and on 4 times. That way, the 10 sources' PIDs have renewed their SIDs 4 times, which matches the strategy CD1313 as well.

Before hiding the server behind the mitigation system, we wanted to observe the impact of the same attack volume on the SS without mitigation. The 10 Slowloris attack sources targeted the SS directly, without mitigation, for nearly 40 minutes. The attack was also switched off and on repeatedly. Results in Fig. 4.11 show the service measurements of the test server. The directed Slowloris attack on the unprotected web-server, utilizing only 10 machines, show an abrupt drop in CoS to 0.0%during attack intervals. On the other hand, the service time is undefined during such intervals.

Now the mitigation system is in place. Mitigation is enforced on two degrees as described in section 3.4.3. Firstly by probing each client with a the slow-responding mechanism before proceeding with actual communication. Secondly, the AN monitors

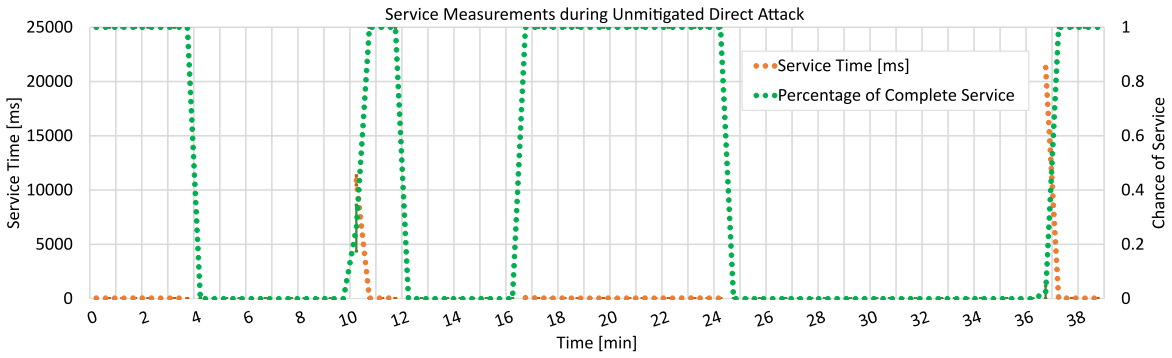



Figure 4.11 Service Measurements of the test web server under direct slow-requesting HTTP-DDoS attack.

the averages of past m values of $C/\alpha$ and $FC/\alpha$ for each unique SID. The value of m is a tunable parameter that need not be too large, and it must be less than or equal to $\beta$. In this experiment, m is set to 4 from experience. An SID is considered aggressive if it shows a nearly constant level of $C/\alpha$ and $FC/\alpha$ for the past m time steps. Aggressive SIDs are put in "jail" for one time step (i.e., 30 seconds) while still observing $C/\alpha$ for each SID. During jail time, all new connection attempts are accepted, then immediately closed and registered in the enhanced record.

As shown in Fig 4.12, attack traffic starts at t = 4 [min]. Then till t = 6 [min] is the mitigation phase of the AN, during which the peak achievable attack rate on the public side reaches 1,044 [r/s], while $Att_{pub}$ is nearly 546 [r/s]. As mentioned, each attack source creates an SID and attacks though it, then renews this SID 4 times throughout the attack operation. Each time a new SID is created, the level of high-level requests spikes temporarily before the AN mitigation takes place. The volume of connection attempts (i.e., low-level) per source remains nearly constant during attack, and only drops when the attack source switches off the attack. On the other hand, the high-level traffic is affected by the AN's measures. From both the MN and M-Shared-1, the AN's effect is seen in terms of CoS and the reduction in attack traffic reaching to the SS. Also, for every time step $i$, the $RF_i$ is observed above 99% after the mitigation phase. However, during the mitigation phase, the value of $RF_i$ drops momentarily. Note that before t = 4 [min], RF is undefined. The AN also shows a 100% $CoS_i$ (i.e., $MF_i = RF_i$), far from the server, with zero knowledge of the requested high-level content. Also, a cost is observed in terms of an increase in $ST_i$ for the requested file. It rises from the normal value of nearly 900 [ms] to reach to around 2 seconds, from the AN. Notice that the service time through the AN is the one that shapes the curve of the total service time (through the PS + AN) since it's the one under attack.

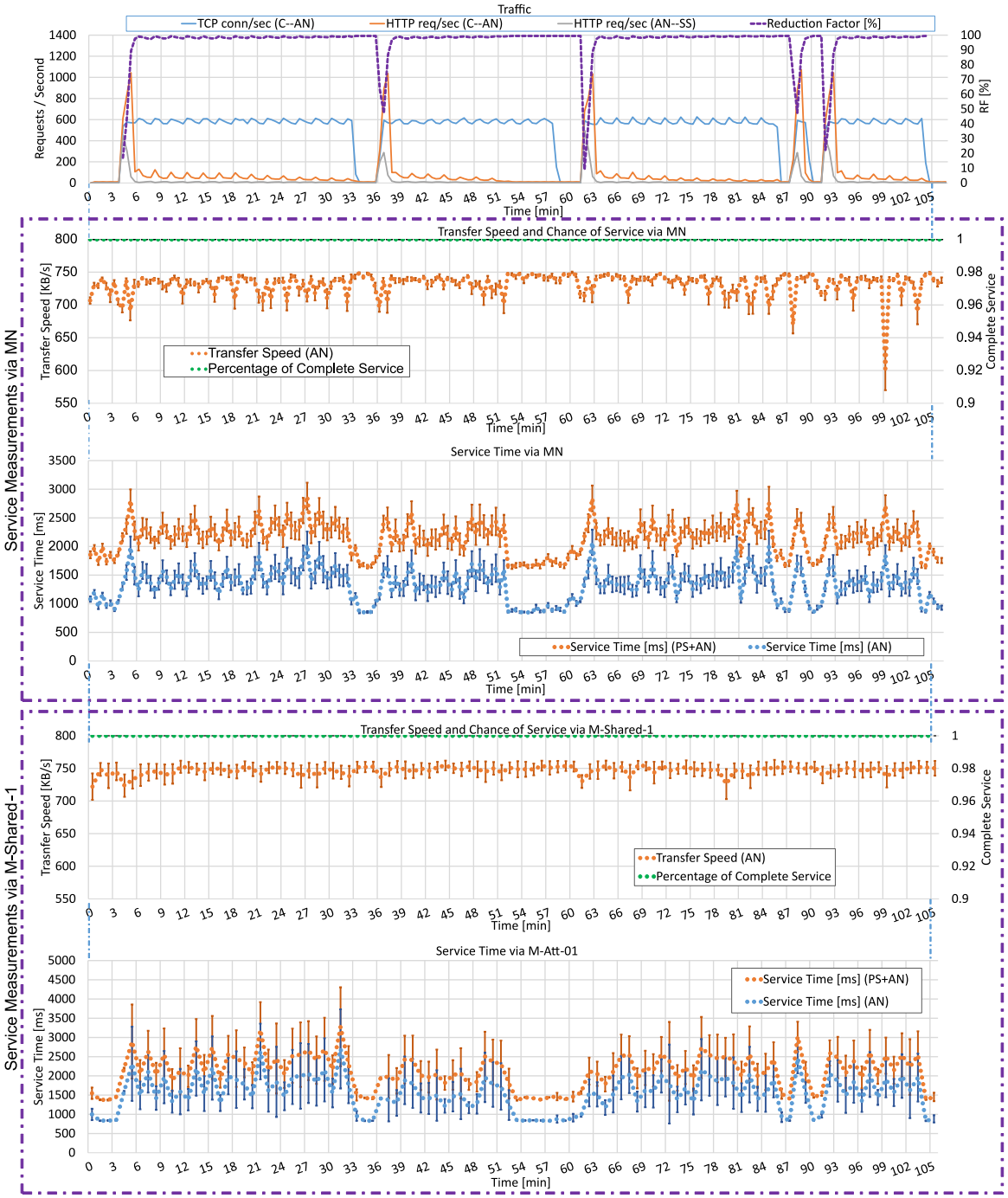


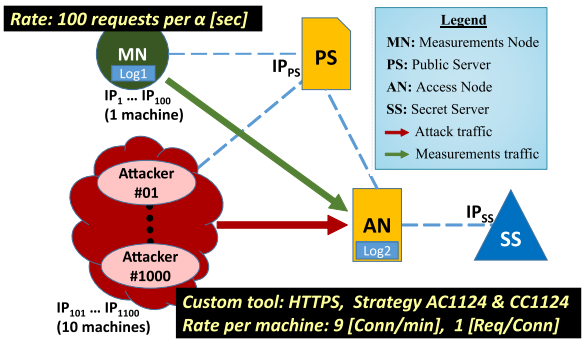

Figure 4.12 Results of Experiment 5.

### 4.2.6 Low Rate HTTPS-DDoS via AN

The purpose of this experiment is to investigate the mitigation effectiveness of the proposed method against a more complex HTTP-DDoS attack condition. We assume that the attacker adapts further by increasing the number of attack sources, switching to encrypted HTTP-DDoS (i.e., HTTPS-DDoS), sending a single request per connection to evade conventional per-connection detection methods, and significantly reducing the attack rate to below the AN's set detection thresholds (i.e., low-rate, single-request per-connection, sub-detection-threshold HTTPS-DDoS attack). As such complex high-level attack conditions are absent from related research, it's investigated for the first time in our research.

