FI MU

Faculty of Informatics
Masaryk University Brno

Traffic characteristics of common DoS tools

by

Vít Bukač

FI MU Report Series **FIMU-RS-2014-02**

Copyright © 2014, FI MU **April 2014**

**Copyright © 2014, Faculty of Informatics, Masaryk University.
All rights reserved.**

**Reproduction of all or part of this work
is permitted for educational or research use
on condition that this copyright notice is
included in any copy.**

Publications in the FI MU Report Series are in general accessible via WWW:

`http://www.fi.muni.cz/reports/`

Further information can be obtained by contacting:

**Faculty of Informatics
Masaryk University
Botanická 68a
602 00 Brno
Czech Republic**

Traffic characteristics of common DoS tools

Vít Bukač
Faculty of Informatics, Masaryk University, Brno, CZ
bukac@mail.muni.cz

April 6, 2014

Abstract

Denial of service (DoS) attacks is an ever growing threat to the availability of computer systems. Numerous solutions have been proposed both for DoS attacks detection and mitigation. However, their evaluation and mutual comparison is complicated due to scarcity of representative contemporary input data. In academia, proposed DoS detection systems are frequently evaluated with obsolete and in practice no longer used tools. Such discrepancy can lead to distinctly different detection efficiency in evaluation environment and real environment. To address this issue, we provide a comparative analysis of traffic features of DoS attacks that were generated by state-of-the-art standalone DoS attack tools. We list frequently used traffic features and verify their presence in analyzed attack traffic. Common denominator of all attack traffic is the presence of repeated similar yet independent operations. Therefore, we propose a new research area for the detection of DoS attacks the source end, based on repeated attack patterns recognition.

Contents

1 Introduction **7**

1.1 Related work 7

1.2 Experiment 8

1.2.1 Environment 8

1.2.2 Measurement principle 9

1.2.3 Processes 10

1.3 DoS Tools 11

1.3.1 Tools selection 11

1.3.2 Capabilities 12

1.3.3 Binaries 15

2 Configurations **15**

3 IP behavior **27**

3.1 Traffic volume 27

3.1.1 Packet rate 27

3.1.2 Byte rate 27

3.1.3 Attack buildup 27

3.1.4 Byte rate and packet rate relationship 30

3.2 Packet rate burst behavior 31

3.3 IP fragmentation 33

3.4 IP spoofing 33

3.5 Average packet size 35

3.6 Packet size distribution 36

3.7 Packet incoming to outgoing ratio 38

4 TCP behavior **41**

4.1 Flow count 41

4.2 Flow parallelity 43

4.3 Flow packet count 45

4.4 TCP flag ratios 50

4.4.1 SYN outgoing to SYNACK incoming 50

4.4.2 FIN segments to all segments ratio 50

4.4.3 RST segments to all segments ratio 53

4.4.4 NS, ECE, CWR, URG to all TCP segments ratio 56
4.5 Average flow duration 56

5 HTTP behavior **58**
5.1 HTTP requests success 58
5.2 HTTP requests per flow 63
5.3 HTTP request method 65
5.4 HTTP requests URIs 66
5.5 HTTP header fields 68
5.5.1 User-Agent 68
5.5.2 Referer 70
5.5.3 Accept-Encoding and Accept-Language 71

6 Summary **74**
6.1 Attacks diversity 74
6.2 Traffic features 74
6.3 Repeating patterns 75
6.4 Evasion techniques 76
6.5 Experiment designs 76
6.6 Tools characteristics 76

7 Conclusions **77**

List of Tables

1 DoS tools analyses . . . 8
2 DoS tools selection . . . 13
3 Tools capabilities . . . 14
4 DoS tool binaries . . . 16
5 TCP Attack configurations Part 1 . . . 17
6 TCP Attack configurations Part 2 . . . 18
7 TCP Attack configurations Part 3 . . . 19
8 HTTP Attack configurations Part 1 . . . 20
9 HTTP Attack configurations Part 2 . . . 21
10 HTTP Attack configurations Part 3 . . . 22
11 HTTP Attack configurations Part 4 . . . 23
12 HTTP Attack configurations Part 5 . . . 24
13 HTTP Attack configurations Part 6 . . . 25
14 HTTP Attack configurations Part 7 . . . 26
15 Average packet rate . . . 28
16 Average byte rate . . . 29
17 Packet rate burstiness . . . 33
18 Average packet size with limit – TCP configurations . . . 36
19 Average packet size with limit – HTTP configurations . . . 37
20 Packet size distribution – TCP . . . 39
21 Packet size distribution – HTTP . . . 40
22 Limited – Interesting values – approximate . . . 41
23 Flow count in 60 s interval – TCP . . . 43
24 Flow count in 60 s interval – HTTP . . . 43
25 Flows parallelity – TCP . . . 46
26 Flows parallelity – HTTP . . . 46
27 Flows attacker packet count – HTTP . . . 48
28 Flows attacker packet count – TCP . . . 48
29 SYN outgoing to SYNACK incoming TCP segments ratio . . . 51
30 FIN segments to all segments ratio . . . 54
31 RST segments to all segments ratio . . . 56
32 Flow duration – Summary . . . 59

33 Flow duration Part 1 60
34 Flow duration Part 2 61
35 Flow duration Part 3 62
36 HTTP requests success 63
37 HTTP requests per flow 64
38 HTTP requests method 65
39 HTTP requests URIs 66
40 HTTP requests URIs modifications 68
41 HTTP requests User-Agents 71
42 HTTP requests header fields – Referer 72
43 HTTP requests header fields – Accept-Encoding 72
44 HTTP requests header fields – Accept-Language 73

List of Figures

1 Attack buildup . . . 30
2 Packet rate . . . 31
3 Byte rate . . . 31
4 Packet rate and byte rate relationship anomalies . . . 32
5 Packet rate burstiness – Full burstiness example . . . 34
6 Packet rate burstiness – Regular peaks example . . . 34
7 Packet rate burstiness – One-time extreme example . . . 34
8 Flow count example . . . 44
9 Flow packet count histogram example . . . 49
10 SYN outgoing to SYNACK incoming TCP segments ratio – Stability example . . . 52
11 SYN outgoing to SYNACK incoming TCP segments ratio – Initiation phase example . . . 52
12 SYN outgoing to SYNACK incoming TCP segments ratio – IP spoofing present example . . . 52
13 FIN segments to all segments ratio – Stability example . . . 55
14 FIN segments to all segments ratio – Anomaly example . . . 55
15 FIN segments to all segments ratio – Repeated pattern . . . 55
16 RST segments to all segments ratio – Stability example . . . 57
17 RST segments to all segments ratio – Planks example . . . 57
18 RST segments to all segments ratio – Anomaly example . . . 57
19 HTTP request per flow count . . . 65
20 HTTP request URIs – Unique HTTP request count . . . 69
21 HTTP request URIs – Unique HTTP request without parameters count . . . 69

1 Introduction

This work has three major contributions:

- An overview of existing standalone DoS attack tools, their attack traffic properties and used evasion techniques. Network traffic profiles of standalone DoS tools will help design detection methods that are based on valid assumptions in the future. Also, by creating a database of attack tools in the wild, it will be possible to estimate which classes of DoS attacks can be detected by each proposed method.
- Identification of network traffic features that are suitable for the source-end DoS attack detection. Evaluate the importance of selected feature for various classes of DoS attacks. Due to difficulties with packet recording at high DoS tools performance settings, the measurement was focused on tools' capabilities and traffic features, not performance comparison.
- Support for the use of state-of-the-art tools for evaluation of DoS intrusion detection systems in academic research. Share discovered properties, tools and experiment design with researchers to facilitate mutually comparable results.

1.1 Related work

Even though DDoS attacks are steadily gaining on popularity both among cyber criminals and among security researchers, there are only few studies focusing on the thorough characteristics of DDoS attack traffic. We observe a serious discrepancy between tools that are being used by attack perpertrators and the tools that are being used for testing of DDoS detection and mitigation solutions proposed by academia. The list of tools and techniques actively used in real environment contains advanced tools such as LOIC, HOIC or Slowloris. Conversely, academia solutions are notoriously evaluated with long obsolete and in practice already forgotten tools, most notably TFN, TFN2k, Shaft, Trinoo, Knight, mstream and Stacheldraht which all date back to year 2000. We still encounter numerous research works which present these tools as representatives of state-of-the-art DDoS attacks, even in respectable periodics (i.e., [OG10], [BKBK13], [AMG+12], [YKP+13]).

Relatively few surveys has been devoted to current trending DoS attack tools. Thing et al. [TSD07] performed a detailed analysis of source code of selected popular DDoS attack bots, namely Agobot, SDBot, RBot and Sybot. Authors emphasize the importance

of randomization in creating the packet which is a view we share. Given the availability of the source code, the analysis is very descriptive with deep understanding of inner works of each tool, but the analysis does not provide a high-level overview of the traffic in real situations.

Another study aimed at properties of DDoS bots has been performed by Jeff Edwards and Jose Nazario [EN11]. The study focuses on families of DDoS botnet malware controlled predominantly from Chinese IP space. An exhaustive summary of characteristics is provided for bot communication protocols and command grammars. Attacks supported by each bot are listed along with a high-level attack type taxonomy. Statistics of attack types and DDoS targets distribution by countries is shown. However, from the perspective of attack traffic characteristics, only few unique properties of chosen bots are discussed.

Some of the most prominent DoS tools are occasionally examined by freelance security specialists or companies dealing in DDoS protection solutions. Such analyses are often thorough and descriptive, but lack mutual comparison and frequently are focused on the tools themselves, without deriving general concepts. Table 1 lists some of the freely available analyses of tools that were also included in our survey.

Table 1: DoS tools analyses

HOIC	spiderlabs.com prolexic.com
HULK	prolexic.com
LOIC	computerbiology.blogspot.cz spiderlabs.com
OWASP HTTP tool	owasp.org
Slowloris	slashroot.in ckers.org

1.2 Experiment

1.2.1 Environment

Analysis was performed in a controlled virtual environment with only minimal background traffic. Virtual environment was used in order to minimize the influence of real intermediate network on measurements. Also, snapshot feature of virtual machines allows returning to a conjoint initial stable state. Therefore, subsequent measurements are not affected by artifacts from previous measurements (e.g., keep-alive packets sent by either side).

Two virtual machines were created. Windows 7 machine designated ATTACKER and Windows Server 2008 R2 designated VICTIM. Both virtual machines were fully updated. Wireshark packet sniffer, 7zip file archiver, Python 2.7.5 and ActivePerl 5.16.3 were installed on ATTACKER. Web server role was added to VICTIM. Tested DoS tools were copied to ATTACKER in separate encrypted ZIP archives. After software and server system features were installed a snapshot of each machine was created in order to get an initial state for measurement.

Internal network was created between the two virtual machines. Firewalls on both machines were configured to allow all incoming traffic from the shared network. Settings for other subnets were kept default. Except of DoS attack tools and operating system itself no other legitimate network traffic was produced. However, due to the nature and dubious origin of some DoS tools, several tested tools may have contained Trojan malware. DoS tools that provably contained malware were not included in the test. Collected packet traces were not tested for the presence of malware network traffic. Tools were executed by a member of Administrators group with UAC enabled.

As a testing target page a CNN.com webpage from 11/19/2012 19:39 UTC was used, renamed to index.htm. Popular existing webpage was selected in order to mimic real conditions under which DoS tools are launched. Saved webpage has 109 files. Total size is 3.3 MB including images. The target webserver was IIS 7.0 installed on Windows Server 2008 R2.

1.2.2 Measurement principle

DoS tools were executed in a common initial state. Both outgoing and incoming network traffic was recorded with dumpcap tool from Wireshark suite. Sixty-second and three hundred-second traffic samples were obtained for every tool configuration. Analysis was performed offline on collected PCAP files. Analysis consisted of two parts. First, the traffic was divided to 1 second intervals. Network features statistics (e.g., byterate, packetrate, TCP flag ratios) were then computed for each interval. Second, the PCAP file was process packet by packet and network flows were reconstructed and flow statistics were computed (e.g., simultaneous flow count, packets per flow). Graphs on the following pages represent values of respective metrics each second of the first minute of the attack.

1.2.3 Processes

First tool execution

1. Shutdown both machines.
2. Restore initial state of both virtual machines.
3. Unpack tested tool.
4. Execute tool. If the tool is a script, use appropriate command interpreter.
5. Note if the tool requires administrative rights to execute.
6. Note if the tool requires installation of additional files.
7. Take screenshot of launched tool.

Network traffic acquisition

1. Shutdown both virtual machines.
2. Restore initial state of both virtual machines.
3. Perform steps necessary to execute the tool (e.g., elevate privileges, register additional OCX files).
4. Execute tool. Only one DoS attack tool can be executed during each network traffic acquisition.
5. Configure the tested settings of the tool.
6. Take screenshot of the settings and save as %TOOLNAME%_%ATTACKTYPE%_%INDEX%.png
7. Start network traffic sniffer with command dumpcap -w %TOOLNAME%_%ATTACKTYPE%_%INDEX%.pcap -a duration:60
8. Timeout runs out. Dumpcap finishes.
9. Stop attack tool.
10. Note the percentage of traffic recorded by dumpcap.
11. Copy pcap and png files to storage.

1.3 DoS Tools

1.3.1 Tools selection

For analysis a subset of existing DoS tools were selected based on their popularity and capabilities. Emphasis is on tools that were used or allegedly used during publicized DDoS campaigns, usually related to Anonymous movement and hacktivism, are a popular choice on public hacker forums, or are created as open source and available in public software repositories. Tools were selected in order to represent a full spectrum of existing types of TCP and HTTP DoS attacks. Tools that may be little popular, but take an extraordinary approach in causing a DDoS effect were added. Several versions of iconic LOIC tool were included in order to track the development in time.