A sub-detection-threshold attack is expected in three cases. The first possibility is that the attacker somehow learns the detection threshold values and is intentionally setting the rate per source below it (i.e., a knowledge-based or defense-aware attack). The second possibility is where the AN threshold is simply misconfigured. The third possibility is that the AN threshold is already configured properly, but is set to the minimum practical value which is above the attack rate. The attacker achieves this combination by reducing the attack rate generated per source (nearly 1,800 times less attack traffic per source in comparison to experiment 1) and increasing the number of unique sources (50 times more). Mitigation effectiveness is measured in terms of the MF, MC, and MT metrics discussed in section 4.1.1.

Figure 4.13 shows the experiment's setup and the physical mapping of nodes on the DeterLab testbed. All nodes are automatically assigned by DeterLab from the bpc3000, pc3060, and bpc3060 physical node types [75]. One prototype AN and PS are deployed before the SS, where each is running on a dedicated bpc3000 node. Like previous experiments, all testbed link speeds of 100Mbps are used. The operating system used for all nodes is Linux V3.13.0-74, while the SS is running the open-source Apache server unmodified (version 2.4.7). The value of $\alpha$ is unchanged from its 30




| Node | Physical Mapping | | CPU | RAM |
|---|---|---|---|---|
| AN, PS | bpc3000 | | Dual Intel Xeon (3 GHz) | 2 GB |
| SS | | | | |
| MN | | | | |
| Attack Nodes (10 machines) | ×3 | pc3060 | Dual Intel Xeon (3 GHz) | 2 GB |
| | ×1 | bpc3060 | | |
| | ×6 | bpc3000 | | |

Figure 4.13 Setup and physical mapping of Experiment 6.

[sec] previous setting.

Experiment measurements data are recorded in Log1 at the MN and Log2 at the AN. Like previous experiments, the MN emulates 100 unique sources, by running 100 threads each with a dedicated virtual IP address. Every time step (30 seconds), a request for a 200 KB file is sent form each MN source IP over HTTPS and then its service time, transfer speed, and service success are recorded. So, we have 100 data points collected every 30 seconds in Log1. The plotted measurements data in Figures 4.14 and 4.15 represent the average values of each 100 points with a 95% confidence margin. The MN itself is also of the bpc3000 physical node type. The traffic recorded by the AN in Log2 includes: the number TCP connections per second observed by the AN on the client (C) side, the number of encrypted HTTP requests per second observed by the AN on the C side, and the number of encrypted HTTP requests per second observed by the SS (i.e., on the AN-SS side).

In this experiment, the attack traffic is generated by 1,000 unique attack sources (emulated on 3 pc3060, 1 bpc3060, and 6 bpc3000 DeterLab physical nodes [75]). Manually passing the PS's preparation stage to attack via the AN was doable in case of 20 attack sources like in experiment 1, but it becomes unimaginable in case of 1,000 sources. Also, existing attack tools such as LOIC (utilized in experiment 1 against the AN) offer limited control over a few attack variables and doesn't execute HTTPS-DDoS. So, a more flexible attack tool is developed for testing. The developed tool also enables the emulation of multiple HTTPS-DDoS attack sources (i.e., each with a unique IP) per physical testbed node. This is particularly useful in case of low-rate HTTPS-DDoS attacks.

Initially, each attack source initiates 100 concurrent threads and normally passes the PS's preparation stage, obtaining a valid SID, then start the low-rate HTTPS-DDoS attack. Through each fixed SID, each attack source sends a valid single encrypted HTTP GET request over a single connection for a 200 KB file to the SS via the AN. Each source then waits on that connection and reads the complete response from the server, then disconnects. Once that request is successfully serviced, each requesting source creates a new connection through the same SID after a 5 seconds delay, re-sends the request, and so on.

Since there's only a single request per connection per source IP, so the actual achievable attack rate per source depends on the time of service for each request. In the resulting attack's peak, each source IP is requesting at 9.78 [connections/min], with 1 [request/connection]. So, in total the 1,000 sources at the start of attack peak at 163 [r/s] (i.e., requests per second). Steady attack volume is assumed since the attack rate per source is too low (measured in requests per minute) to be affected by link variations (measured in milliseconds). This matches the single-request-per-connection attack category and sub-detection-threshold attack category (strategy AC1124). The attack sources also try renewing their SID multiple times in the middle of the attack

(strategy CC1124).

At first, to observe the impact of these attack conditions on the SS without protection, the generated HTTPS-DDoS attack traffic is targeted directly on the SS (i.e., without mitigation).

The test against the unprotected server is executed for 60 minutes. Its results can be seen in Fig. 4.14. As soon as the attack starts, the unprotected server suffers an increase in service time (ST) from what initially was nearly 90 [ms] to around 15 [sec]. That increase doesn't subside throughout the attack duration. In addition, the chance of receiving a complete service (CoS) is degraded.

As in the previous experiment (section 4.2.5), mitigation is enforced on two degrees as described in section 3.4.3. Firstly by probing each requester with a the slow-responding mechanism before proceeding with actual communication. Secondly, by analyzing the enhanced behavior records of $C/\alpha$ and $FC/\alpha$ for each unique SID. The value of m is a tunable parameter, which is set to 4 in this experiment from experience as need not be too large and must be less than or equal to $\beta$. An SID is considered aggressive if it shows a nearly constant level of $C/\alpha$ and $FC/\alpha$ for the past m time steps. Aggressive SIDs are put in "jail" for one time step (i.e., 30 seconds) while still observing $C/\alpha$ for each SID. During jail time, all new connection attempts are accepted, then immediately closed and registered in the enhanced record.

Now the mitigation system is in place. The attack lasts for nearly 280 minutes, which is considered a long-duration high-level DDoS attack recently [8]. As shown in Fig. 4.15, the attack is started nearly at t = 3 [min], where at t = 5 [min] the mitigation time is complete. The attack is later switched off and on repeatedly in the interval starting from t = 242 [min] till t = 252 [min]. Results in Fig. 4.15 focus on these two time intervals. During that the first mitigation time of the AN, the peak achievable attack rate on the public side reaches 163 [r/s], while $Att_{pub}$ is

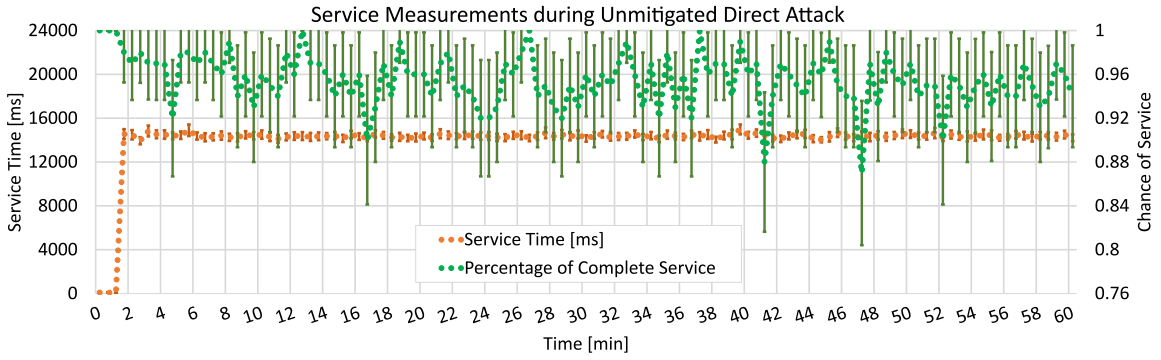



Figure 4.14 Service Measurements of the test web server under a very low rate, direct HTTPS-DDoS attack.

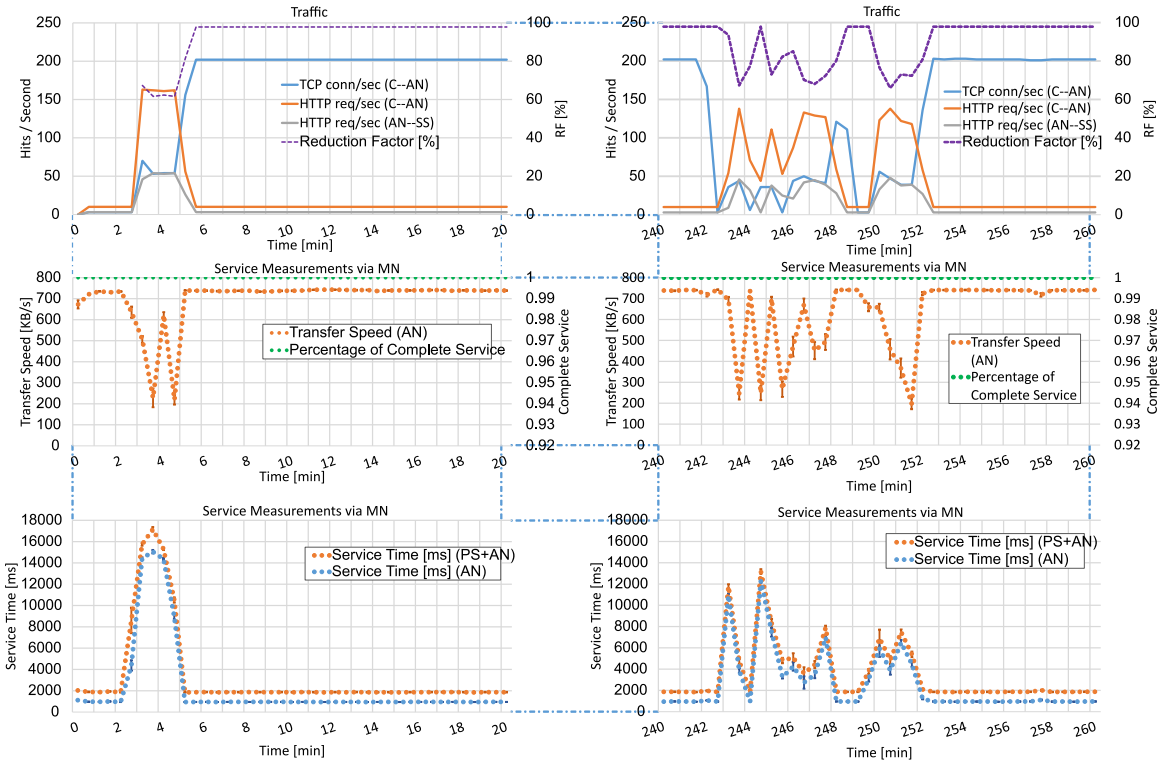



Figure 4.15 Results of Experiment 6.