Since our study is aimed at standalone DoS tools, any attack tools that either require controlled bots or perform reflection attacks with help of unknowing middle-men were discarded from the selection.

Hacktivism

Hacktivism is the use of computers to manifest political opinions. By performing denial of service attacks hacktivists may express their disagreement with a targeted organization. Large hacktivist operation can target many organizations simultaneously and can span over period of several months. The operation usually starts with public announcement and subsequent recruitment of individuals willing to participate in the protest. Tools to be used during the operation are either mentioned directly in the announcement or are suggested during chats of organizers with recruits.

Operation Israel was announced on November 14, 2012 by members of Anonymous as a response on military action performed by Israel Defense Forces in Gaza Strip [Unk12]. One of the aims of this operation was performing distributed DoS attacks against Israel web sites. Company Radware identified 7 DoS tools that were suggested by attack coordinators and other active participants: ByteDoS 3.2, Mobile LOIC, LOIC for android devices, Tor's Hammer, Slowloris, PyLoris and THC SSL DOS [EYA12].

On May, 1 2013 US Department of Homeland Security warned against a planned cyber-attack campaign against US Government agencies, financial institutions and commercial organizations [oIA13]. This campaign was dubbed OpUSA. Later report from US National Cybersecurity and Communications Integration Center announced a list of DoS tools that are often used by hacktivist and could be used during the campaign [Nat13]. List includes: LOIC, HOIC, HULK, Slowloris, DDos Notepad, ByteDOS,

Turbinas, Syn Flood DOS, Jays Booter, HTTPFlooder, Torshammer, R.U.D.Y., OWASP HTTP Tool, Anonymous DOSer, Windows_DNS_Attack_Tool and GoodBye.

Operation Myanmar is a hacktivist campaign started in December 2012 by group DangerHackers [Tea12]. This campaign is a protest against killing of Muslim Rohingya in Myanmar. Operation declaration contains a list of target webpages and DDoS tools to be used against them. Advertised DoS tools are: LOIC, FireFlood, Anonymous DoSer, HOIC, ByteDOS and UDPUnicorn.

Other sources

Respected security companies that provide anti-DDOS solutions often publish lists of DDoS tools that are either trending or present a new step in development of DoS tools. These lists provide an up-to-date insight into what attack variants are popular and what attack tools can most likely be faced. Pavitra Shankdhar in his report for InfoSec institute mentions these tools: LOIC, XOIC, HULK, DDOSIM, R.U.D.Y, Tor's Hammer, PyLoris, OWASP DDoS HTTP POST, DAVOSET and GoldenEye HTTP Denial of Service Tool [Sha13]. Another comprehensive list from Curt Wilson of Arbor Networks focuses mostly on bots and paid DDoS services, but it also includes several standalone DoS tools, most notably Drop-Dead DDoS, AlbaDDOS, Manta d0s, GoodBye, PHPDOS and Janidos [Wil12].

An important source of feedback for DoS tool creators are hacker forums. During our monitoring of hackforums.net we have identified following tools to be popular, freely available and fairly advanced: Longcat, UnknownDoser. Last remaining tools were added because they are representatives of extraordinary approaches to DoS attacks and should be included both for attack spectrum completeness and to provide more data sources for comparison.

1.3.2 Capabilities

For each selected tool one release version was obtained, preferably the version that was advertised in sources that we examined. For AnonymousDOS, BanglaDOS, HTTPFlooder, Janidos, SimpleDoSTool and Syn Flood DOS we were unable to determine the version number, probably because its releases were one-time. Due to the popularity of LOIC several different samples have been collected in order to map the changes in time.

Most standalone DoS tools are single-purpose programs that are capable of only one type of attack. Moreover, even tools that support multiple attack types rarely can launch

Table 2: DoS tools selection

Name	Source info
Anonymous DoSer	OpUSA, OpMyanmar
AnonymousDOS	Representative
BanglaDOS	Representative
ByteDOS	OpIsrael, OpUSA, OpMyanmar
DoS	Representative
FireFlood	OpMyanmar
Goodbye	OpUSA, Arbor Networks report
HOIC	OpUSA, OpMyanmar
HULK	OpUSA, InfoSec report
HTTP DoS Tool	Representative
HTTPFlooder	OpUSA
Janidos -Weak edition-	Arbor Networks report
JavaLOIC	OpUSA, OpMyanmar, InfoSec report
LOIC	OpUSA, OpMyanmar, InfoSec report
Longcat	Hackforums
OWASP HTTP Tool	OpUSA, InfoSec report
SimpleDoSTool	Representative
Slowloris	OpIsrael, OpUSA
Syn Flood DOS	OpUSA
TORSHAMMER	OpIsrael, OpUSA, InfoSec report
UnknownDoser	Hackforums
XOIC	InfoSec report

Table 3: Tools capabilities

Name	Version	Source	Tool ID	Attacks
Anonymous DoSer	2.0		AD	HTTP
AnonymousDOS		Script	ADR	HTTP
BanglaDOS			BAD	HTTP
ByteDOS	3.2		BD	SYN, ICMP
DoS	5.5		DS	TCP
FireFlood	1.2		FF	HTTP
Goodbye	3.0		GB3	HTTP
Goodbye	5.2		GB5	HTTP
HOIC	2.1.003		HO	HTTP
HULK	1.0	Script	HU	HTTP
HTTP DoS Tool	3.6	YES	HDT	slow headers, slow POST
HTTPFlooder			HF	HTTP
Janidos -Weak edition-			JA	HTTP
JavaLOIC	0.0.3.7	YES	JL	TCP, UDP, HTTP
LOIC	1.0.4.0	YES	LO1	TCP, UDP, HTTP
LOIC	1.0.7.42	YES	LO2	TCP, UDP, HTTP
LOIC	1.1.1.25	YES	LO3	TCP, UDP, HTTP
LOIC	1.1.2.0b	YES	LO4	TCP, UDP, HTTP, ReCoil, slowLOIC
Longcat	2.3	YES	LC	TCP, UDP, HTTP
SimpleDoSTool			SD	TCP
Slowloris	0.7	Script	SL	HTTP
Syn Flood DOS			SF	SYN
TORSHAMMER	1.0b	Script	TH	HTTP
UnknownDoser	1.1.0.2		UD	HTTP GET, HTTP POST
XOIC	1.3	YES	XO	Normal (=TCP), TCP, UDP, ICMP

several attacks simultaneously. This is in contrast with Arbor Networks annual report of 2012 [Net12], which points out the growth of multi-vector attacks. Apparently, in case of standalone DoS tools the heterogeneity of attacks is caused by loose coordination of attackers, who can agree on a target, but choice of attack tools is left upon discretion and preferences of each participant.

Only Syn Flood DOS and UnknownDoser require administrative privileges on the host to execute their attacks. However, several other tools require registering OCX files (usually MSWINSCK.OCX) that are not present in a standard Windows 7 SP1 installation. This is a one-time operation that requires elevated privileges. For any subsequent execution of the tool standard user privileges are sufficient.

1.3.3 Binaries

Table 4 lists tool binaries that were analyzed. Version numbers for AnonymousDOS, BanglaDOS, HTTPFlooder, Janidos, SimpleDoSTool and SYN Flood DOS were either not availabl or their author does not differentiate version numbers. No single author nickname was found for HOIC and Longcat tools, although authors of HOIC are believed to be members of group Anonymous. MD5 hash of primary executable is provided. For HTTP DoS Tool and R-U-Dead-Yet an MD5 hash of ZIP archive with all required files is provided.

2 Configurations

Each tool has been tested with various configurations. The first configuration of each tool has been set with default tool settings if such exist. An unique identifier is provided to each configuration for easy reference. Configurations were chosen in order to test primarily settings that can alter the form of produced network traffic. Secondary criterion was to test the tool performance in closed environment. If an input string was required and default string was not provided, the string AABBCCDDEE123456 was used. Total number of tested configurations depends primarily on the variety of attack properties that can be changed.

Every configuration has an unique identifier by which it is referenced. Identifier consists of tool ID, attack type (T – TCP-based, G – HTTP GET, P – HTTP POST, R – Recoil, SL – SlowLOIC) and a sequence number.

Table 4: DoS tool binaries

Name	Version	Author	MD5 hash
Anonymous DoSer	2.0	1337 Haxxor	b86e117d120264bf7d165ed578843510
AnonymousDOS		PrOtOn_An0n	3ea5ca4f7a9a06cf91fc184b77853368
BanglaDOS		Samin Yasar	086ea03e34eb3b603ffc1e0c16ba92d7
ByteDOS	3.2	VanX	997d9bb1c8453de00e6c806fca09b54e
DoS	5.5	xyr0x	e85553ed14ad99b876b0d4ff19e09e7c
FireFlood	1.2	BackStar	45706cd746212ae91fb32a47904033ee
Goodbye	3.0	Puridee	eea3c1840dadeeb2b53b5fe1091c9314
Goodbye	5.2	Puridee	f10bc6dd9b82bf380fedb0c121f1465e
HOIC	2.1.003		451c94a23536dcbba422d7612b34b6ff
HULK	1.0	Barry Shteiman	9851ec582aee27ddfdc966fc4ce9ffd9
HTTP DoS Tool	3.6	Tom Brenann	c8fc6281162085fc4f70dea0141fca68
HTTPFlooder		van1lle	5dbd0ee777d1c96194192c9bd3dd5fa3
Janidos -Weak ed.-		Janizary	ef83f0fe8a4125b4f03737df01bb5e71
JavaLOIC	0.0.3.7	acruxaldebaran	b54c6f08236d8f16d6e80ea5b2661714
LOIC	1.0.4.0	Praetox	9dbe2c1a0f3360af6a9e24b2b303113d
LOIC	1.0.7.42	Praetox	b596e7cacbad1e814b0cd053086c4900
LOIC	1.1.1.25	NewEraCracker	54a4ccfecce789344ff858a85839c531
LOIC	1.1.2.0b	NewEraCracker	976104ade0e9e67a275ae4a5ea58ece9
Longcat	2.3		f5985b20190f56f882f7ed58bd1f92a4
R-U-Dead-Yet	2.2	Hybrid Security	306389a496410e5be9ecfeed8f30decb
SimpleDoSTool		evileXe	83a2e2daeac52f6e2f93e463190ead45
Slowloris	0.7	Rsnake, J. Kinsella	ba43c68709a67e8e233575641e3c7d17
Syn Flood DOS		Defc0n1	da71708b7b9ab59308a89c73ce99c1a9
TORSHAMMER	1.0b	solarstone	6acdb872b766f089eac3ada04043c444
UnknownDoser	1.1.0.2	unKn0wn_H4CK3r	b6ae4be140ffe447889b43050522dd11
XOIC	1.3	DLR	b6c4e2c4fa384212126d7dbb832460c9

Table 5: TCP Attack configurations Part 1

Tool	**Version**	**ID**	**Config**
ByteDOS	3.2	[BD-T-1]	attack: SYN flood, IP: 192.168.1.20, port: 80, Attack: 2x (same attacker)
ByteDOS	3.2	[BD-T-2]	attack: SYN flood, IP: 192.168.1.20, port: 80, Attack: 2x (each attacker one button)
DoS	5.5	[DS-T-1]	target: 192.168.1.20, port: 80, connections: 5000
DoS	5.5	[DS-T-2]	target: 192.168.1.20, port: 80, connections: 10000
JavaLOIC	0.0.3.7	[JL-T-1]	target: 192.168.1.20, port: 80, method: TCP, TCP/UDP message: AABBCCDDEE123456, threads: 10, wait for reply: True, timeout: 9000, delay: 0
JavaLOIC	0.0.3.7	[JL-T-2]	target: 192.168.1.20, port: 80, method: TCP, TCP/UDP message: random, threads: 2, wait for reply: True, timeout: 9000, delay: 0
JavaLOIC	0.0.3.7	[JL-T-3]	target: 192.168.1.20, port: 80, method: TCP, TCP/UDP message: AABBCCDDEE123456, threads: 2, wait for reply: False, timeout: 3000, delay: 0
LOIC	1.0.4.0	[LO1-T-1]	target: 192.168.1.20, speed: minimal, port: 80, threads: 10, timeout: 9001, TCP/UDP message: AABBCCDDEE123456, wait for replay: true
LOIC	1.0.4.0	[LO1-T-2]	target: 192.168.1.20, speed: middle, port: 80, threads: 10, timeout: 9001, TCP/UDP message: AABBCCDDEE123456, wait for replay: true
LOIC	1.0.4.0	[LO1-T-3]	target: 192.168.1.20, speed: middle, port: 80, threads: 5, timeout: 9001, TCP/UDP message: AABBCCDDEE123456, wait for replay: true

Table 6: TCP Attack configurations Part 2

Tool	Version	ID	Config
LOIC	1.0.4.0	[LO1-T-4]	target: 192.168.1.20, speed: middle, port: 80, threads: 5, timeout: 9001, TCP/UDP message: AABBCCDDEE123456, wait for replay: false
LOIC	1.0.7.42	[LO2-T-1]	target: 192.168.1.20, speed: minimal, port: 80, threads: 10, timeout: 9001, TCP/UDP message: AABBCCDDEE123456, wait for replay: true
LOIC	1.0.7.42	[LO2-T-2]	target: 192.168.1.20, speed: middle, port: 80, threads: 10, timeout: 9001, TCP/UDP message: AABBCCDDEE123456, wait for replay: true
LOIC	1.0.7.42	[LO2-T-3]	target: 192.168.1.20, speed: middle, port: 80, threads: 5, timeout: 9001, TCP/UDP message: AABBCCDDEE123456, wait for replay: true
LOIC	1.0.7.42	[LO2-T-4]	target: 192.168.1.20, speed: middle, port: 80, threads: 5, timeout: 9001, TCP/UDP message: AABBCCDDEE123456, wait for replay: false
LOIC	1.1.1.25	[LO3-T-1]	target: 192.168.1.20, speed: minimal, port: 80, method:TCP, threads: 10, timeout: 9001, TCP/UDP message: AABBCCDDEE123456, wait for replay: true
LOIC	1.1.1.25	[LO3-T-2]	target: 192.168.1.20, speed: middle, port: 80, method:TCP, threads: 10, timeout: 9001, TCP/UDP message: U dun goofed, wait for replay: true
LOIC	1.1.1.25	[LO3-T-3]	target: 192.168.1.20, speed: middle, port: 80, method:TCP, threads: 5, timeout: 9001, TCP/UDP message: AABBCCDDEE123456, wait for replay: true
LOIC	1.1.1.25	[LO3-T-4]	target: 192.168.1.20, speed: middle, port: 80, method:TCP, threads: 5, timeout: 9001, TCP/UDP message: AABBCCDDEE123456, wait for replay: false
LOIC	1.1.2.0b	[LO4-T-1]	target: 192.168.1.20, speed: minimal, port: 80, method:TCP, threads: 10, timeout: 30, TCP/UDP message: AABBCCDDEE123456, wait for replay: true

Table 7: TCP Attack configurations Part 3

Tool	**Version**	**ID**	**Config**
LOIC	1.1.2.0b	[LO4-T-2]	target: 192.168.1.20, speed: middle, port: 80, method:TCP, threads: 10, 30, TCP/UDP message: AABBCCDDEE123456, wait for replay: false
LOIC	1.1.2.0b	[LO4-T-3]	target: 192.168.1.20, speed: middle, port: 80, method:TCP, threads: 5, timeout: 30, TCP/UDP message: AABBCCDDEE123456, wait for replay: true
Longcat	2.3	[LC-T-1]	target: 192.168.1.20, port: 80, protocol: SYN Flood, Speed: 2nd from left, use socket blocking: true, number of TCP/UDP users to simulate: 10
Longcat	2.3	[LC-T-2]	target: 192.168.1.20, port: 80, protocol: SYN Flood, Speed: 2nd from left, use socket blocking: true, number of TCP/UDP users to simulate: 2
SimpleDoSTool		[SD-T-1]	IP: 192.168.1.20, port: 80, socks: 300
SimpleDoSTool		[SD-T-2]	IP: 192.168.1.20, port: 80, socks: 100
SimpleDoSTool		[SD-T-3]	IP: 192.168.1.20, port: 80, socks: 500
SYN Flood DoS		[SF-T-1]	target: 192.168.1.20, port: 80
SYN Flood DoS		[SF-T-2]	target: 192.168.1.20, port: 80
XOIC	1.3	[XO-T-1]	IP: 192.168.1.20, port: 80, protocol: Normal, msg: ~MESSAGE~
XOIC	1.3	[XO-T-2]	IP: 192.168.1.20, port: 80, protocol: Normal, msg: AABBCCDDEE123456
XOIC	1.3	[XO-T-3]	IP: 192.168.1.20, port: 80, protocol: TCP, msg: ~MESSAGE~
XOIC	1.3	[XO-T-4]	IP: 192.168.1.20, port: 80, protocol: TCP, msg: AABBCCDDEE123456

Table 8: HTTP Attack configurations Part 1

Tool	**Version**	**ID**	**Config**
AnonymousDoS		[AD-G-1]	threads: 10 (default), message: <empty>
AnonymousDoS		[AD-G-2]	threads: 25, message: AABBCCDDEE123456
AnonymousDoSer	2.0	[ADR-P-1]	Website: 192.168.1.20, time: 120 s
BanglaDOS		[BAD-G-1]	target: http://192.168.1.20/index.htm, requests per second: 10, append message: AABBCCDDEE123456
BanglaDOS		[BAD-G-2]	target: http://192.168.1.20/index.htm, requests per second: 1000 (default), append message: "We Are Legion - Fear US"
FireFlood	1.2	[FF-G-1]	hostname: http://192.168.1.20/index.htm, power: (default)
FireFlood	1.2	[FF-G-2]	hostname: http://192.168.1.20/index.htm, power: full left
FireFlood	1.2	[FF-G-3]	hostname: http://192.168.1.20/index.htm, power: full right
GoodBye	3.0	[GB3-G-1]	IP: 192.168.1.20, uri: /index.htm
GoodBye	5.2	[GB5-G-1]	IP: 192.168.1.20, uri: /index.htm
GoodBye	5.2	[GB5-G-2]	5 simultaneous: IP: 192.168.1.20, uri: /index.htm
HOIC	2.1.003	[HO-G-1]	target: http://192.168.1.20/index.htm, threads: 2 (default), power: low (default), no booster
HOIC	2.1.003	[HO-G-2]	target: http://192.168.1.20/index.htm, threads: 4, power: low (default), no booster
HOIC	2.1.003	[HO-G-3]	target: http://192.168.1.20/index.htm, threads: 2, power: middle, no booster

Table 9: HTTP Attack configurations Part 2

Tool	**Version**	**ID**	**Config**
HTTP DoS Tool (OWASP)	3.6	[HDT-P-1]	attack type: Slow headers, URL: http://192.168.1.20/index.htm, proxy: <empty>, connections: 400, connection rate: 50, timeout: 40 s, random: False, User agent: Mozilla/4.0..., diagnostics: False, Use POST (instead of GET): False
HTTP DoS Tool (OWASP)	3.6	[HDT-P-2]	attack type: Slow POST, URL: http://192.168.1.20/index.htm, proxy: <empty>, connections: 400, connection rate: 50, timeout: 100 s, random: False, User agent: Mozilla/4.0..., diagnostics: False, Content length: 1000000, Random: False, POST field: <empty>, Randomise payload: False
HTTP DoS Tool (OWASP)	3.6	[HDT-P-3]	attack type: Slow headers, URL: http://192.168.1.20/index.htm, proxy: <empty>, connections: 400, connection rate: 50, timeout: 60 s, random: True, User agent: OWASP DDoS, diagnostics: False, Use POST (instead of GET): True
HTTP DoS Tool (OWASP)	3.6	[HDT-P-4]	attack type: Slow POST, URL: http://192.168.1.20/index.htm, proxy: <empty>, connections: 400, connection rate: 50, timeout: 100 s, random: True, User agent: Mozilla/4.0..., diagnostics: False, Content length: 1000000, Random: True, POST field: <empty>, Randomise payload: True
HTTPFlooder		[HF-P-1]	target: 192.168.1.20, duration: 120 s
HULK	1.0	[HU-G-1]	http://192.168.1.20/index.htm

Table 10: HTTP Attack configurations Part 3

Tool	**Version**	**ID**	**Config**
Janidos		[JA-G-1]	target: 192.168.1.20, bot: 50 (default), timeout: 5 (default), get confirm to all requests: FALSE (default), Request type: Firefox (default), Browser: Lynx/2.8.5dev.7 (default), Connection: Keep-alive, Object: image/gif, image/x-xbitmap (default), request page: default, language: tr (default), Referance: <empty> (default)
Janidos		[JA-G-2]	target: 192.168.1.20, bot: 1, timeout: 5 (default), get confirm to all requests: FALSE (default), Request type: Internet Explorer, Browser: Mozilla/4.0, Connection: Keep-alive, Object: image/gif, image/x-xbitmap (default), request page: index.htm, language: en, Referance: <empty> (default)
Janidos		[JA-G-3]	target: 192.168.1.20, bot: 50 (default), timeout: 5 (default), get confirm to all requests: TRUE, Request type: Firefox (default), Browser: Mozilla/4.0, Connection: Keep-alive, Object: image/gif, image/x-xbitmap (default), request page: index.htm, language: tr (default), Referance: <empty> (default)
JavaLOIC	0.0.3.7	[JL-G-1]	target: 192.168.1.20, port: 80, method: HTTP, HTTP subsite: /index.htm, threads: 10 (default), wait for reply: True (default), timeout: 9000 (default), delay: 0 (default)
JavaLOIC	0.0.3.7	[JL-G-2]	target: 192.168.1.20, port: 80, method: HTTP, HTTP subsite: random, threads: 2, wait for reply: True (default), timeout: 9000 (default), delay: 0 (default)
JavaLOIC	0.0.3.7	[JL-G-3]	target: 192.168.1.20, port: 80, method: HTTP, HTTP subsite: /index.htm, threads: 2, wait for reply: False, timeout: 3000, delay: 0 (default)

Table 11: HTTP Attack configurations Part 4

Tool	Version	ID	Config
LOIC	1.0.4.0	[LO1-G-1]	target: 192.168.1.20, subsite: /index.htm, threads: 10 (default), speed: minimal, wait for reply: true (default), port: 80, attack: HTTP, TCP/UDP message: AABBCCDDEE123456, timeout: 9001
LOIC	1.0.4.0	[LO1-G-2]	target: 192.168.1.20, subsite: /index.htm, threads: 1, speed: maximum, wait for reply: true (default), port: 80, attack: HTTP, TCP/UDP message: QWERTYUIOP, timeout: 9001
LOIC	1.0.4.0	[LO1-G-3]	target: 192.168.1.20, subsite: /index.htm, threads: 5, speed: middle, wait for reply: false, port: 80, attack: HTTP, TCP/UDP message: AABBCCDDEE123456, timeout: 4001
LOIC	1.0.7.42	[LO2-G-1]	target: 192.168.1.20, subsite: /index.htm, threads: 10 (default), speed: minimal, wait for reply: true (default), port: 80, attack: HTTP, TCP/UDP message: AABBCCDDEE123456, timeout: 9001
LOIC	1.0.7.42	[LO2-G-2]	target: 192.168.1.20, subsite: /index.htm, threads: 1, speed: maximum, wait for reply: true (default), port: 80, attack: HTTP, TCP/UDP message: QWERTYUIOP, timeout: 9001
LOIC	1.0.7.42	[LO2-G-3]	target: 192.168.1.20, subsite: /index.htm, threads: 5, speed: middle, wait for reply: false, port: 80, attack: HTTP, TCP/UDP message: AABBCCDDEE123456, timeout: 4001
LOIC	1.0.7.42	[LO2-G-4]	target: 192.168.1.20, subsite: /index.htm, threads: 1, speed: minimal, wait for reply: true (default), port: 80, attack: HTTP, TCP/UDP message: AABBCCDDEE123456, timeout: 9001

Table 12: HTTP Attack configurations Part 5

Tool	Version	ID	Config
LOIC	1.1.1.25	[LO3-G-1]	target: 192.168.1.20, speed: minimal, port: 80, method:HTTP, threads: 10 (default), timeout: 9001 (default), wait for replay: true (default), use GZIP: true (default), HTTP subsite: /index.htm, append random characters: false
LOIC	1.1.1.25	[LO3-G-2]	target: 192.168.1.20, speed: minimal, port: 80, method:HTTP, threads: 2, timeout: 9001 (default), wait for replay: true (default), use GZIP: true (default), HTTP subsite: /index.htm?, append random characters: true
LOIC	1.1.1.25	[LO3-G-3]	target: 192.168.1.20, speed: minimal, port: 80, method:HTTP, threads: 1, timeout: 9001 (default), wait for replay: true (default), use GZIP: false, HTTP subsite: /index.htm?, append random characters: true
LOIC	1.1.2.0b	[LO4-G-1]	target: 192.168.1.20, speed: minimal, port: 80, method:HTTP, threads: 10 (default), timeout: 30, wait for replay: true, HTTP subsite: /index.htm, append random characters: false
LOIC	1.1.2.0b	[LO4-G-2]	target: 192.168.1.20, speed: minimal, port: 80, method:HTTP, threads: 2, timeout: 30, wait for replay: false, HTTP subsite: /index.htm?, append random characters: true
LOIC	1.1.2.0b	[LO4-G-3]	target: 192.168.1.20, speed: maximal, port: 80, method:HTTP, threads: 1, timeout: 30, wait for replay: true, HTTP subsite: /index.htm, append random characters: true

Table 13: HTTP Attack configurations Part 6

Tool	Version	ID	Config
LOIC	1.1.2.0b	[LO4-R-1]	target: 192.168.1.20, speed: minimum, port: 80, method:Recoil, threads: 10, socket/thread: 50, http subsite: /index.htm, timeout: 30
LOIC	1.1.2.0b	[LO4-R-2]	target: 192.168.1.20, speed: middle, port: 80, method:Recoil, threads: 5, socket/thread: 15, http subsite: /index.htm, timeout: 30
LOIC	1.1.2.0b	[LO4-SL-1]	target: 192.168.1.20, speed: minimum, port: 80, method:Recoil, threads: 10, socket/thread: 50, http subsite: /index.htm, timeout: 30, use GET: False
LOIC	1.1.2.0b	[LO4-SL-2]	target: 192.168.1.20, speed: middle, port: 80, method:Recoil, threads: 5, socket/thread: 15, http subsite: /index.htm, timeout: 30, use GET: False
LOIC	1.1.2.0b	[LO4-SL-3]	target: 192.168.1.20, speed: minimum, port: 80, method:Recoil, threads: 10, socket/thread: 50, http subsite: /index.htm, timeout: 30, use GET: True
LOIC	1.1.2.0b	[LO4-SL-4]	target: 192.168.1.20, speed: middle, port: 80, method:Recoil, threads: 5, socket/thread: 15, http subsite: /index.htm, timeout: 30, use GET: True
Longcat	2.3	[LC-G-1]	target: http://192.168.1.20/index.htm, number of HTTP users to simulate: 25 (default)
Longcat	2.3	[LC-G-2]	target: http://192.168.1.20/index.htm, number of HTTP users to simulate: 5
Longcat	2.3	[LC-G-3]	target: http://192.168.1.20/index.htm, number of HTTP users to simulate: 1
Slowloris	0.7	[SL-G-1]	(HTTP DoS example) -dns 192.168.1.20 -port 80 -timeout 2000 -num 500 -tcpto 5
Slowloris	0.7	[SL-G-2]	(stealth host DoS example) -dns 192.168.1.20 -port 80 -timeout 30 -num 500 -tcpto 1 -shost www.virtualhost.com
Slowloris	0.7	[SL-G-3]	-dns 192.168.1.20 -port 80 -timeout 50 -num 100 -tcpto 5

Table 14: HTTP Attack configurations Part 7

Tool	**Version**	**ID**	**Config**
Torshammer	1.0b	[TH-P-1]	-t 192.168.1.20 -p 80 -r 256
Torshammer	1.0b	[TH-P-2]	-t 192.168.1.20 -p 80 -r 64
UnknownDoser	1.1.0.2	[UD-G-1]	target: 192.168.1.20, port: 80, method: GET, randomize request: NO (default), timeout: 5 s (default), duration: 300 s, number of threads: 5
UnknownDoser	1.1.0.2	[UD-G-2]	target: 192.168.1.20, port: 80, method: GET, randomize request: YES, timeout: 5 s (default), duration: 300 s, number of threads: 2
UnknownDoser	1.1.0.2	[UD-P-1]	target: 192.168.1.20, port: 80, method: POST, randomize request: NO (default), timeout: 20 s, duration: 300 s, number of threads: 5
UnknownDoser	1.1.0.2	[UD-P-2]	target: 192.168.1.20, port: 80, method: POST, randomize request: YES, timeout: 10 s, duration: 300 s, number of threads: 5
UnknownDoser	1.1.0.2	[UD-G-3]	target: 192.168.1.20, port: 80, method: GET, randomize request: NO, timeout: 5 s (default), duration: 120 s (default), number of threads: 2

3 IP behavior

3.1 Traffic volume

Byte rate and packet rate are the most common characteristics associated with DoS attacks. Traditionally, DoS attacks were believed to produce excessively high volume of attack traffic in order to overwhelm the target. However, even though the peak volumes of observed DoS attacks are steadily increasing, the ratio between the number of volume-based attacks and the number of vulnerability-based attacks is slowly shifting towards low-rate attacks.