approximately 140.8 [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 observed $RF_i$ is nearly 97.8% after the mitigation phase. However, during the mitigation phase, the value of $RF_i$ is 61.6% or higher. Before the attack start, $RF_i$ is undefined. The AN also shows a 100% $CoS_i$ during and after the mitigation time. So, the value of $MF_i = RF_i$. Yet, the mitigation cost is observed as an increase in $ST_i$ for the requested file during the mitigation phase.

Similar to previous experiments, in response to the AN's measures, the new TCP connection attempts per second on the C-AN side rises from 54 requests per second initially during the mitigation phase, to nearly steadily 200 requests per second afterwards, despite of the reduction in high-level request rate.

As the attack sources switch their malicious traffic off and back on, while renewing their SIDs repeatedly, like the case seen starting from t = 242 [min], we observe the AN's effect again in terms of $RF_i$ and $ST_i$, which indicates how the AN re-enters a new mitigation phase. The switching attack resulted in service time increase to nearly 12 seconds at first, then around 7 seconds. On the other hand, the chance of service metric doesn't change throughout the experiment.

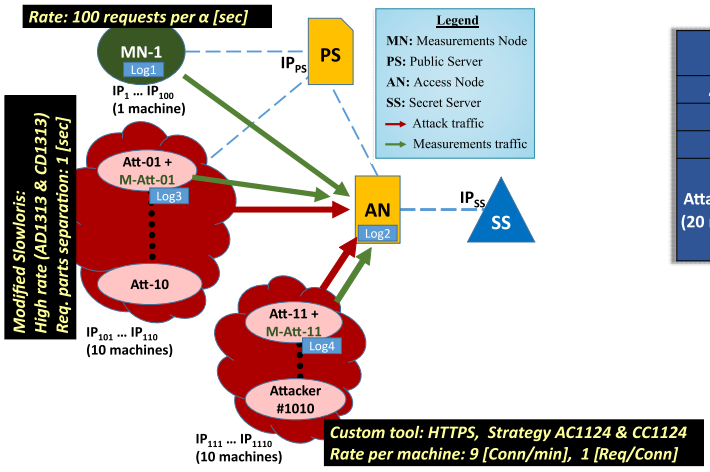
### 4.2.7 Multivector HTTP(S)-DDoS Attack via AN

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. The goal is to observe the AN's effect on such traffic far from the server and evaluate the MF, MC, MT, and CD.

This time, an updated version of the AN prototype is used. It includes the second countermeasure described in Sect. 3.4.3, and a reduced $P(0,3)$ (in the $P_i$ lookup table described at the end of Sect. 3.4.2) from 8 to 6, for a less aggressive initial response against new SIDs.

The setup of the experiment is as shown in Fig. 4.16. The utilized node types bvx2200, bpc2133, and pc2133 were automatically selected by DeterLab [75]. 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 V3.13.0-74, while the SS is running an unmodified Apache server (version 2.4.7). 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,




| Node | Physical Mapping | | CPU | RAM |
|---|---|---|---|---|
| AN, SS | bvx2200 | | Dual Core AMD Opteron 1.8 GHz | 4GB |
| PS | | | | |
| MN | | | | |
| Attack Nodes (20 machines) | ×5 | bvx2200 | | |
| | ×2 | bpc3000 | Dual Intel Xeon (3 GHz) | 2 GB |
| | ×13 | pc2133 | One Intel Xeon (2.13 GHz) | 4 GB |

Figure 4.16 Setup and physical mapping of Experiment 7.

we utilize Slowloris which is commonly used by attackers [72]. 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 generated multi-vector attack traffic matches the single-request per connection, multi-behavior per-shared-IP, and sub-detection-thresholds attack categories (or strategies: AC1124, CC1124, AD1313, and CD1313).

Before hiding the SS, these attack conditions are directed against the SS without mitigation. Results showed 0.0% CoS under this direct double-vector attack. This is inline with the impact observed in Figures 4.14 and 4.11.

Then, the experiment is run for 120 [min], of which the first 40 [min] are shown in Fig. 4.17. 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 the mitigation phases, coinciding with the attack start and later restarts. The value of $ST_i$ temporarily 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 (Section 3.4.3) and enhanced identification can be effective in remotely isolating automated attack categories, even without knowledge of the traffic content, and 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 (i.e., more than $\alpha$). 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 temporarily. The prototype AN showed consistent performance throughout the experiment.

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. 3.4.3). The increase results in extra probing time only for these IPs, until the

reputation automatically returns to normal after $\beta$ 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 complex attack may adapt

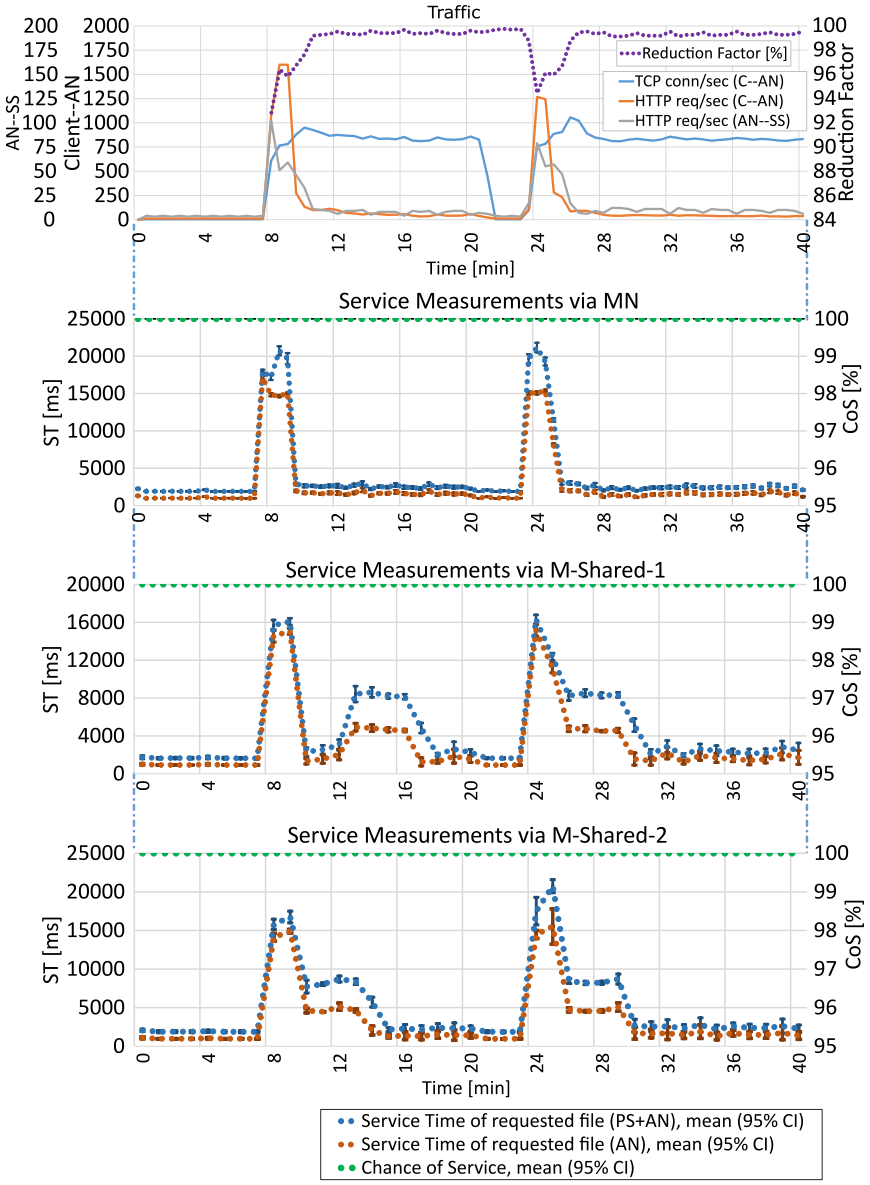



Figure 4.17 Results of Experiment 7: Traffic and Service Measurements via MN, M-shared-1, and M-shared-2.

by constantly switching its traffic. This is discussed in Sect.5.4.