Traffic volume is simply measured with NetFlow data, which assures high scalability of attack detection. Intuitively, traffic volume metric is the most efficient when deployed in the target network, where the attack traffic from all sources is aggregated. We intend to provide a notion of what traffic volume can be observed in the source end network.

3.1.1 Packet rate

Packet rate is arguably the most frequent input feature of DoS detection. Gil and Poletto construct a tree of packet rate statistics in order to track significant disproportional differences between packet rates going to and coming from a host or subnet [GP01]. Dainotti et al. process a time series of packet rates [DPV06]. Lee et al. calculate packet volume in conjunction with other parameters as input for cluster analysis of traffic [LKK+08].

3.1.2 Byte rate

Byte rate is frequently used as an input feature for DoS attack detection. It has been utilized in many aspects. Öke and Loukas employ both total byte rate and byte rate change to detect high volume attacks [LO09]. Shevtekar and Ansari measure flow bytes and total bytes for low-rate attacks detection [SA07]. Tao et al. measure separately traffic volume from the attacker towards the victim and from the victim back to the attacker [TYP+09].

3.1.3 Attack buildup

Attack can either start with full possible strength or a short period of gradual attack buildup may take place. Immediate full strength attack benefit from a higher attack

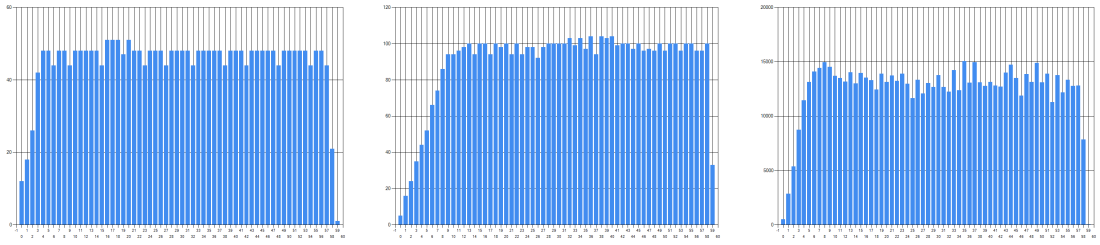
Table 15: Average packet rate

0 – 100 pps	[BD-T-1] [BD-T-2] [HDT-P-1] [HDT-P-2] [HDT-P-3] [HDT-P-4] [JA-G-2] [LO1-T-1] [LO1-T-2] [LO1-T-3] [LO1-T-4] [LO2-T-1] [LO2-T-2] [LO2-T-3] [LO2-T-4] [LO4-R-2] [LO4-SL-1] [LO4-SL-2] [LO4-SL-3] [LO4-SL-4] [SL-G-1] [SL-G-2] [SL-G-3] [SF-T-1] [SF-T-2] [TH-P-2]
101 – 500 pps	[AD-G-1] [AD-G-2] [ADR-P-1] [BAD-G-1] [DS-T-1] [DS-T-2] [GB3-G-1] [GB5-G-1] [GB5-G-2] [HF-P-1] [LO3-T-1] [LO3-T-3] [LO3-T-4] [LO4-G-3] [LO4-T-1] [LO4-T-2] [LO4-T-3] [LO4-R-1] [LC-T-2] [TH-P-1] [UD-G-2] [UD-P-1] [UD-P-2]
501 – 1000 pps	[HO-G-1] [LO3-T-2] [LC-G-1]
1001 – 5000 pps	[BAD-G-2] [HO-G-2] [HO-G-3] [HU-G-1] [JA-G-1] [JA-G-3] [JL-T-2] [LO1-G-1] [LO2-G-4] [LO3-G-2] [LO3-G-3] [LO4-G-2] [LC-T-1] [UD-G-3]
5001 – 10000 pps	[JL-G-1] [JL-G-2] [JL-G-3] [JL-T-1] [JL-T-3] [LO1-G-2] [LO1-G-3] [LO3-G-1] [LC-G-3] [SD-T-1] [SD-T-2] [SD-T-3] [UD-G-1]
10000+ pps	[FF-G-1] [FF-G-2] [FF-G-3] [LO2-G-1] [LO2-G-2] [LO2-G-3] [LO4-G-1] [LC-G-2] [XO-T-1] [XO-T-2] [XO-T-3] [XO-T-4]

Table 16: Average byte rate

0 – 100 kbit/s	[LO4-SL-2] [LO4-SL-4] [SF-T-1] [SF-T-2]
100 Kbit/s – 1 Mbit/s	[BD-T-1] [BD-T-2] [DS-T-1] [DS-T-2] [HDT-P-1] [HDT-P-3] [JA-G-2] [LO1-T-1] [LO1-T-3] [LO1-T-4] [LO2-T-1] [LO2-T-3] [LO2-T-4] [LO4-R-2] [LO4-SL-1] [LO4-SL-3] [SL-G-1] [SL-G-2] [SL-G-3] [TH-P-2]
1 Mbit/s – 10 Mbit/s	[ADR-P-1] [GB3-G-1] [GB5-G-1] [GB5-G-2] [HDT-P-2] [HDT-P-4] [HF-P-1] [LO1-T-2] [LO2-T-2] [LO3-T-1] [LO3-T-2] [LO3-T-3] [LO3-T-4] [LO4-G-3] [LO4-T-1] [LO4-T-2] [LO4-T-3] [LO4-R-1] [LC-T-1] [LC-T-2] [TH-P-1] [UD-G-2] [UD-P-1] [UD-P-2]
10 Mbit/s – 100 Mbit/s	[AD-G-1] [AD-G-2] [BAD-G-1] [HO-G-1] [JA-G-1] [JA-G-3] [JL-G-1] [JL-G-3] [JL-T-1] [JL-T-3] [LO1-G-1] [LO1-G-2] [LO1-G-3] [LC-G-1] [SD-T-1] [SD-T-2] [SD-T-3]
100 Mbit/s – 1 Gbit/s	[BAD-G-2] [FF-G-1] [FF-G-2] [FF-G-3] [HO-G-2] [HO-G-3] [HU-G-1] [JL-G-2] [JL-T-2] [LO2-G-4] [LO3-G-1] [LO3-G-2] [LO3-G-3] [LO4-G-1] [LO4-G-2] [LC-G-3] [UD-G-1] [UD-G-3] [XO-T-1] [XO-T-2] [XO-T-3] [XO-T-4]
1 Gbit/s+	[LO2-G-1] [LO2-G-2] [LO2-G-3] [LC-G-2]

Figure 1: Attack buildup



(a) [LO1-T-3] Packet rate (b) [LO2-T-1] Packet rate (c) [LO4-G-1] Packet rate

intensity and flexibility. Quick response makes this approach ideal for attacks on online gamers or ransom victims. On the other hand, gradual buildup is more subtle. It is more difficult to recognize for intrusion detection systems that are based on the change detection.

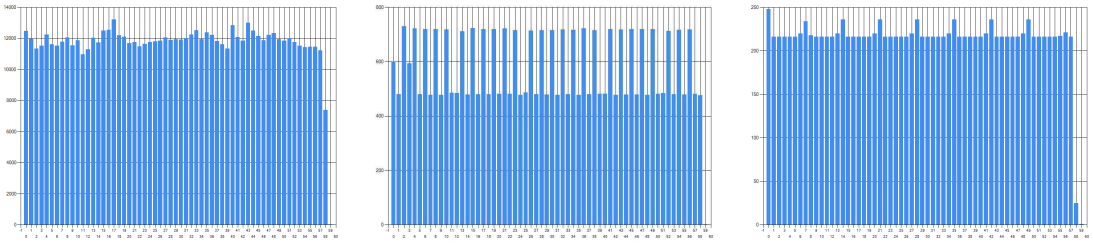
In our set, the clear majority of tools employs immediate full attack strength approach. Exceptions are LO and JL configurations which may have an initiation period up to 10 seconds long (Figure 1). We consider this revelation important, because it is a strong indicator that detection methods based on change detection can be widely adopted in real environments.

3.1.4 Byte rate and packet rate relationship

Division of configurations into classes by byte rate shows that we can encounter both volume-rich tools and tools that produce hardly any traffic. When choosing configurations for the analysis, we focused on low performance settings in order to record every packet transmitted. Therefore, byte rate and packet rate values are especially interesting for tools that do not enable to specify attack intensity (e.g., ADR, GB, HF, XO). Tables 15 and 16 show these tools are spread across the whole spectrum of possible values. Oppositely, tools that enable performance settings can trivially circumvent simple volume thresholds.

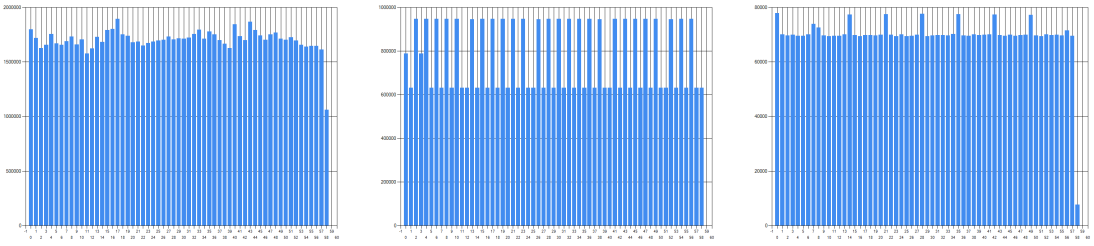
For the vast majority of configurations the changes of byte rate value in time correspond to the changes of packet rate value. Examples for comparison are offered in Figures 2 and 3. The reason is a very homogenous distribution of attack traffic packet sizes as will be seen in Section 3.6.

Figure 2: Packet rate



(a) [FF-G-1] (b) [HO-G-1] (c) [UD-G-2]

Figure 3: Byte rate



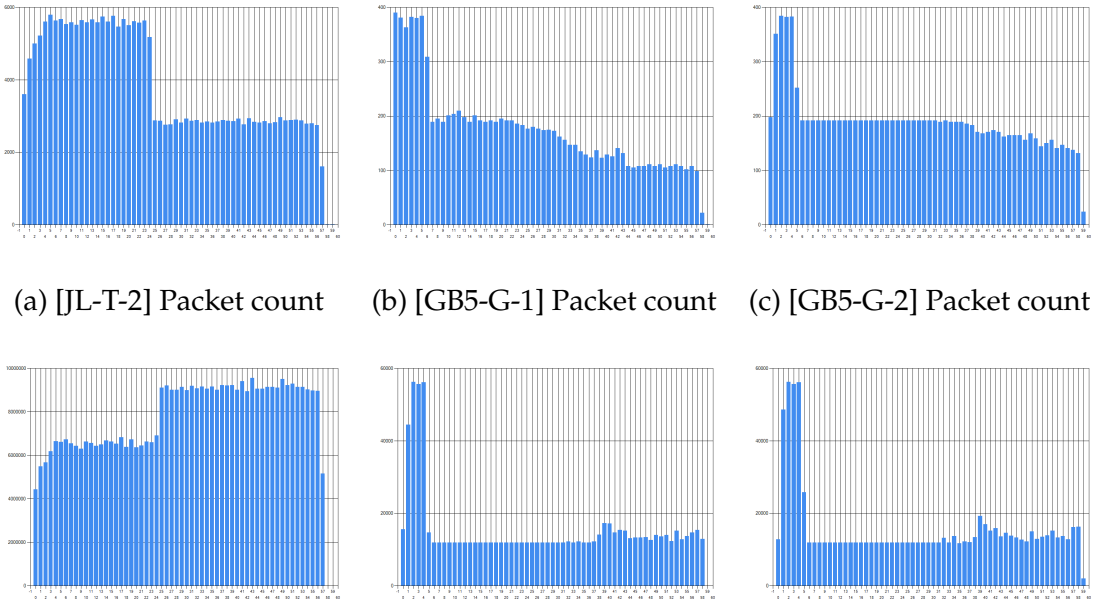
(a) [FF-G-1] (b) [HO-G-1] (c) [UD-G-2]

Only exceptions are configurations [GB5-G-1], [GB5-G-2] and [JL-T-2]. In case of GB, while the packet rate starts to slowly decline after approximately 35 seconds, the byte rate rises slightly and becomes much more volatile. This effect is caused by a change in attack method on-the-fly. Most attack flows of GB5 just establish a TCP connection and leave it open. However, during the first 5 seconds of the attack and again since 35 second mark some flows also send a malformed HTTP request towards the victim, hence the increase in byterate. [JL-T-2] is a configuration that produces fully random payloads of packets in a TCP flooding attack. After 25 seconds the packet rate drops considerably while byte rate significantly increases. The number of distinct packet sizes in [JL-T-2] is 35. The packet drop is caused by several higher-than average packet sizes, that are used more frequently after 25 second mark.

3.2 Packet rate burst behavior

Packet rates of many DoS tools in our set exhibit burst behavior. The cause may be ticking of internal tool timer, synthesis of requests between multiple process threads or an intentional design characteristic. We divide observed burst types into four types.

Figure 4: Packet rate and byte rate relationship anomalies



(a) [JL-T-2] Packet count (b) [GB5-G-1] Packet count (c) [GB5-G-2] Packet count

(d) [JL-T-2] Byte rate (e) [GB5-G-1] Byte rate (f) [GB5-G-2] Byte rate

Attribution of configurations to burst types is shown in Table 17. We did not observe any significant differences between the burst behavior of TCP DoS attack tools and HTTP DoS attack tools.

- Full burstiness. The attack traffic is delivered only in bursts. Minimal or none traffic is exchanged between bursts. Full burstiness is very popular with slow attacks, often probably due to guidance by an internal clock. However, tools that produce low rate traffic are also most likely to be modified in order not to produce traffic in bursts.
- Regular peaks. Produced network traffic is very stable except for regular repeating anomalies.
- One-time extreme. At some point of the tool run, often at the beginning of the attack, the traffic characteristics are significantly different from the rest.
- None. The tool does not produce traffic that has observable bursts in packet rate.

Athough to our knowledge DoS burst behavior has not yet been used in the source end DoS attack detection, it could become a valid alternative to existing detection methods. A new method could be based on the detection of burst behavior, recognition of

repeated occurrences of bursts and on similarity comparisons of these bursts. Advantages are low input data collection costs and a possibility of continuous detection which does not require comparison to a model of normal network traffic.

Table 17: Packet rate burstiness

Full burstiness	[HDT-P-1] [HDT-P-2] [HU-G-1] [LO4-SL-1] [LO4-SL-2] [LO4-SL-3] [LO4-SL-4] [SL-G-1] [SL-G-2] [SL-G-3] [SF-T-1] [SF-T-2]
Regular peaks	[BD-T-1] [BD-T-2] [HO-G-1] [HO-G-2] [HO-G-3] [LO1-G-1] [LO3-G-2] [LO3-G-3] [LO4-G-3] [LC-T-1] [LC-T-2] [UD-G-2] [UD-P-1] [UD-P-2]
One-time extreme	[AD-G-1] [AD-G-2] [BAD-G-1] [BAD-G-2] [DS-T-1] [DS-T-2] [GB3-G-1] [GB5-G-1] [GB5-G-2] [HDT-P-3] [HDT-P-4] [TH-P-1] [TH-P-2]
None	[ADR-P-1] [FF-G-1] [FF-G-2] [FF-G-3] [HF-P-1] [JA-G-1] [JA-G-2] [JA-G-3] [JL-G-1] [JL-G-2] [JL-G-3] [JL-T-1] [JL-T-2] [JL-T-3] [LO1-G-2] [LO1-G-3] [LO1-T-1] [LO1-T-2] [LO1-T-3] [LO1-T-4] [LO2-G-1] [LO2-G-2] [LO2-G-3] [LO2-G-4] [LO2-T-1] [LO2-T-2] [LO2-T-3] [LO2-T-4] [LO3-G-1] [LO3-T-1] [LO3-T-2] [LO3-T-3] [LO3-T-4] [LO4-G-1] [LO4-G-2] [LO4-T-1] [LO4-T-2] [LO4-T-3] [LO4-R-1] [LO4-R-2] [LC-G-1] [LC-G-2] [LC-G-3] [SD-T-1] [SD-T-2] [SD-T-3] [UD-G-1] [UD-G-3] [XO-T-1] [XO-T-2] [XO-T-3] [XO-T-4]

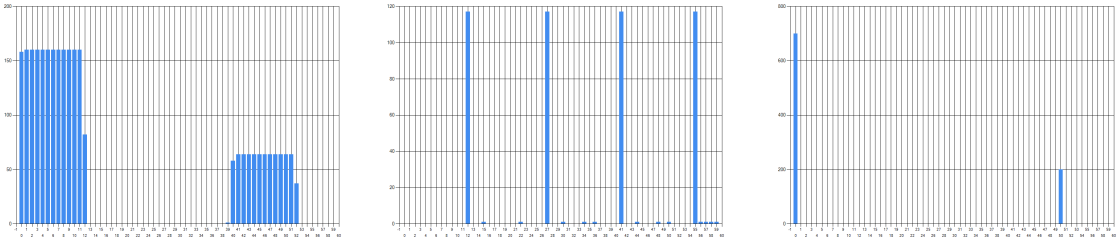
3.3 IP fragmentation

Packet fragmentation can be employed to boost the efficiency of any DoS attack. By dividing attack packets into several fragments, a victim is forced to commit some of its computational resources to packet reassembly. However, packet fragmentation is becoming less and less common in current networks. Therefore, fragmented traffic can be efficiently filtered out by ISPs or at the border router of victim's network. None of the standalone DoS tools in our set employed fragmentation.

3.4 IP spoofing

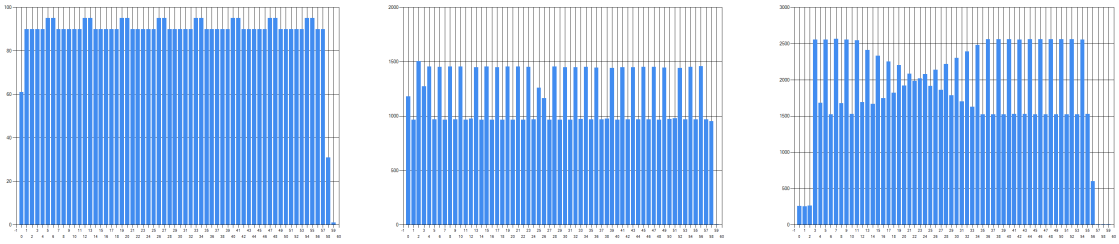
Forging packet source IP address is traditionally associated with SYN flood attacks, reflection attacks and amplification attacks [Pax01]. By using IP spoofing, the volume

Figure 5: Packet rate burstiness – Full burstiness example



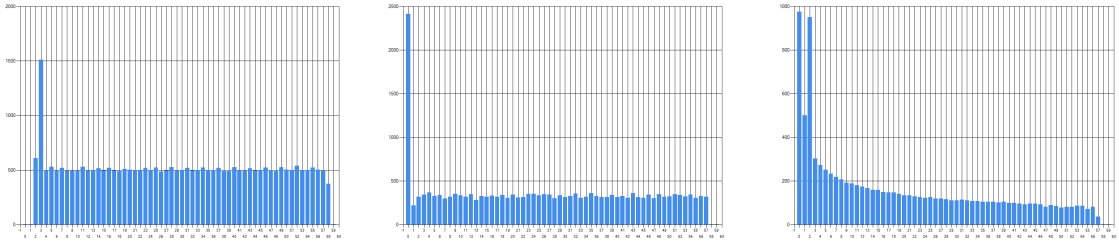
(a) [HDT-P-1] (b) [SF-T-2] (c) [SL-T-3]

Figure 6: Packet rate burstiness – Regular peaks example



(a) [BD-T-1] (b) [HO-G-2] (c) [LC-T-1]

Figure 7: Packet rate burstiness – One-time extreme example



(a) [AD-G-1] (b) [TH-P-1] (c) [DS-T-1]

of attack traffic can be multiplied by poorly configured hosts in the middle. Also, attacker's real IP addresses are hidden from the target and the attack traffic is received from a significantly higher number of sources. On the other hand, spoofed attacks must be very simple, because it is not possible for the attacker to complete a handshake with the target. Numerous methods for source-end DoS detection based on the recognition of IP spoofing were proposed [XCH06, PYHR10] in the past.

Efficient IP spoofing attacks require tight cooperation between participating hosts and an up-to-date list of available proxy servers or a large number of hosts some of which serve as proxies. This level of coordination is difficult to achieve with standalone DoS attack tools. Users of standalone DoS attack tools can be located in different timezones, have different means of communication and use various attack tools. Therefore, the popularity of IP spoofing among standalone DoS attack tools is low, SYN Flood DoS being the only representative in our set.

3.5 Average packet size

Continuous measurement of average connection packet size in time is a simple metrics to distinguish between legitimate and non-standard flows. The detection can be based on observing divergence from a predefined threshold or on observation of long-term metric values. TCP-based DoS attacks are rarely based on the volume of transmitted data, but rather on exploiting the design weaknesses of TCP protocol. Therefore, in many cases no payload data is actually transmitted over established connections or the transmitted messges are short, often expressing political or sociological opinion (e.g., default string at LOIC "U dun goofed"). Since all packets during the TCP 3-way handshake have small constant packet sizes, average packet size measurement can be an effective detection method for these connections.

Seo at al. measure occurrences of connections with average byte size less than 64 bytes [SWH11] as a mean how to detect DoS attacks at attacker's network. Table 18 summarizes our comparative results. We agree that average packet size is less than 64 bytes for most TCP-based DoS attacks, including TCP SYN flood with spoofed source IP addresses. Notable exceptions are LOIC, JavaLOIC and XOIC that employ volumetric flooding. In case of DoS attack tools, average packet size depends mostly on lengths of IP and TCP header, length of payload and also on internal mechanics of each tool. Since modifications of IP and TCP headers are relatively rare, the string to be inserted in payload of TCP segments is crucial. For example, JavaLOIC allows to use a random

string as a payload for each TCP segment and LOIC creates several concatenations of the user-chosen string to create a TCP payload. Long payloads subsequently result in average packet sizes higher than the threshold.

The 64-byte threshold is also useable for some types of HTTP DoS attacks based on malformed HTTP requests (Table 19). Especially slow attacks, that exploit long server-side timeouts for sending or receiving HTTP messages, are likely candidates. Other parameters that can influence the success rate of detection are the length of URL, the size of error message that is sent from server or lengths of optional fields in a malformed HTTP request.

Long-term observation of static average packet size cannot be considered a detection metrics by itself. However, it can serve as an secondary evidence that the flow is homogenous and repeating in time. Also, attack tools that can circumvent host system network stack, may enforce a different behavior from an expected behavior of TCP protocol. When a target server is becoming overwhelmed with messages, it requests clients to lower their rate of sending via a TCP protocol mechanics. Attacker tools with root access can ignore the request and maintain constant byte rate and packet rate, therefore revealing themselves as anomalous.

Table 18: Average packet size with limit – TCP configurations

Average packet size =< 64 B	[BD-T-1] [BD-T-2] [DS-T-1] [DS-T-2] [JL-T-1] [JL-T-3] [LC-T-1] [LC-T-2] [SD-T-1] [SD-T-2] [SD-T-3] [SF-T-1] [SF-T-2]
Average packet size > 64 B	[JL-T-2] [LO1-T-1] [LO1-T-2] [LO1-T-3] [LO1-T-4] [LO2-T-1] [LO2-T-2] [LO2-T-3] [LO2-T-4] [LO3-T-1] [LO3-T-3] [LO3-T-4] [LO4-SL-1] [LO4-SL-2] [LO4-SL-3] [LO4-SL-4] [LO4-T-1] [LO4-T-2] [LO4-T-3] [XO-T-1] [XO-T-2] [XO-T-3] [XO-T-4]

3.6 Packet size distribution

A common TCP connection consists of a 3-way handshake (3WH) part, data transmission part and closure part. TCP segments in the 3WH part and the closure part have constant packet sizes. Among those the most frequent values are one 60/62/66 B packet per flow for connection establishment (SYN segments from the attacker and SYNACK segments from the victim), several 54 B packets (ACK segments) and one FIN/RST segment for connection closure. Connection closure packets often have 54 B also. Some

Table 19: Average packet size with limit – HTTP configurations

Average packet size =< 64 B	[LO4-R-1] [LO4-R-1] [LO4-R-2] [SL-G-1] [SL-G-2] [SL-G-3] [TH-P-1] [TH-P-2]
Average packet size > 64 B	[ADR-P-1] [AD-G-1] [AD-G-2] [BAD-G-1] [BAD-G-2] [FF-G-1] [FF-G-2] [FF-G-3] [GB3-G-1] [GB5-G-1] [GB5-G-2] [HDT-P-1] [HDT-P-2] [HDT-P-3] [HDT-P-4] [HO-G-1] [HO-G-2] [HO-G-3] [HF-P-1] [HU-G-1] [JA-G-1] [JA-G-2] [JA-G-3] [JL-G-1] [JL-G-2] [JL-G-3] [LO1-G-1] [LO1-G-2] [LO1-G-3] [LO2-G-1] [LO2-G-2] [LO2-G-4] [LO3-G-1] [LO3-G-2] [LO3-G-3] [LO4-G-1] [LO4-G-2] [LO4-G-3] [LC-G-1] [LC-G-2] [LC-G-3] [UD-G-1] [UD-G-2] [UD-G-3] [UD-P-1] [UD-P-2]

types of DoS attacks completely omit data transmission part. Therefore, by examining the distribution of packet sizes it may be possible to detect these types of DoS attacks. Du and Abe discovered that many applications have typical packet sizes with respect to requests and responses or data and acknowledgements [DA07]. We argue, that the same holds for many standalone DoS tools and by extension also for types of DoS attacks.