Figure 4.18 summarizes the results of the experiments discussed in this chapter.

| | Sec. | Category | Duration | Tool | Attack sources | Attack peak (per source) | Direct CoS | CoS via PS | MC |
|---|---|---|---|---|---|---|---|---|---|
| PS | 4.2.2 | Slow-Requesting HTTP-DDoS | 3.3 [hr] | Slowloris | 20 | 5,000 (250 [r/s]) | 0% | 100% | ~ 0 |
| | 4.2.3 | Distributed, High Rate, Slow-Requesting HTTP-DDoS | 5.2 [hr] | Custom | 2,000 | 20,000 (10 [r/s]) | 0% | >98% | < 2.5 [sec] |
| | 4.2.4 | Highly Distributed, Low Rate, Slow-Requesting HTTP-DDoS | 12 [hr] | Custom | 10,000 | 6,450 (0.645 [r/s]) | 0% | 100% | ~ 0 |

| | Sec. | Category | Duration | Tool | Attack sources | Attack peak (per source) | MT | Direct CoS (ST) | CoS during MT via AN (RF) | Cos after MT via AN (RF) | MC during MT | MC after MT |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| AN | 4.2.1 | High Rate HTTP-DDoS | 24 [hr] | LOIC | 20 | 6,149 (307.5 [r/s]) | 2 [min] | 66.5% (8.47 [sec]) | >78% (>99.6%) | 100% (99.8%) | < 16.5 [sec] | < 6.2 [sec] |
| | 4.2.5 | Slow-Requesting HTTP-DDoS | 1.8 [hr] | Modified Slowloris | 10 | 1,044 (104.4 [r/s]) | 2 [min] | 0% | 100% (>9%) | 100% (99%) | < 1 [sec] | < 1 [sec] |
| | 4.2.6 | Low Rate HTTPS-DDoS | 4.7 [hr] | Custom | 1,000 | 163 (0.163 [r/s]) | 2 [min] | ~90% (15 [sec]) | 100% (>61%) | 100% (97.8%) | < 14 [sec] | ~ 0 |
| | 4.2.7 | Multivector HTTP(S)-DDoS | 2 [hr] | Custom + modified Slowloris | 1,000 + 10 | 1,601 (1.6 [r/s]) | 1.5 [min] | 0% | 100% (>92.8%) | 100% (99.2%) | < 16 [sec] | < 1 [sec] |

Figure 4.18 Summary of experiments.

# Chapter 5

# Discussion

## 5.1 Service Time Cost

With the extra preparation and communication steps, the proposed framework introduces a cost in service time even in absence of attack.

For example, consider Figure 5.1. It compares the non-attack measured service time for three file sizes for demonstration when requested directly from the SS, and via the mitigation system considering the effect of the PS (via the PS + AN) and excluding it (via AN). For the same requested file, with no concurrency, the service time via the prototype system is much larger in comparison to the direct C-SS communication case.

However, several points to be taken into consideration. First, service time variation is expected with the SS's load, especially in case of attack. For example, the service time of the requested file directly from the SS rises to 1825.4 [ms] with only 100 concurrent requests (see section 4.1.3's Figure 4.1). So, service time directly from the SS may be lower during non-attack time, but as DDoS attacks overwhelm it with unmitigated requests then the service time via the prototype system become comparable. For example in section 4.2.4, with nearly 10,000 concurrent attempts per single PS we observe a flat service time cost of 2,600 [ms] for the first request (i.e., via PS + AN), and 1,800 [ms] for subsequent ones (i.e., via AN). Second, the PS's contribution to service time is only observed once by the client. In this case, it's nearly constant 567 [ms] for all three requested files. Consequent requests from the client experience only the cost by the AN. Third, the current system implementation is not an optimal version, but rather a proof of concept prototype. For example, optimizing the AN's data relay thread management would lead to more efficient data relaying by the AN and so reduce the service time seen by clients.

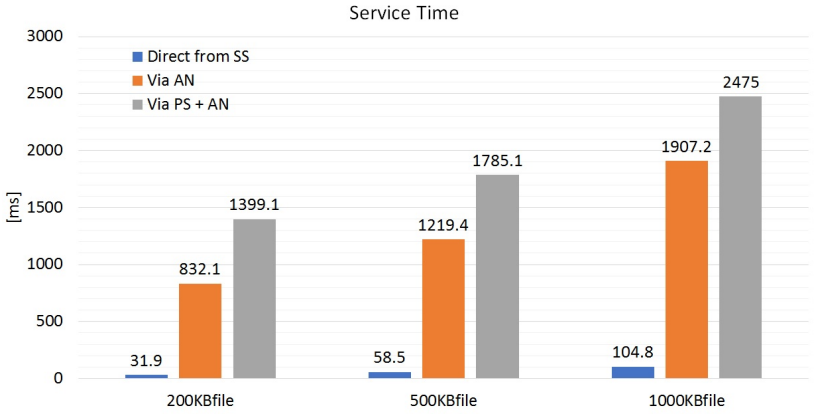



Figure 5.1 Non-attack service time measurements.

## 5.2 Scalability

Any server has a finite local network capacity ($C_{NW_S}$), and processing capacity ($C_{P_S}$). For this server to be available for service, then $C_{NW_S}$ must never fall below the combined volume of the inbound attack ($V_{NW_{Att}}$) and legitimate traffic ($V_{NW_{Legit}}$). In addition, $C_{P_S}$ must never fall below the combined processing requirements volume of the inbound attack ($V_{P_{Att}}$) and legitimate traffic ($V_{P_{Legit}}$). This can be expressed in formulas (5.1) and (5.2). So, denial of service takes place any time $V_{NW_{Att}} + V_{NW_{Legit}}$ exceed $C_{NW_S}$, or if $V_{P_{Att}} + V_{P_{Legit}}$ exceed $C_{P_S}$.

$$V_{NW_{Att}} + V_{NW_{Legit}} < C_{NW_S} \tag{5.1}$$

$$V_{P_{Att}} + V_{P_{Legit}} < C_{P_S} \tag{5.2}$$

That said, we define two kinds of scalability. Scalability against attacks (i.e., against $V_{P_{Att}}$ and $V_{NW_{Att}}$), and scalability with legitimate traffic (i.e., with $V_{P_{Legit}}$ and $V_{NW_{Legit}}$).

Scalability against attacks, means that as DDoS attacks increase in volume, the mitigation scheme is still able to handle it efficiently. In that sense, scalability against $V_{P_{Att}}$ and $V_{NW_{Att}}$ is fully the mitigation solution's responsibility. It has nothing to do with the server's capacity to handle traffic (i.e., $C_{P_S}$ or $C_{NW_S}$). Assuming enough over provisioning of nodes by the mitigation framework.

On the other hand, scalability with legitimate traffic depends on the mitigation framework, as well as the server's capacity. Thus, it's a shared responsibility. They both have to provide the enough capacity required by legitimate users' traffic. So, scalability with $V_{P_{Legit}}$ and $V_{NW_{Legit}}$ is a concern for the protected server too, which is required to provide the necessary local processing and network capacity for legitimate

users. Yes, local, since the proposed method is geared towards end-to-end-encrypted content. From the mitigation framework's perspective, scalability is achievable by distributing traffic over thousands of nodes, thus eliminating the possibility of choke points.

**Scalability against attacks:**

So, the scalability question becomes; can the proposed framework scale against massive attack traffic?

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 [8], which can't reach to the SS anyway in the proposed scheme. We discuss about low-level attacks further in Sect. 5.4.

Conversely, high-level attacks are inherently much smaller in volume since they can't be amplified, and to evade detection. It is practical to assume tens of thousands of nodes in operation, especially because of the modularity of the PSs and ANs which makes them easy to replicate. Also, traffic sources (clients or attackers) have no choice on which AN to use, and are assigned by the system. In addition, all ANs and PSs are shared by all SSs. DDoS traffic by nature is focused on a finite number of targets at any point in time. So, actual clients currently being served during the attack incident are transferred to new nodes, where the attacking sources are left with no clue on how to deny the specific services being targeted.

For example, results of the experiment in section 4.2.3 suggest that a single PS can handle nearly 5000 new connections per second from 2,000 different attack sources, generating nearly 20,000 simultaneous attempts with attack rate of 20,000 request parts per second, without a large cost in service chance or time. On a similar scale, an attack traffic of 6,149 [r/s] towards the AN is utilized in the experiment in section 4.2.1. This is a significant volume per overlay node, considering the reported peak high-level attack rate of 268,800 [r/s] [8]. So, to mitigate the reported peak for example, nearly 44 ANs would be required which is possible with tens of thousands of overlay nodes. Also in later experiments, starting from Section 4.2.5, the prototype AN is equipped with the second degree countermeasure described in Section 3.4.3, which restricts the aforementioned cost to only within the mitigation phase. This can apply to the experiment in Section 4.2.1 as well.

Yet, even with a mitigation time as short as 2 minutes, it's desired to reduce the mitigation cost inside that interval too. An open research direction would be utilizing the proposed framework to develop additional degrees of countermeasures based on machine learning that can reduce the cost even within the short detection phase. One possible such countermeasure would be based on monitoring past records of $S/\alpha$, $S/P$,

$FS/\alpha$, and $FS/P$ so that switching attack behaviors are identified.

**AN's port exhaustion:**

Further, the AN should not run out of available ports for allowing new SIDs. 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.

## 5.3 Evaluation Method

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. It's assumed that evaluating the mitigation effectiveness against the complex version of an attack category suggests the same effectiveness is achievable against its less complex version.

We emulated various 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. In the conducted experiments, we utilized popular tools among attackers (e.g., LOIC and Slowloris), which generate a high-rate single-request per-connection types of attacks [72, 76]. In addition, we utilized custom tools that we developed to emulate multi-behavior per-shared-IP, sub-detection-threshold, and multi-vector HTTP(S)-DDoS attacks. Next section discusses about other attack possibilities.

Note that DDoS datasets are often utilized to compare the performance of different machine learning algorithms [77–79]. Although utilizing attack traffic datasets can be useful in quantitative comparison between mitigation methods, existing datasets would require pre-experiment preparation for each source IP in order to be usable with the proposed system. Firstly, it's required to specify the number of clients per IP and specify the first connection per client (SID). Both are unknown. Secondly, for each client, it's required to specify which of following connections are related. Also, create

a session for them at the AN (i.e., synthetic redirection). Synthetic redirection means that eventually the attack traffic will be generated similar to our method. Therefore, such datasets offer no advantage to our traffic generation method due to the redirection message from the PS. In addition, at the PS, datasets that record the type of requested resource would have no effect on the PS performance since the PS performs constant tasks regardless of the actual requested resource. Similarly, the type of requested resource has no effect on AN detection. Also, the number of requests per connection expected by the PS must be one, otherwise the attack is easy to mitigate.

Also, in some of the experiments the number of attack sources is multiplied via virtualization, while in others the number of sources is small. That's necessary given a finite number of testbed nodes. However, in the real situation, more attack sources are anticipated. Notice that we categorize attacks into two kinds; low-rate per source and high-rate per source. High-rate attacks are not as complex as low rate attacks, since they are above thresholds, and therefore are more easily detectable. Their difficulty lies in their traffic volume. So, testing with a small number of high-rate sources offers an understanding of how an $n$ times larger number of such sources can be mitigated by distributing them over $n$ Ans, for example. So, the small number of high-rate attack sources per AN is realistic when put in that context. On the other hand, low-rate attacks are the most difficult to detect, since they are below detection thresholds. We assume that since the rate per source in this case is small, then combining multiple virtual sources per one testbed node do not affect the realism of the generated traffic.

Further, telling how much the utilized experiment topologies can predict the behavior on a larger, real, topology is an open research problem [66]. Eventually, live deployment and experimentation of performance would be needed before final adoption, and is a promising research direction, that builds on our work which demonstrated the soundness of the concept itself.

The conducted evaluations are based not only on the amount of attack traffic reduction, but rather on a comprehensive set of metrics (described in section 4.1.1) that are periodically measured with respect to time. These metrics aim to describe the mitigation from three main angles; the amount of traffic reduction in high-level traffic by the overlay-node (i.e., RF), the chance of receiving the service by a non-attacking client (i.e., CoS), and the time taken to mitigate the attack (i.e., MT). For the AN, both RF and CoS are jountly represented in a single metric (i.e., MF). In addition, the MC and CD metrics are measured in attempt to further describe the effect of mitigation on the service. Yet, these metrics are to be used with caution. For example, the definition of mitigation time describes 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 $Att_{\text{pub}}$. Also, the measured mitigation cost is a function of the service time which may vary depending on the requested file size, the degree of

concurrency of measurements requests, the server's configuration and the prototype implementation code. Therefore, an absolute value is used since otherwise a percentage value would imply normalization. In related works, the CoS is used as a metric in [53] and called "Client Success Ratio", while in [54] the used metric is equivalent to our defined RF whereas researchers in [48] consider service time in their metrics. Similarly, the metrics used in [57] 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 7, the service time for the shared IPs appears similar to the non-attacking sources, except with a temporary increase in service time, which is a kind of collateral damage despite of the 100% CoS.

Also, among the experiments' factors are the arbitrarily set parameters of the implemented prototype, such as, $\alpha$ and $C/\alpha\big|_{Thd}$. For example, a small $\alpha$ 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).

## 5.4 Attack Conditions

DDoS mitigation has been an endless battle where attack strategies evolve just as mitigation solutions do. Security threats are constantly evolving and will continue to evolve. The more complex the attack categories a defense can handle, the smarter the attackers will try. It's an endless game. No one can claim that a mitigation method requires no future evolution. However, it's possible to propose a framework, like the one presented here, with the necessary trust requirements of these days that in the past were less of a concern for users and organizations. The proposed framework is tested with today's complex attacks. As security threats evolve we argue that the proposed method offers a practical framework that can evolve as future types of application level misbehavior evolve, with no compromise between end-to-end data encryption and service availability.

In this research, we focus on the increasingly popular high-level DDoS attacks against web servers, more specifically complex HTTP(S)-DDoS attacks. Yet, it's also

necessary to discuss the possibility of a low-level DDoS attack. As an overlay-based method, assuming thousands of ANs and PSs, the proposed method renders all low-level based attacks ineffective, i.e., preventable. That includes new ones. 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, in contrast to a conventional web server, 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 4.1.3) compared to the PS prototype (sections 4.2.2 ~ 4.2.4). Therefore, the specialized PSs are expected to handle larger request rates than a conventional web server. In addition, with a proper DNS failover mechanism, multiple PSs can be sacrificed to stop a multi-Gbps massive attack of such kind far from the actual client-server communication. On the other hand, each AN expects only connections from specified source IPs to specified port numbers at specified times. Even if a low-level DDoS attack is towards a legitimately acquired SID, attack traffic is spread over multiple ANs beyond the attacker's choice, where any incomplete attempts, or high rate of $C/\alpha$, 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.

On the other hand, there's a variety of high-level attack possibilities. They vary in commonness and complexity. Mitigating complex HTTP(S)-DDoS attacks, far from the server, without inspecting the client-server traffic content is an especially challenging task. In the presented experiments, we investigated the mitigation of complex HTTP(S)-DDoS attack categories, such as; single-request per connection, multi-behavior per-shared-IP, sub-detection-thresholds, slow-requesting HTTP-DDoS, encrypted, and multi-vector attacks. Despite of the AN's short mitigation times, the attacker can still renew its session and use the additional detection time to affect the server. Such hypothetical scenario is possible and should be anticipated. In addition, the attacker may keep trying again repeatedly with different attack parameters combinations that may eventually be effective. Notice that an attack focused on the AN implies that the sources can successfully pass the PS preparation steps, thus revealing their true IP address and the targeted server. It also implies that the affected ANs are selected by the PSs, beyond the sources' choice. So, sources can't choose which AN to attack through on the high-level. Therefore, an attack source can't distribute its high-level attack connections over multiple ANs. In this case, the concerned AN with the introduced secure enhanced identification will have the attacking PID showing steady $S/\alpha$, rampant S/P, but small C/S = $C/\alpha$. So, once again it can be detected utilizing these attributes details that couldn't be seen conventionally without data inspection in the overlay.

There's a chance that an evolved HTTP(S)-DDoS attack sends multiple requests per connection, or multiple SIDs per PID (e.g., the traffic switching behavior). For

that, additional attributes must be utilized in detection, including $M/\alpha$ and $S/\alpha$. The attacker's options are limited; if the attacker reduces $M/\alpha$, $C/\alpha$ and $C/S$ to evade detection, then intuitively the rate of $S/\alpha$ and $S/P$ would rise. So, a PID with small $C/\alpha$, small $C/S$, and small $M/\alpha$, but large $S/\alpha$ would indicate the likelihood of suspicious behavior. Also, analyzing with a suitable machine learning algorithm the recorded values of $S/\alpha$ for the past $\beta$ 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 Section 4.2 can be extended to the other attack types once their respective attributes are considered.

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 Section 3.4.1 alone, even with automatic mitigation parameters tuning. In this worst case, a large part of the attack traffic is not detected by the AN, i.e., passed to the SS, exceeding its capacity. Then, the undetected traffic would require a supplementary detection component. Section 5.7 discusses the supplementary component as part of the actual implementation considerations, while Section 81 discusses its concept from a trust perspective.

Notice that complex HTTP(S)-DDoS attacks are inherently of low request rate per source. Let's say the attacker could gather 100,000 online distributed sources, a figure not reported before in a high-level attack, generating total about 17,000 encrypted HTTP GET requests per second (i.e., nearly 10 requests per minute per source, like the investigated case in section 4.2.6). Therefore, only about 100 ANs with just 97.8% RF (as in section 4.2.6) would be enough to reduce the attack to only 374 of the original 17,000 (i.e., DRR), while simultaneously serving other clients. Reducing the attack traffic reaching the SS by 97.8% would also reduce the supplementary detection load on the SS by the same amount. This reduction is crucial for the supplementary detection component to avoid being overwhelmed.

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. Large enough number of ANs and PSs would be required for this always-on assumption, which is made possible by the shared and modular nature of such components.

## 5.5 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 overlay-based methods may also offer non-split SSL as an option utilizing SNI, which we discuss about its limitations in Sect. 5.6.

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, assuming an additional local supplementary detection component (as discussed in Section 5.7) 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 [80]. 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 last 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 [26–28]. 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).

## 5.6 Conventional Solutions

Increasing the content's availability by utilizing existing commercial overlay-based methods, may achieve effective mitigation of HTTP(S)-DDoS attacks. However, this implies authorizing the third-party to manage the server's encryption parameters. Recent studies suggest that authorizing a third-party to manage the server's private key may not be acceptable by large sector of organizations and web users [26–28].