In our measurements, we consider only the outgoing (attack) traffic in order to create a more focused profile of DoS attack tools and to minimize affects of server-side processing. We believe that any potential DoS detection method based on packet size distribution of DoS traffic should focus on flow from the attacker to the victim. This approach minimizes errors caused by changing server state (e.g., dynamic target webpages, server's growing unresponsiveness). Tables 21 and 20 show packet size distributions of all configurations. Packet size is considered major if its ratio in the traffic exceeds 1%. We have introduced this second distribution in order to filter out rare events (e.g., initial probing flows, retransmissions).

From Table 20 it can be seen that all configurations except those of LOIC and JavaLOIC in randomization mode produce flows with at most 3 unique packet sizes, often only 2 of those being dominant. These unique packet sizes correspond to our perception of three parts of a TCP connection. As in case of average packet size, LOIC and JavaLOIC are again outliers from remaining TCP-based DoS attack tools. Oppositely, XOIC's behavior is now comparable with the rest of the tools. It's high average packet

size is caused by a big response from the server, which has no impact in our design of packet size distribution measurement.

As expected, HTTP traffic is more diverse from the point of packet size distribution. Established TCP connection is a prerequisite to execute HTTP DoS attack. Also, at least one another packet is required – HTTP request message. Yet still AD, ADR, BAD, GB, JA and JL still maintain unique packet size count at three. Reason is an improper handling of the connection closure. AD, ADR, BAD and GB leave the connection closure entirely up to the server, therefore saving one of three common unique packet sizes for other use, in this case for an HTTP request. Oppositely, JA and JL do close their connections, but the RST/FIN segments are encapsulated in packets of 54B length, therefore being grouped with common ACK segments.

3.7 Packet incoming to outgoing ratio

Ratio of the number of incoming and outgoing packets has been used as a detection metrics by Laurens et al. [LMSD09] and Suresh and Anitha [SA11]. The intention of the attacker is to overwhelm the target. Authors assume that the attacker will not wait for server acknowledgements and will aggresively flood the victim, making the ratio abnormaly high.

Ratio can be also utilized for traffic pattern matching. Our other measurements have shown that the attack traffic tends to exhibit only little variance. We highlight two significant values for incoming to outgoing packet count ratio that were encountered at multiple configurations during initial 60 seconds of the attack tool run. Even if the ratio does not fall into one of these two classes, the value may still be important because it is often long-term constant (e.g., [XO-T-1] holds 0.6 very stable in time).

- 0.5. This value is characteristic for TCP attacks that are closed one-sidedly by the victim. I.e., victim may send a FIN TCP segment which is ignored by the attacker or the victim may send RST TCP segment, which does not mandate response from the attacker in the first place.

- 0.66. This ratio is common for attacks when both sides participate in the flow closure, after it was established via a 3-way handshake, but without any data packets exchanged. Any more packets transmitted via a TCP connection will increase the ratio in the longterm.

Table 20: Packet size distribution – TCP

Config ID	Unique packet size count	Major packet size count
[BD-T-1]	3	3
[BD-T-2]	3	3
[DS-T-1]	2	2
[DS-T-2]	2	2
[JL-T-1]	2	2
[JL-T-2]	30+	8
[JL-T-3]	2	2
[LO1-T-1]	8	2
[LO1-T-2]	13	3
[LO1-T-3]	7	2
[LO1-T-4]	11	2
[LO2-T-1]	6	2
[LO2-T-2]	6	2
[LO2-T-3]	7	4
[LO2-T-4]	2	2
[LO3-T-1]	3	1
[LO3-T-2]	2	1

Config ID	Unique packet size count	Major packet size count
[LO3-T-3]	3	1
[LO3-T-4]	3	1
[LO4-T-1]	3	1
[LO4-T-2]	3	1
[LO4-T-3]	3	1
[LC-T-1]	3	2
[LC-T-2]	3	2
——		
[SD-T-1]	3	3
[SD-T-2]	3	2
[SD-T-3]	3	3
[SF-T-1]	2	2
[SF-T-2]	2	2
[XO-T-1]	3	3
[XO-T-2]	3	3
[XO-T-3]	3	3
[XO-T-4]	3	3

Table 21: Packet size distribution – HTTP

Config ID	Unique packet size count	Major packet size count
[AD-G-1]	3	2
[AD-G-2]	3	2
[ADR-P-1]	3	3
[BAD-G-1]	3	2
[BAD-G-2]	3	2
[FF-G-1]	4	3
[FF-G-2]	4	3
[FF-G-3]	4	3
[GB3-G-1]	3	3
[GB5-G-1]	3	3
[GB5-G-2]	3	3
[HO-G-1]	3	3
[HO-G-2]	3	3
[HO-G-3]	3	3
[HDT-P-1]	6	6
[HDT-P-2]	3	3
[HDT-P-3]	7	6
[HDT-P-4]	8	7
[HF-P-1]	3	3
[HU-G-1]	100+	2
[JA-G-1]	3	3
[JA-G-2]	3	3
[JA-G-3]	3	3
[JL-G-1]	3	3
[JL-G-2]	3	3
[JL-G-3]	3	3
[LC-G-1]	10+	3
[LC-G-2]	7	4
[LC-G-3]	5	2

Config ID	Unique packet size count	Major packet size count
[LO1-G-1]	3	3
[LO1-G-2]	3	3
[LO1-G-3]	3	3
[LO2-G-1]	6	4
[LO2-G-2]	7	5
[LO2-G-3]		
[LO2-G-4]	6	5
[LO3-G-1]	6	4
[LO3-G-2]	6	5
[LO3-G-3]	4	4
[LO4-G-1]	7	6
[LO4-G-2]	7	4
[LO4-G-3]	4	4
[LO4-R-1]	4	1
[LO4-R-2]	4	1
[LO4-SL-1]	5	5
[LO4-SL-2]	5	5
[LO4-SL-3]	5	5
[LO4-SL-4]	5	5
[SL-G-1]	4	4
[SL-G-2]	4	4
[SL-G-3]	4	4
[TH-P-1]	20+	3
[TH-P-2]	20+	3
[UD-G-1]	6	4
[UD-G-2]	100+	2
[UD-P-1]	3	3
[UD-P-2]	100+	2
[UD-G-3]	6	4

Because the detection based on dwindling victim responses depends heavily on the victim side and also on the success or failure of the attack, it was not included in our analysis.

Table 22: Limited – Interesting values – approximate

0.49-0.51	[DS-T-1] [DS-T-2] [GB3-G-1] [GB5-G-1] [GB5-G-2] [JA-G-1] [LO3-T-1] [LO3-T-2] [LO3-T-3] [LO3-T-4] [LO4-T-1] [LO4-T-2] [LO4-T-3] [UD-G-2] [UD-P-2]
0.65-0.67	[BD-T-1] [BD-T-2] [JL-T-1] [JL-T-3] [SD-T-1] [SD-T-2] [SD-T-3]

4 TCP behavior

4.1 Flow count

One of the most common assumptions about DoS attacks is that attacker's establish many connections towards a victim. Reasoning behind the assumption states that multiple connections imply higher attack performance. Also, some attacks are based on the number of connections or on the rate of their generation and therefore high number of flows is a desirable property. A number of detection methods that calculate with flow count have been proposed, for example by Malliga et al. [MTJ08], Cheng et al.[CYL+09], Lee et al.[LKK+08] and Mirkovic and Reiher [MR05]. High number of flows is frequently associated with the presence of IP spoofing. Since the response is never received, flows with spoofed source IPs are considered open for a prolonged time, emphasizing the presence of an anomaly in the monitored network.

Our observations partially contradict and partially supplement these perceptions. Tables 23 and 24 classifies configurations by the number of flows that were observed during initial 60 seconds of the attack.

- Of our set, 28 configurations generate 100 or less flows during a first 60 seconds of a DoS attack. Without regards to tools' versions, following tools can be configured to launch an attack with 100 or less flows: AD, BAD, HDT, LO (TCP, Recoil, SlowLOIC), LC (HTTP), SL, TH. Low flow counts make these tools undetectable for source-end intrusion detection systems that are based on analysis of flow count.

- At least 2 tools in the set can generate more than 1000 flows per second without IP spoofing on a standard laptop. Depending on the configuration and the performance of the source host, several more tools can be expected to reach such limit, especially if the tool is executed several times in parallel (e.g., HU, FF).

Low flow count attacks can be seen usually with HTTP-based DoS tools, such as AD, BAD and LC (HTTP attack). These tools open a static low number of connections towards the victim and subsequently send HTTP requests repeatedly over existing connections. Slow attacks also tend to have a smaller number of connections. Common principle of slow attacks is to bind available server resources for long-term reservations. Slow attacks are therefore limited to hundreds or thousands connections, usually simultaneous. As expected, TCP-based DoS attacks usually cause a high number of connections, LOIC TCP flooding attack as the only exception. Flow count differences between TCP-based attacks are caused mostly by the efficiency of tool implementation.

From the point of long-term performance, we have recognized four patterns.

- Stability. Most tools exhibit only minor changes while the long term trend remains steady, e.g. FF (Figure 8a) or JA (Figure 8b). In many cases it takes several seconds of a transition period to build up the attack. Figure 8c gives an example of LO gradually opening flows up to the desired limit during the first 5 seconds and subsequently keeping the count unchanged.

- Pulsing. Intentionally pulsing attack is generally viewed as an attempt to stay undetected while maintaining a reasonable per host attack strength. Our measurement shows that pulsing can also be an integral part of the attack. Representatives are LO, which achieves pulsing by batch flow closures (Figure 8d) or HDT, which alternates between calm no-traffic periods and periods of batch packet sendings (Figure 8e).

- Decreasing count. Several tools such as DS, GB and HDT tend to decrease the number of observable flows, even if the victim has not been made unavailable (e.g., Figure 8f or Figure 8g).

- Increasing count. Although attacker is expected to attempt to use all available resources as soon as possible to overwhelm the victim, increasing strength could be used to circumvent reputation-based and some anomaly-based intrusion detection systems. Subtle attack start phase could lead to the attack being undetected

for a prolonged time. Naturally, subtle attacks are not tempting for hacktivists, who want the publicity of the attack. None of the tools in our set has shown an increasing strength trend from the point of flows count.

Table 23: Flow count in 60 s interval – TCP

0-10	[LO1-T-1] [LO1-T-2] [LO1-T-3] [LO1-T-4] [LO2-T-1] [LO2-T-2] [LO2-T-3] [LO2-T-4] [LO3-T-1] [LO3-T-2] [LO3-T-3] [LO3-T-4] [LO4-T-1] [LO4-T-2] [LO4-T-3]
11-100	
101-1000	[SF-T-1] [SF-T-2]
1001-10000	[BD-T-1] [BD-T-2] [DS-T-1] [DS-T-2] [LC-T-2]
10001-100000	[JL-T-1] [JL-T-2] [JL-T-3] [LC-T-1]
100000+	[SD-T-1] [SD-T-2] [SD-T-3] [XO-T-1] [XO-T-2] [XO-T-3] [XO-T-4]

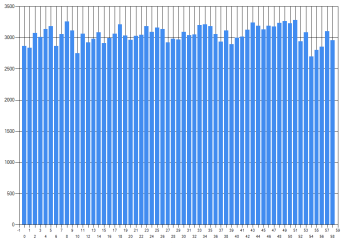
Table 24: Flow count in 60 s interval – HTTP

0-10	[AD-G-1] [AD-G-2] [BAD-G-1] [BAD-G-2] [LC-G-1] [LC-G-2] [LC-G-3]
11-100	[LO4-R-1] [LO4-R-2] [LO4-SL-2] [LO4-SL-4] [SL-G-3] [TH-P-2]
101-1000	[ADR-P-1] [HO-G-1] [HO-G-2] [HO-G-3] [HDT-P-1] [HDT-P-2] [HDT-P-3] [HDT-P-4] [HF-P-1] [JA-G-2] [LO4-SL-1] [LO4-SL-3] [SL-G-1] [SL-G-2] [SF-T-1] [SF-T-2] [TH-P-1]
1001-10000	[GB3-G-1] [GB5-G-1] [GB5-G-2] [HU-G-1] [LO2-G-4][LO3-G-2] [LO3-G-3] [LO4-G-1] [LO4-G-2] [LO4-G-3] [UD-G-1] [UD-G-2] [UD-P-1] [UD-P-2] [UD-G-3]
10001-100000	[FF-G-1] [FF-G-2] [FF-G-3] [JA-G-1] [JA-G-3] [JL-G-1] [JL-G-2] [JL-G-3] [LO1-G-1] [LO1-G-2] [LO1-G-3] [LO2-G-1] [LO2-G-2] [LO2-G-3] [LO3-G-1] [LO4-G-1] [LO4-G-2] [LO4-G-3]
100000+	

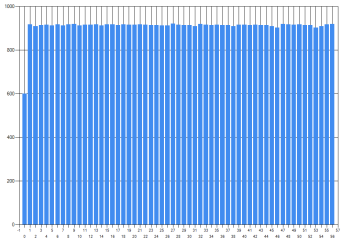
4.2 Flow parallelity

Flow parallelity is an important secondary feature to flow count. Flows can be sampled either in regular time intervals or per packet in real time. If two or more connections

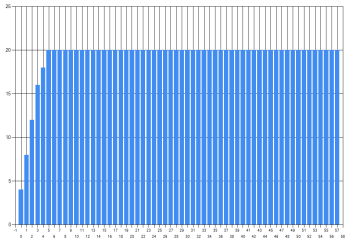
Figure 8: Flow count example



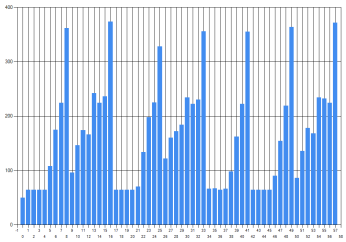
(a) [FF-G-2]



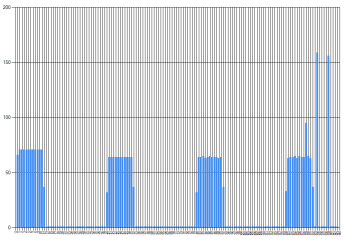
(b) [JA-G-3]



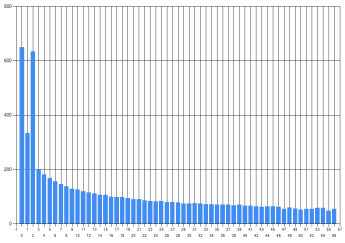
(c) [LO3-T-1]



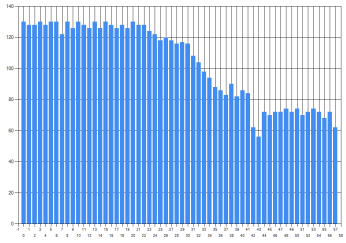
(d) [LO2-G-4]



(e) [HDT-P-1]



(f) [DS-T-1]



(g) [GB3-G-1]

have been established and closed within one time interval, these flows seem to be simultaneous even though in reality they might have existed in succesion. That may increase false positives rate of DoS detection systems. Oppositely, real time sampling is more resource demanding and prone to intermittent network errors, such as packet retransmissions. Both sampling types has been utilized extensively. We divide tools into four categories by flows parallelity.

- All simultaneous. Flows that are initiated in short succession and are never closed under normal circumstances. Attacker keeps these flows open for the duration of the attack and sends attack packets over them.
- Mostly simultaneous. Flows are closed after a prolonged time, usually by the victim after connection timeout. Many flows are open at the same time. Flows duration often exceeds 60 seconds.
- Long-term consecutive, many simultaneous. Generation and existence of flows themselves is one of the means of attack. Flows are generated rapidly, often by several process threads simulaneously. Flow duration varies with the performance of the attack tool.
- Mostly consecutive. Flows are established and closed in succession, eventually only a few flows overlaps. Attacks aim to overwhelm the victim with flow generation rate. Flows have very short duration.

Results of flow parallelity measurements in Tables 25 and 26 support our observations from flow count measurement. The level of flow parallelity decreases with flow count. Our observations show real time parallelity is not very common. Many tools actually produce flows in succession or in small batches of simultaneous flows. The outer effect of massive flow parallelity is caused by the length of flow sampling interval. With decreasing interval, thresholds for DoS detection via simultaneous flows count should be lowered in order to maintain detection accuracy, because the count of seemingly simultaneous flows will decrease. In contrast, the count of truly simultaneous flows would remain constant.

4.3 Flow packet count

Packet count is one of the most important properties of every flow. Collection of flow packet counts is trivial, being a part of NetFlow standard, yet it still has a high infor-

Table 25: Flows parallelity – TCP

All simultaneous	[SF-T-1] [SF-T-2]
Mostly simultaneous	[LO1-T-1] [LO1-T-2] [LO1-T-3] [LO1-T-4] [LO2-T-1] [LO2-T-2] [LO2-T-3] [LO2-T-4] [LO3-T-1] [LO3-T-2] [LO3-T-3] [LO3-T-4] [LO4-T-1] [LO4-T-2] [LO4-T-3] [DS-T-1] [DS-T-2]
Long-term consecutive, many simultaneous	[SD-T-1] [SD-T-2] [SD-T-3]
Mostly consecutive	[BD-T-1] [BD-T-2] [JL-T-1] [JL-T-2] [JL-T-3] [LC-T-1] [LC-T-2] [XO-T-1] [XO-T-2] [XO-T-3] [XO-T-4]

Table 26: Flows parallelity – HTTP

All simultaneous	[AD-G-1] [AD-G-2] [BAD-G-1] [BAD-G-2] [LC-G-1] [LC-G-2] [LC-G-3]
Mostly simultaneous	[GB3-G-1] [GB5-G-1] [GB5-G-2] [HDT-P-1] [HDT-P-2] [HDT-P-3] [HDT-P-4] [HU-G-1] [LO4-R-1] [LO4-R-2] [LO4-SL-1] [LO4-SL-2] [LO4-SL-3] [LO4-SL-4] [SL-G-1] [SL-G-2] [SL-G-3] [TH-P-1] [TH-P-2] [UD-G-3]
Long-term consecutive, many simultaneous	[LO1-G-1] [LO1-G-2] [LO1-G-3] [LO2-G-1] [LO2-G-2] [LO2-G-3] [LO2-G-4] [LO3-G-1] [LO3-G-2] [LO3-G-3] [LO4-G-1] [LO4-G-2] [LO4-G-3]
Mostly consecutive	[ADR-P-1] [FF-G-1] [FF-G-2] [FF-G-3] [HF-P-1] [HO-G-1] [HO-G-2] [HO-G-3] [JA-G-1] [JA-G-2] [JA-G-3] [JL-G-1] [JL-G-2] [JL-G-3] [UD-G-1] [UD-G-2] [UD-P-1] [UD-P-2]

mational value. Flooding DoS attacks operate with many flows and many packets. Our analysis focused on the characterization of flows by their packet numbers.

Siaterlis and Maglaris calculate 'flow length' metrics as a number of packets in a flow in the last 30 seconds [SM05]. Subsequently, flows with just one packet are suspected to be a part of an outgoing DoS attack with spoofed source IP address. We aim to broaden the scope of flow packet count metric. We believe it can be used to detect spoofed attacks, some classes of non-spoofed DoS attacks and, most importantly, it can serve as an indicator of similarity between seemingly unrelated flows.

Tables 27 and 28 show our findings for flow packet count of closed flows. Configurations that do not close flows under normal circumstances (AD-G, BAD-G, LC-G) were excluded from testing. Only packets packets sent by the attacker (i.e., initiator of the connection) are considered. Configurations were divided into three groups.

- All flows same packet count. All closed flows during the lifetime of the attack have the same packet count, with exceptions of packet retransmissions.
- Minimal differences. All flows have the packet count within 2 % of the packet count median rounded up (i.e., difference by one packet is always eligible). Flow packet count can serve as a secondary indice of flow similarity, but not as a deciding factor.
- Significant differences. No correlation between packet counts of individual flows have been found.

Flow packet count is a precise metric for detection of outgoing TCP-based DoS attacks. Of the tested tools, only LO is not suitable for the detection based on packet flow count. Spoofed SYN flooding attack produced by SF can be detected with by method of Siaterlis and Maglaris. Remaining TCP attack tools produce traffic where every flow has exactly the same packet count. We believe that when applied to high flow counts tools (e.g., SD, XO, JL), this metric can be both very precise and computationally efficient. The detection model would also be configurable in the terms of false positives rate. Precision can be devised from how many flow counts must be correctly predicted in order to consider those flows being a part of DoS attack. Adding to the effectiveness of the proposed method is also a relatively low flow packet count of TCP-based attacks. The purpose of normal traffic is to transmit data between communication participants. In terms of TCP protocol, three packets are required to establish the connection and one

or more packets to terminate the connection. Of those, two or more packets must be sent by connection initiator. Therefore, any terminated connection with two or less packets sent by initiator could not transmit any data.

The detection of HTTP-based attacks can work on the same basic principle, however there are less eligible tools. Especially tools with low number of flows (e.g., AD, BAD, LC) have packet count differences that are preventing succesful detection.

Table 27: Flows attacker packet count – HTTP

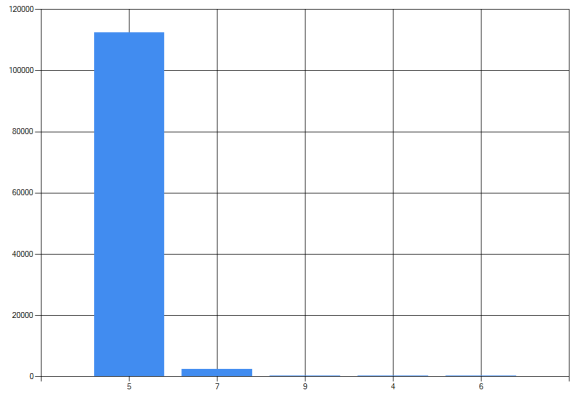
All flows same packet count	[ADR-P-1] [FF-G-1] [FF-G-2] [FF-G-3] [GB3-G-1] [GB5-G-1] [GB5-G-2] [HDT-P-1] [HDT-P-2] [HF-P-1] [JA-G-1] [JA-G-2] [JA-G-3] [JL-G-1] [JL-G-2] [JL-G-3] [LO1-G-1] [LO1-G-2] [SL-G-1] [SL-G-2] [SL-G-3] [UD-G-2] [UD-P-2]
Minimal differences	[LO2-G-1] [LO4-G-3] [LO4-SL-1] [LO4-SL-2] [LO4-SL-3] [LO4-SL-4] [UD-P-1]
Significant differences	[HDT-P-3] [HDT-P-4] [HO-G-1] [HO-G-2] [HO-G-3] [HU-G-1] [LO1-G-3] [LO2-G-2] [LO2-G-3] [LO2-G-4] [LO3-G-1] [LO3-G-2] [LO3-G-3] [LO4-G-1] [LO4-G-2] [LO4-R-1] [LO4-R-2] [TH-P-1] [TH-P-2] [UD-G-1] [UD-G-3]

Table 28: Flows attacker packet count – TCP

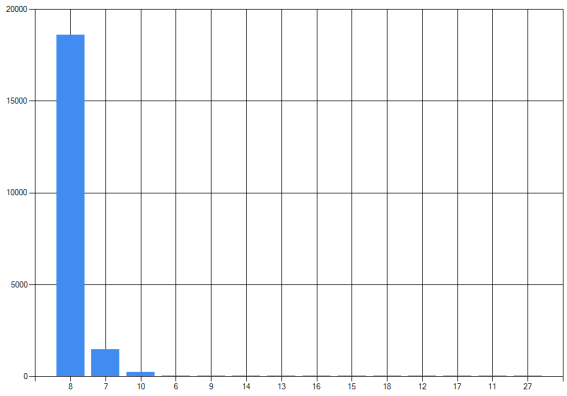
All flows same packet count	[BD-T-1] [BD-T-2] [DS-T-1] [DS-T-2] [JL-T-1] [JL-T-3] [LC-T-1] [LC-T-2] [SD-T-2] [SF-T-1] [SF-T-2] [XO-T-1] [XO-T-2] [XO-T-3] [XO-T-4]
Minimal differences	[SD-T-1]
Significant differences	[JL-T-2] [LO1-T-1] [LO1-T-2] [LO1-T-3] [LO1-T-4] [LO2-T-1] [LO2-T-2] [LO2-T-3] [LO2-T-4] [LO3-T-1] [LO3-T-2] [LO3-T-3] [LO3-T-4] [LO4-T-1] [LO4-T-2] [LO4-T-3] [SD-T-3]

Even though some tools have been put into Minimal differences group or Significant differences group, sometimes we can still find that a clear majority of flows has a similar value of flow packet count feature, for example, [JL-T-2] (Figure 9b), [LO1-G-3], [LO3-G-3], [LO4-G-3] or [SD-T-3] (Figure 9a). Configurations with more uniform distribution are for example [HDT-P-3], [HO-G-2] (Figure 9c), [LO3-G-2] or [LO4-G-2] (Figure 9d).

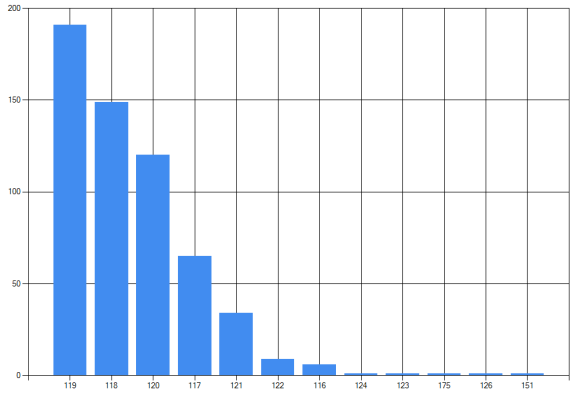
Figure 9: Flow packet count histogram example



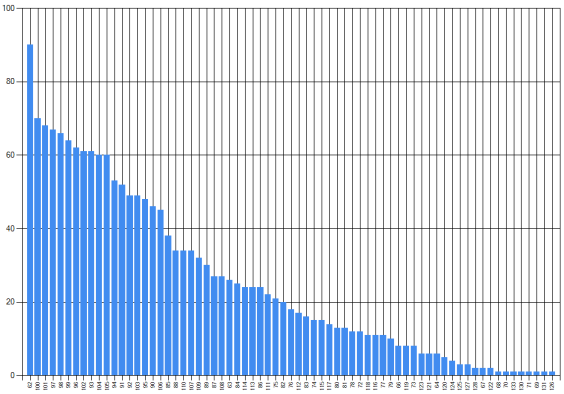
(a) [SD-T-3]



(b) [JL-T-2]



(c) [HO-G-2]



(d) [LO4-G-2]

4.4 TCP flag ratios

4.4.1 SYN outgoing to SYNACK incoming

During TCP 3-way handshake, three TCP segments are exchanged. SYN sent by connection initiator/client, SYNACK response from the server and ACK from client that confirms the succesful receival of the response. Attackers that employ IP spoofing in their packets are unable to finish the 3WH, because the server response is never received. The measurement differences SYN between SYNACK TCP segment couns is a natural approach to detect outgoing DoS attacks that employ IP spoofing. A system based on this metrics were suggested by Nashat et al. [NJH08]. We show the graphs of ratio of outgoing SYN TCP segments and incoming SYNACK TCP segments. An outgoing TCP SYN attack with IP spoofing is manifested by the ratio exceeding 1.0. We have encountered three various states. The attribution of configurations to states is provided in Table 29 and example graphs in Figures 10, 11 and 12.