Instead, conventional overlay-based solutions can also offer non-split SSL as an option utilizing SNI. However, conventional overlay-based solutions with SNI suffer 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 overlay 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 conventional overlay-based solutions with SNI can be effective against massive low-level DDoS attacks, we argue that they can't effectively mitigate complex HTTP(S)-DDoS attack types such as; single-request per-connection, multi-behavior per-shared-IP, encrypted, and sub-detection-thresholds. This means that for conventional overlay-based solutions with SNI and non-split SSL, more attack categories would need to be detected locally, and stopped 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.

There's one important aspect to consider, for setting the ground for comparison with conventional methods. To compare two mitigation methods, it's mandatory to firstly specify the category of attack over which the comparison is constructed. If both methods to be compared can mitigate a complex category of attacks, then the points of the comparison would be shown in terms of the degree of mitigation (or MF), the speed of mitigation (or MT), etc. However, if one of the mitigation methods can't mitigate such complex attack category, then quantitative comparison would be pointless and hard to construct given the difference in scope.

Reviewing recent research works on the mitigation of HTTP(S)-DDoS in academia,

we are yet to see an overlay-based approach that assumes non-split end-to-end-encrypted client-server connections. In addition, certain complex attacks, which are missing from related works, cannot be detected by conventional methods given their common design assumptions. For example, single-request per-connection encrypted sub-detection-thresholds HTTP(S)-DDoS attack (such as in section 4.2.6). Yet, utilizing the introduced secure enhanced identification, the implemented proof-of-concept prototype shows high mitigation factors and low mitigation time, even for such attack condition, in contrast to related methods that inspect the content. On the other hand, even under less complex attacks, related research show lower mitigation performance than that of the proposed method. For example, the simulations by [53] with 280 attacker sources, show an 80% "Client Success Ratio" (i.e., CoS), while in [54], 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 [81] 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.

Although the concept of pre-service verification have been previously proposed, the key originality of our proposed verification (i.e., pre-service probing) technique is compatibility with existing standards without assuming impractical modifications, and compliance with the true end-to-end encryption requirement. In contrast to the pre-service verification discussed in in [41, 42], the system proposed in this dissertation requires no client-side or server-side special plugins for remote mitigation to work. Also, the proposed method in this dissertation requires no traffic decryption for embedding mitigation enabling script in the server' s response. Clients are tested using the proposed pre-service probing instead. Further, in [41, 42], the communication permission is granted locally at the server which adds a load on the server which is avoided in my method where the communication permission is granted externally by the overlay-nodes.

Also, the concept of client accounting itself (i.e., reputation) is not new [46, 47]. In contrast to related previous methods, the proposed method is the first to build a client reputation based on three levels of behavior attributes without assuming traffic decryption at the overlay nodes or special software at the client or server. Also, supplementary locally-based detection is only assumed for the small fraction traffic that's unmitigated far from the server.

As discussed in Chapters 1 and 2, there are many conventional DDoS mitigation solutions. Figure 5.2 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

| | 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 |

Figure 5.2 Proposed vs conventional solutions.

these assume either a split SSL connection between the client and server, or limited (per-IP and per-connection) identification, or both [29–31, 36]. 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.

## 5.7 Actual Implementation Considerations

The purpose of the current prototype is having a simplified initial working version of the proposed platform. Such platform readies the way for promising research by demonstrating, even with suboptimal prototype parameters, the soundness of the mitigation concept. However, in a real implementation, several changes should be considered.

Firstly, implementing a supplementary local detection component and feedback mechanism from the SS is a necessary feature to mitigate certain complex attacks. For example, an invalid request HTTPS-DDoS attack may appear similar to normal behavior to the AN on the low-level and therefore require a high-level feedback from the SS directly, or indirectly, to the AN.

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 section 5.5, we discuss it from the trust perspective.

Upon receiving a warning, the PSs and ANs can respond in several 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 existing approaches that employ scheduling policies to prioritize how clients receive service. Such approaches originally assume traffic decryption, limited identification, or both. For example, the concept considered in [50] can be integrated in our system, benefiting from enhanced identification, 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. So, the AN should implement three queues for arriving high-level messages; one queue for each reputation level, where good reputation PIDs receive the highest priority. This way, the cost observed by bad reputation PIDs would only increase when necessary in case of a high actual demand on the server from good users.

In addition, expanding the AN with an additional countermeasure against slow-requesting HTTP-DDoS to improve the mitigation factor during the mitigation time is a promising open research point. The AN can aggregate the received data bytes during $T_{a(max)} = \tau$ [ms], before contacting the SS, while the value of $\tau$ is modified according to the client-SS pair's local penalty. Relevant detection parameters here are; Warning feedback from the SS due to an incomplete or invalid request ($Warn_{SS}$), and the pair's reputation $R_L$ at the AN (which also propagates to the PS). The value of $R_L$ will deteriorate as the client's consistent behavior is recorded over multiple time steps, especially that this client can't distribute its behavior through multiple ANs to avoid detection. A combination of a bad $R_L$ and high $Warn_{SS}$ would indicate the likelihood of a suspiciously behaving PID.

Further, to counter the possible off/on switching behavior of attack traffic, an additional degree of AN based countermeasures should be added based on monitoring past records of $S/\alpha$, $S/P$, $FS/\alpha$, and $FS/P$ so that switching attack behaviors are identified on the PID level.

In addition, the current implementation code itself has room for optimization. For example, in the current code, for client-to-server and server-to-client data transfer, the AN uses a single thread for each. It serves the purpose for a proof of concept prototype, however, a real implementation should consider distributing each transfer direction's load over a pool of multiple threads and improving the thread code itself. This would improve the service time through the AN, especially during high traffic loads.

Furthermore, automatic parameter tuning via machine learning is also necessary. The proposed detection concept, which we implemented a manual simplified version of it, offers a detailed set of traffic parameters on the OID, PID, SID, CID and AID levels. This simplified version should be enhanced with existing approaches on integrating machine learning to application-level DDoS detection to achieve faster and higher mitigation.

Also, the unimplemented system parameters should be used. For instance, the global reputation $R_G$ was simply set to a fixed value of 1 throughout the conducted experiments. In a real implementation, $R_G$ should be managed by the mitigation service provider based on shared parameters from different ANs and PSs. Additionally, the formulas for obtaining the reputations in a real implementation should include $R_G$ as an input. Further, the value of the past records m was set to 4, while in reality a larger value should be used. In addition, the vectors $\hat{E}_P$ and $\hat{E}_S$ were only partially utilized on the 3 most significant bits, and different exceptions were combined into one. These vectors should be fully utilized, and different exceptions should also be considered individually. So, on the AN's SID level, a high rate of messages and a high rate of failed connection should have different weights in updating the penalty function.

# Chapter 6

# Conclusion

Protecting web servers against HTTP(S)-DDoS attacks should not contradict with the recent rise in awareness, among users and organizations, about traffic encryption due to trust concerns. This dissertation presented a new practical overlay-based DDoS mitigation concept that affirms this awareness. In order to effectively mitigate HTTP(S)-DDoS while complying with the encryption requirement, practical enhancement of overlay-based identification is investigated. To enable the desired enhanced identification, firstly a new overlay-based system is designed which practically introduces per-session identification to the conventional limited two-level identification. In contrast to conventional methods, the proposed system requires no client-side or server-side special plugins for remote mitigation to work. Then, a novel taxonomy of HTTP(S)-DDoS attacks is introduced organizing possible source behavior strategies from the overlay-nodes' perspective. In addition, a unique reputation and penalty system based on the enhanced behavior records is designed. The new introduced reputation system is based on three levels of behavior attributes without assuming traffic decryption at the overlay nodes or special software at the client or server. Moreover, the designed PS and AN are each equipped with two degrees of attack countermeasures, depending on their unique roles, offering a double-layered defense system against HTTP(S)-DDoS.

Additionally, a proof of concept prototype of the proposed system is implemented with simplified detection measures and attack countermeasures to demonstrate the concept's soundness, then tested for usability with actual commercial websites. Further, evaluations are conducted considering simple and conventionally hard to detect complex HTTP(S)-DDoS attack conditions. Among the conducted experiments, the results of seven experiments are presented and discussed, including; brute-force, below detection thresholds, single-request per-connection, slow-requesting HTTP-DDoS, multi-behavior per-shared-IP, encrypted, and multi-vector attack conditions. For attack traffic, attack tools popular among attackers are utilized (i.e., LOIC and

Slowloris) for experiments with limited number of sources (10 to 20 sources), and similarly built custom tools for highly distributed centrally controlled automated attacks (1,000 to 10,000 sources).

Evaluations results suggest that utilizing the introduced practical enhanced identification can eliminate the necessity for traffic decryption by overlay-nodes and for inspection of client-server traffic content, and enable an enhanced reputation and penalty system, which can accomplish high mitigation factors of conventionally hard to detect HTTP(S)-DDoS attack categories in relatively short mitigation times, in contrast to conventional overlay-based methods. In contrast to conventional methods, the proposed method is tested with complex attack conditions that are missing in related research such as single-request per-connection sub-detection-threshold HTTPS-DDoS. It suggests that less complex attack categories can be equally mitigated. In addition, results suggest that enhanced identification can achieve low collateral damage in terms of the chance of receiving service and service time for non-attacking clients that share an attack IP. However, the unoptimized implementation of the prototype system shows a cost in service time even in absence of attack. Also, it shows a temporarily decrease in mitigation factor and rise in cost during mitigation time. Experimentally demonstrating the concept's soundness based on the introduced per-session identification alone opens the way for investigating the inclusion of conventional per-IP and per-connection identification levels within various machine learning techniques for adaptive system parameters' tuning and for analyzing behavior record patterns.

# References


[1] M. S. A. Eid and H. Aida, "Securely Hiding the Real Servers from DDoS Floods," in 10th IEEE/IPSJ International Symposium on Applications and the Internet (SAINT), pp. 165–168, July 2010.

[2] 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.

[3] M. S. A. Eid, and H. Aida, "Secure Double-layered Defense against HTTP-DDoS Attacks", The 7th IEEE International COMPSAC Workshop on Network Technologies for Security, Administration and Protection (NETSAP2017), Turin, ITALY, Jul. 2017. In press.

[4] "McAfee Labs Threats Report." `http://www.mcafee.com/us/resources/reports/rp-quarterly-threats-mar-2016.pdf`, Mar. 2016. [Online; accessed 20-Apr-2017].

[5] "VeriSign Distributed Denial of Service Trends Report - Vol. 3, Q1 2016." `https://www.verisign.com/assets/report-ddos-trends-Q12016.pdf`, 2016. [Online; accessed 20-Apr-2017].

[6] Kaspersky: DDoS attacks in Q4 2016, `https://securelist.com/analysis/quarterly-malware-reports/77412/ddos-attacks-in-q4-2016/`, Feb 2017. [Online; accessed 28-Apr-2017].

[7] Incapsula: Global DDoS Threat Landscape -2015Q2 Security Report, `https://lp.incapsula.com/rs/804-TEY-921/images/DDoS%20Report%20Q2%202015.pdf`, 2015. [Online; accessed 28-Apr-2017].

[8] Incapsula Report: 2015-2016 Annual DDoS Threat Landscape Report, `https://www.incapsula.com/blog/2015-16-ddos-threat-landscape-report.html`, Aug 2016. [Online; accessed 28-Apr-2017].

[9] Neustar: Worldwide DDoS Attacks & Protection Report, https://ns-cdn.neustar.biz/creative_services/biz/neustar/www/resources/whitepapers/it-security/ddos/2016-fall-ddos-report.pdf, Oct 2016. [Online; accessed 28-Apr-2017].

[10] "Akamai: State of the Internet - 2015Q1 Security Report." https://www.akamai.com/us/en/about/news/press/2015-press/akamai-state-of-the-internet-security-report.jsp, 2015. [Online; accessed 20-Apr-2017].

[11] J. Lemon, "Resisting SYN Flood DoS Attacks with a SYN Cache," in Proceedings of the BSD Conference, BSDC'02, (Berkeley, CA, USA), pp. 10–10, 2002.

[12] "Apache mod_evasive." https://www.linode.com/docs/web-servers/apache-tips-and-tricks/modevasive-on-apache. [Online; accessed 20-Apr-2017].

[13] "Apache Security Tips." http://httpd.apache.org/docs/trunk/misc/security_tips.html. [Online; accessed 20-Apr-2017].

[14] "A10 Networks: Threat Protection Systems." https://www.a10networks.com/products/thunder-series/thunder-tps-ddos-protection, 2016. [Online; accessed 20-Apr-2017].

[15] J. McCallion, "A10 storms the DDoS prevention market with Thunder TPS." http://www.itpro.co.uk/security/21393/a10-storms-the-ddos-prevention-market-with-thunder-tps, 2014. [Online; accessed 20-Apr-2017].

[16] M. Walfish, M. Vutukuru, H. Balakrishnan, D. Karger, and S. Shenker, "DDoS Defense by Offense," in Proceedings of the 2006 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications, SIGCOMM '06, (New York, NY, USA), pp. 303–314, 2006.

[17] M. Walfish, M. Vutukuru, H. Balakrishnan, D. Karger, and S. Shenker, "DDoS Defense by Offense," ACM Trans. Comput. Syst., vol. 28, pp. 3:1–3:54, Aug. 2010.

[18] R. Mahajan, S. M. Bellovin, S. Floyd, J. Ioannidis, V. Paxson, and S. Shenker, "Controlling High Bandwidth Aggregates in the Network," SIGCOMM Comput. Commun. Rev., vol. 32, pp. 62–73, July 2002.

[19] J. Ioannidis and S. M. Bellovin, "Implementing Pushback: Router-Based Defense Against DDoS Attacks," in In Proceedings of Network and Distributed System Security Symposium, 2002.

[20] X. Xu, Y. Sun, and Z. Huang, "Defending DDoS Attacks using Hidden Markov Models and Cooperative Reinforcement Learning," Intelligence and Security Informatics, pp. 196–207, 2007.

[21] H. V. Nguyen and Y. Choi, "Proactive Detection of DDoS Attacks Utilizing k-NN Classifier in an Anti-DDos Framework," International Journal of Electrical, Computer, and Systems Engineering, vol. 4, no. 4, pp. 247–252, 2010.

[22] M. Casado, A. Akella, P. Cao, N. Provos, and S. Shenker, "Cookies Along Trust-Boundaries (CAT): Accurate and Deployable Flood Protection," in Proceedings of the 2nd conference on Steps to Reducing Unwanted Traffic on the Internet - Volume 2, pp. 3–3, USENIX Association, 2006.

[23] J. Li, S. Berg, M. Zhang, P. Reiher, and T. Wei, "Drawbridge: Software-defined DDoS-resistant Traffic Engineering," SIGCOMM Comput. Commun. Rev., vol. 44, pp. 591–592, Aug. 2014.

[24] G. S. Mengle, "Cyber attack on ISPs", `http://www.thehindu.com/todays-paper/tp-national/tp-mumbai/Cyber-attack-on-ISPs-police-file-FIR/article14507031.ece`, Jul 2016. [Online; accessed 28-Apr-2017].

[25] D. Raywood, "Australian ISP Fights DDoS Attack", `https://www.infosecurity-magazine.com/news/australian-isp-fights-ddos-attack/`, Apr 2017. [Online; accessed 28-Apr-2017].

[26] R. Goldberg, "Lack of Trust in Internet Privacy and Security May Deter Economic and Other Online Activities." https://www.ntia.doc.gov/blog/2016/lack-trust-internet-privacy-and-security-may-deter-economic-and-other-online-activities, 2016. [Online; accessed 20-Apr-2017].

[27] Centre for International Governance Innovation (CIGI), IPSOS, Internet Society, United Nations Conference on Trade & Development(UNCTAD), International Development Research Center (IDRC), "2017 CIGI-Ipsos Global Survey on Internet Security and Trust", `https://www.cigionline.org/internet-survey`, Apr. 2017. [Online; accessed 6-May-2017].

[28] "Thales e-Security: Encryption Application Trends Study." `https://www.thales-esecurity.com/knowledge-base/analyst-reports/encryption-application-trends-study-2016`, June 2016. [Online; accessed 20-Apr-2017].

[29] Akamai, "Kona DDoS defender", `https://www.akamai.com/us/en/multimedia/documents/product-brief/akamai-kona-site-defender-product-brief.pdf`. [Online; accessed 28-Apr-2017].

[30] VeriSign, "DDoS protection services", https://www.verisign.com/en_US/security-services/ddos-protection/ddos-service/index.xhtml. [Online; accessed 28-Apr-2017].

[31] CloudFlare, "SSL Options", https://support.cloudflare.com/hc/en-us/articles/200170416-What-do-the-SSL-options-mean-. [Online; accessed 28-Apr-2017].

[32] "VeriSign: DDoS Protection Services." https://www.verisign.com/en_GB/security-services/ddos-protection/index.xhtml, 2016. [Online; accessed 20-Apr-2017].

[33] "CloudFlare: Plans." https://support.cloudflare.com/hc/en-us/articles/200170326-How-much-does-the-Enterprise-Plan-cost-, 2016. [Online; accessed 20-Apr-2017].

[34] M. S. A. Eid, "DDoS Avoidance by Securely Hiding Web Servers." Master's thesis, The University of Tokyo, 2010. http://hdl.handle.net/2261/37658.

[35] Arbor: Worldwide Infrastructure Security Report - Vol. XI, https://www.arbornetworks.com/images/documents/WISR2016_EN_Web.pdf, 2016. [Online; accessed 28-Apr-2017].

[36] AWS Best Practices for DDoS Resiliency, https://d0.awsstatic.com/whitepapers/Security/DDoS_White_Paper.pdf, June 2016. [Online; accessed 28-Apr-2017].

[37] T. Benzel, "The science of cyber security experimentation: The deter project," in Proceedings of the 27th Annual Computer Security Applications Conference, ACSAC '11, pp. 137–148, ACM, 2011.

[38] N. I. Mowla, I. Doh, and K. Chae, "Multi-defense mechanism against ddos in sdn based cdni," in 2014 Eighth International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing, pp. 447–451, July 2014.

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

[40] Y. Gilad, A. Herzberg, M. Sudkovitch, and M. Goberman, "Cdn-on-demand: An affordable ddos defense via untrusted clouds," In Network and Distributed System Security Symposium (NDSS), Feb. 2016.

[41] S. H. Khor and A. Nakao, "spow: On-demand cloud-based eddos mitigation mechanism," in Proc. of the 5th Workshop on Hot Topics in System Dependability (HotDep), pp. 1–6, 2009.

[42] S. H. Khor and A. Nakao, "Daas: Ddos mitigation-as-a-service," in IEEE/IPSJ 11th International Symposium on Applications and the Internet (SAINT), pp. 160 –171, jul 2011.

[43] P. Du and A. Nakao, "Ddos defense as a network service," in IEEE Network Operations and Management Symposium (NOMS), pp. 894 –897, Apr. 2010.

[44] W. Alosaimi and K. Al-Begain, "An enhanced economical denial of sustainability mitigation system for the cloud," in 2013 Seventh International Conference on Next Generation Mobile Apps, Services and Technologies, pp. 19–25, Sept 2013.

[45] 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, Apr. 2014.

[46] P. Du and A. Nakao, "Overcourt: Ddos mitigation through credit-based traffic segregation and path migration," Computer Communications, vol. 33, no. 18, pp. 2164 – 2175, 2010.

[47] P. Du and A. Nakao, "Mantlet trilogy: Ddos defense deployable with innovative anti-spoofing, attack detection and mitigation," in 2010 Proceedings of 19th International Conference on Computer Communications and Networks, pp. 1–7, Aug 2010.

[48] A. Krizhanovsky, "Tempesta: A Framework for HTTP DDoS Attacks Mitigation," in Proceedings of 24th Annual International Conference on Computer Science and Software Engineering, CASCON '14, pp. 148–162, IBM Corp., 2014.

[49] Y. Xie and S.-Z. Yu, "A large-scale hidden semi-markov model for anomaly detection on user browsing behaviors," IEEE/ACM Trans. Netw., vol. 17, pp. 54–65, Feb. 2009.

[50] 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, Feb. 2009.

[51] Y. Xie and S.-Z. Yu, "Monitoring the application-layer ddos attacks for popular websites," IEEE/ACM Trans. Netw., vol. 17, pp. 15–25, Feb. 2009.

[52] S. Khanna, S. S. Venkatesh, O. Fatemieh, F. Khan, and C. A. Gunter, "Adaptive Selective Verification: An Efficient Adaptive Countermeasure to Thwart DoS Attacks," IEEE/ACM Trans. Netw., vol. 20, pp. 715–728, June 2012.

[53] 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. 75–82, Sept 2014.

[54] H. Beitollahi and G. Deconinck, "Connectionscore: a statistical technique to resist application-layer ddos attacks," Journal of Ambient Intelligence and Humanized Computing, vol. 5, no. 3, pp. 425–442, 2014.

[55] Q. Wu, S. Shiva, S. Roy, C. Ellis, and V. Datla, "On Modeling and Simulation of Game Theory-based Defense Mechanisms Against DoS and DDoS Attacks," in Proceedings of the 2010 Spring Simulation Multiconference, SpringSim '10, pp. 159:1–159:8, 2010.

[56] Y. Liu, D. Feng, Y. Lian, K. Chen, and Y. Zhang, Optimal Defense Strategies for DDoS Defender Using Bayesian Game Model, pp. 44–59. Springer Berlin Heidelberg, 2013.

[57] T. Spyridopoulos, G. Karanikas, T. Tryfonas, and G. Oikonomou, "A game theoretic defense framework against dos/ddos cyber attacks," Computers & Security, vol. 38, pp. 39 – 50, 2013.

[58] C. B. Simmons, S. G. Shiva, H. S. Bedi, and V. Shandilya, ADAPT: A Game Inspired Attack-Defense and Performance Metric Taxonomy, pp. 344–365. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013.

[59] M. Wright, S. Venkatesan, M. Albanese, and M. P. Wellman, "Moving target defense against ddos attacks: An empirical game-theoretic analysis," in Proceedings of the 2016 ACM Workshop on Moving Target Defense, MTD '16, (New York, NY, USA), pp. 93–104, ACM, 2016.

[60] 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 Communications, vol. 36, no. 3, pp. 303 – 319, 2013.

[61] W. Zhou, W. Jia, S. Wen, Y. Xiang, and W. Zhou, "Detection and Defense of Application-Layer DDoS Attacks in Backbone Web Traffic," Future Generation Computer Systems, vol. 38, pp. 36 – 46, 2014.

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

[63] M. Jog, M. Natu, and S. Shelke, "Distributed and Predictive-Preventive Defense Against DDoS Attacks," in Proceedings of the 2015 International Conference on

Distributed Computing and Networking, ICDCN '15, (New York, NY, USA), pp. 29:1–29:4, ACM, 2015.

[64] V. Aghaei-Foroushani and A. N. Zincir-Heywood, "Investigating unique flow marking for tracing back DDoS attacks," in 2015 IFIP/IEEE International Symposium on Integrated Network Management (IM), pp. 762–765, May 2015.

[65] "The Network Simulator - ns-2." http://www.isi.edu/nsnam/ns/. [Online; accessed 14-March-2016].

[66] J. Mirkovic, S. Fahmy, P. Reiher, and R. K. Thomas, "How to Test DoS Defenses," in Conference For Homeland Security, 2009. CATCH '09. Cybersecurity Applications Technology, pp. 103 –117, Mar. 2009.

[67] "OPNET Academic Edition." http://www.opnet.com/university_program/itguru_academic_edition/. [Online; accessed 14-March-2016].

[68] B. Van den Broeck, P. Leys, J. Potemans, J. Theunis, E. Van Lil, and A. Van de Capelle, "Validation of Router Models in OPNET," OPNETWORK 2002, Washington D.C., USA, 2002., 2002.

[69] "PlanetLab." https://www.planet-lab.org/. [Online; accessed 20-Apr-2017].

[70] Akamai: State of the Internet - 2016Q1 Security Report, Vol. 3, No. 1, https://www.akamai.com/es/es/multimedia/documents/state-of-the-internet/akamai-q1-2016-state-of-the-internet-security-report.pdf, . [Online; accessed 28-Apr-2017].

[71] "NewEraCracker / LOIC." https://github.com/NewEraCracker/LOIC/releases/tag/1.1.1.25, 2011. [Online; accessed 14-May-2016].

[72] "Worldwide Infrastructure Security Report - Volume X – 2015." https://www.arbornetworks.com/arbor-networks-10th-annual-worldwide-infrastructure-security-report-finds-50x-increase-in-ddos-attack-size-in-past-decade, 2015. [Online; accessed 20-Apr-2017].

[73] "Incapsula Attack Glossary: Slowloris." https://www.incapsula.com/ddos/attack-glossary/slowloris.html. [Online; accessed 14-March-2016].

[74] Hypertext Transfer Protocol – HTTP/1.1: Connections (RFC2616, Sec.8).

[75] DETERLab Node Types, http://docs.deterlab.net/core/node-types/. [Online; accessed 28-Apr-2017].

[76] Radware: Common DDoS Attack Tools, https://security.radware.com/ddos-knowledge-center/ddos-attack-types/common-ddos-attack-tools/, Jan 2016. [Online; accessed 28-Apr-2017].

[77] D. Mahajan and M. Sachdeva, "Distinguishing ddos attack from flash event using real-world datasets with entropy as an evaluation metric," in 2013 International Conference on Machine Intelligence and Research Advancement, pp. 90–94, Dec 2013.

[78] K. Johnson Singh, K. Thongam, and T. De, "Entropy-based application layer ddos attack detection using artificial neural networks," Entropy, vol. 18, no. 10, 2016.

[79] S. Behal and K. Kumar, "Trends in validation of ddos research," Procedia Computer Science, vol. 85, pp. 7 – 15, 2016.

[80] European Parliament resolution P8_TA(2015)0388 on the electronic mass surveillance of EU citizens, `http://www.europarl.europa.eu/sides/getDoc.do?pubRef=-//EP//NONSGML+TA+P8-TA-2015-0388+0+DOC+PDF+V0//EN`, Oct 2015. [Online; accessed 28-Apr-2017].

[81] LEVEL 3® DDoS MITIGATION, `http://www.level3.com/-/media/files/brochures/en_secur_br_ddos_mitigation.pdf`. [Online; accessed 28-Apr-2017].

# Publications

### Journal Paper

- 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.

### Proceedings of International Conferences

- M. S. A. Eid, and H. Aida, "Secure Double-layered Defense against HTTP-DDoS Attacks", The 7th IEEE International COMPSAC Workshop on Network Technologies for Security, Administration and Protection (NETSAP2017), Turin, ITALY, Jul. 2017. In press.
- 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, `https://doi.org/10.1109/SAINT.2010.62`, Seoul, KOREA, Jul. 2010.

### Other Presentations

- M. S. A. Eid, and H. Aida, "Enabling True End-to-End Encryption and Client-Behavior Identification in Overlay-based DDoS Mitigation", IEICE Tech. Rep. on 34th Cyberworlds, CW2016-09, pp.19-23, Tokyo, JAPAN, Dec. 2016.
- M. S. A. Eid, and H. Aida, "Overlay Based, Distributed Defense-Framework against DDoS Attacks", IEICE Tech. Rep., vol. 111, no. 347, IA2011-51, pp. 37-42, `http://www.ieice.org/ken/paper/20111216v0LT/eng/`, Hiroshima, JAPAN, Dec. 2011.
- M. S. A. Eid, and H. Aida "DDoS Attacks Avoidance by Securely Hiding Web Servers", 26th Annual Computer Security Applications Conference (ACSAC 26),

https://www.acsac.org/2010/program/posters/eid.pdf Austin, TX, USA, Dec. 2010. (Poster)