- Stability. Ratio is longterm stable with minimal or none deviations. This is expectable from tools that rely on high number of flows. In controlled environment with zero packet drops the ratio is 1.0. Minor anomalies may be caused when a SYN segment is sent during a first sampling interval and SYNACK segment is received during the next sampling interval.
- Initiation phase. Ratio is countable only during a short initiation phase. After the end of initiation phase, no SYN and SYNACK TCP segments are exchanged. We define the initiation phase as first 5 seconds of the attack. This is expected behavior from tools that use a long-term established connections to deliver the attack payload.
- IP spoofing present. During intervals when SYN TCP segments are sent to the network, the ratio value highly exceeds 1.0. For fully spoofed attacks, the value is equal to the number of packets with a spoofed source IP address.

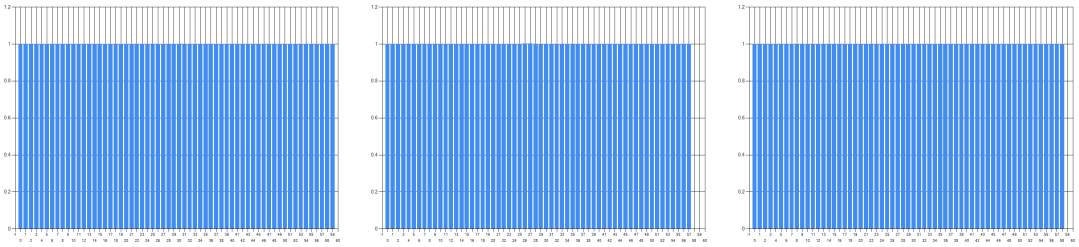
4.4.2 FIN segments to all segments ratio

TCP flows are usually closed both half of the flow independently. Both client and server transmit FIN TCP segment and acknowledge receive with ACK TCP segment. The responding side may react with FIN+ACK TCP segment, therefore lowering the number

Table 29: SYN outgoing to SYNACK incoming TCP segments ratio

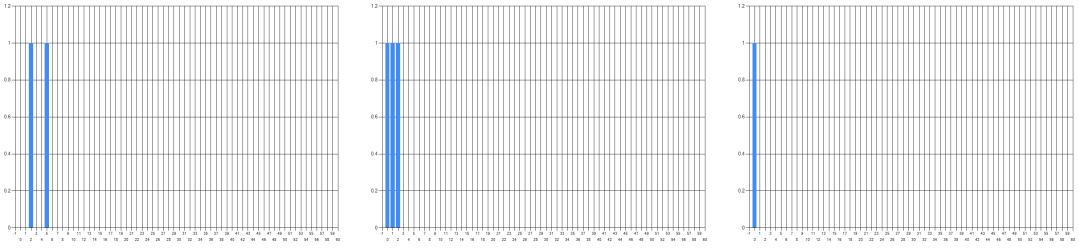
Stability	[ADR-P-1] [BD-T-1] [BD-T-2] [DS-T-1] [DS-T-2] [FF-G-1] [FF-G-2] [FF-G-3] [GB3-G-1] [GB5-G-1] [GB5-G-2] [HO-G-1] [HO-G-2] [HO-G-3] [HF-P-1] [HU-G-1] [JA-G-1] [JA-G-2] [JA-G-3] [JL-G-1] [JL-G-2] [JL-G-3] [JL-T-1] [JL-T-2] [JL-T-3] [LO1-G-1] [LO1-G-2] [LO1-G-3] [LO2-G-1] [LO2-G-2] [LO2-G-3] [LO2-G-4] [LO3-G-1] [LO3-G-2] [LO3-G-3] [LO4-G-1] [LO4-G-2] [LO4-G-3] [LO4-SL-1] [LO4-SL-3] [LC-T-1] [LC-T-2] [SD-T-1] [SD-T-2] [SD-T-3] [UD-G-1] [UD-G-2] [UD-P-1] [UD-P-2] [UD-G-3] [XO-T-1] [XO-T-2] [XO-T-3] [XO-T-4]
Initial phase	[AD-G-1] [AD-G-2] [BAD-G-1] [BAD-G-2] [HDT-P-1] [HDT-P-2] [HDT-P-3] [HDT-P-4] [LO1-T-1] [LO1-T-2] [LO1-T-3] [LO1-T-4] [LO2-T-1] [LO2-T-2] [LO2-T-3] [LO2-T-4] [LO3-T-1] [LO3-T-2] [LO3-T-3] [LO3-T-4] [LO4-R-1] [LO4-R-2] [LO4-SL-2] [LO4-SL-4] [LO4-T-1] [LO4-T-2] [LO4-T-3] [LC-G-1] [LC-G-2] [LC-G-3] [SL-G-1] [SL-G-2] [SL-G-3] [TH-P-1] [TH-P-2]
IP spoofing	[SF-T-1] [SF-T-2]

Figure 10: SYN outgoing to SYNACK incoming TCP segments ratio – Stability example



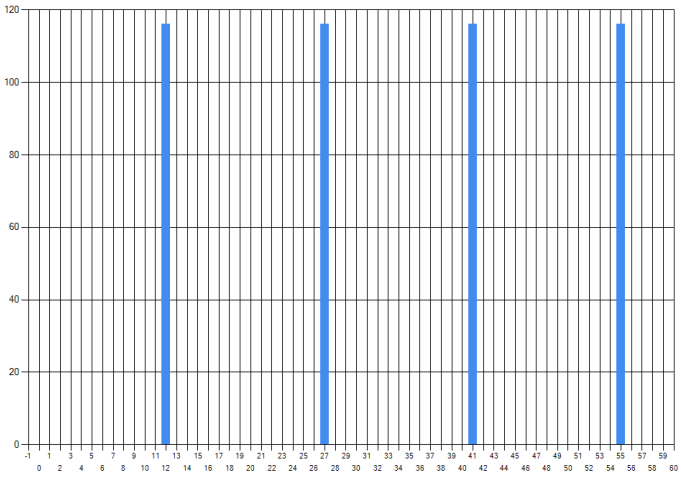
(a) [GB5-G-1] (b) [SD-T-1] (c) [XO-T-2]

Figure 11: SYN outgoing to SYNACK incoming TCP segments ratio – Initiation phase example



(a) [AD-G-2] (b) [LO1-T-4] (c) [TH-P-2]

Figure 12: SYN outgoing to SYNACK incoming TCP segments ratio – IP spoofing present example



(a) [SF-T-1]

of packets required for termination. As a second possibility, TCP flow may be aborted with RST TCP segment. RST segment terminates the flow immediately, no response from the other party is required. Famous paper by Wang, Zhang and Shin exploits the differences between the number of SYN TCP segments and FIN TCP segments. We explore the relationship between the number of FIN or RST segments and the number of all packets [WZS02].

Huge majority of tools always close a flow with the same combination of TCP flags. Grouping flows by their termination method is another measure to identify similarities between flows originating from a DDoS attack tool. Four distinct states have been observed regarding the ratio of FIN segments and all segments in our analysis in the initial 60 seconds of the attack. The attribution of configurations to states is provided in Table 30 and example graphs in Figures 13, 14 and 15.

- No FIN TCP segments. Most tools do not close TCP connections with FIN TCP segments or the closure takes place with delay that is longer than our 60 second observation window. This is somewhat surprising, because FIN closure is default for ordinary connections. Included are tools that would not be closed during the lifetime of the attack (e.g., AD, ADR, LC) and configurations with flows that are closed with RST TCP segments.
- Stability. Ratio holds approximately the same value in long-term.
- Anomaly. No predictions can be made about the ratio. In case of GB, the cause is an uneven distribution of HTTP requests in attack traffic.
- Repeated pattern. Patterns with LO1 are caused by regular peaks of packet rate. Steady packet rate would result in stable ratio.

4.4.3 RST segments to all segments ratio

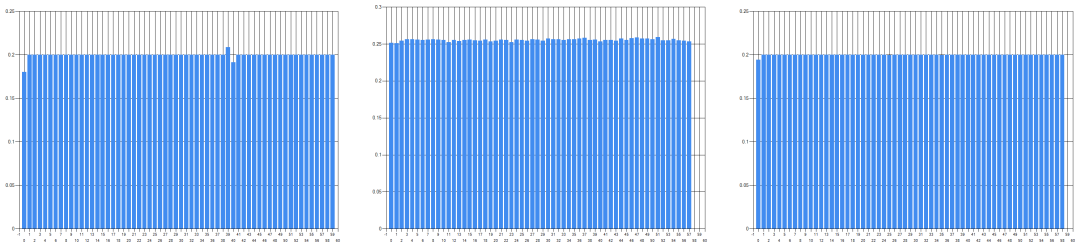
TCP segments with RST flags are second option how to terminate a TCP flow. States are similar to states observed with FIN TCP segments. Classification of configurations is provided in Table 31 and examples in Figures 16, 17 and 18.

- No RST TCP segments. Flows are closed with FIN segment or are not closed during the sample window at all.

Table 30: FIN segments to all segments ratio

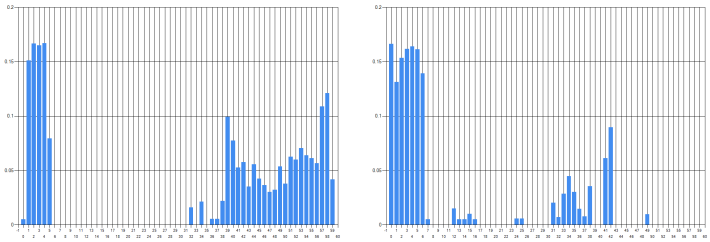
None	[AD-G-1] [AD-G-2] [ADR-P-1] [BAD-G-1] [BAD-G-2] [DS-T-1] [DS-T-2] [FF-G-2] [HDT-P-1] [HDT-P-2] [HDT-P-3] [HDT-P-4] [HF-P-1] [HU-G-1] [LO1-T-1] [LO1-T-2] [LO1-T-3] [LO1-T-4] [LO2-G-1] [LO2-G-2] [LO2-G-3] [LO2-G-4] [LO2-T-1] [LO2-T-2] [LO2-T-3] [LO2-T-4] [LO3-G-1] [LO3-G-2] [LO3-G-3] [LO3-T-1] [LO3-T-2] [LO3-T-3] [LO3-T-4] [LO4-G-1] [LO4-G-2] [LO4-G-3] [LO4-T-1] [LO4-T-2] [LO4-T-3] [LO4-R-1] [LO4-R-2] [LO4-SL-1] [LO4-SL-2] [LO4-SL-3] [LO4-SL-4] [LC-G-1] [LC-G-2] [LC-G-3] [LC-T-1] [LC-T-2] [SL-G-1] [SL-G-2] [SL-G-3] [SF-T-1] [SF-T-2] [TH-P-1] [TH-P-2] [UD-G-1] [UD-G-2] [UD-P-1] [UD-P-2] [UD-G-3]
Stability	[BD-T-1] [BD-T-2] [FF-G-1] [FF-G-3] [HO-G-1] [HO-G-2] [HO-G-3] [JA-G-1] [JA-G-2] [JA-G-3] [JL-G-1] [JL-G-2] [JL-G-3] [JL-T-1] [JL-T-2] [JL-T-3] [SD-T-1] [SD-T-2] [SD-T-3] [XO-T-1] [XO-T-2] [XO-T-3] [XO-T-4]
Anomaly	[GB3-G-1] [GB5-G-1] [GB5-G-2]
Repeated pattern	[LO1-G-1] [LO1-G-2] [LO1-G-3]

Figure 13: FIN segments to all segments ratio – Stability example



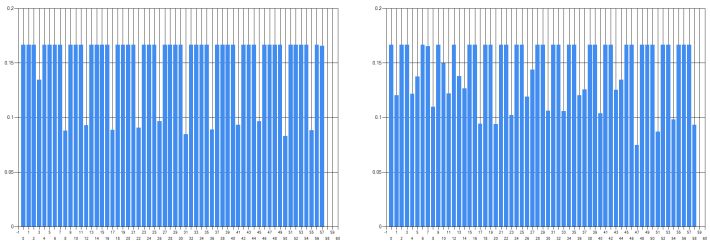
(a) [BD-T-2] (b) [JL-G-3] (c) [SD-T-2]

Figure 14: FIN segments to all segments ratio – Anomaly example



(a) [GB5-G-2] (b) [GB3-G-1]

Figure 15: FIN segments to all segments ratio – Repeated pattern



(a) [LO1-G-1] (b) [LO1-G-2]

- Stability. Ratio holds approximately the same value in long-term.
- Planks. Corresponds to repeated patterns state of TCP RST segments. Caused by LO closing flows in batches with idle periods in between and by SF sending all packets in batches.
- Anomaly. Ratio does not yield any observable trend.

Table 31: RST segments to all segments ratio

None	[AD-G-1] [AD-G-2] [ADR-P-1] [BAD-G-2] [DS-T-1] [DS-T-2] [FF-G-2] [GB3-G-1] [GB5-G-1] [GB5-G-2] [HO-G-1] [HO-G-2] [HO-G-3] [HDT-P-1] [HDT-P-2] [HDT-P-3] [HDT-P-4] [HF-P-1] [JA-G-3] [LO1-T-1] [LO1-T-2] [LO1-T-3] [LO1-T-4] [LO2-T-1] [LO2-T-2] [LO2-T-3] [LO3-T-1] [LO3-T-2] [LO3-T-3] [LO3-T-4] [LO4-T-1] [LO4-T-2] [LO4-T-3] [LO4-R-1] [LO4-R-2] [LO4-SL-1] [LO4-SL-2] [LO4-SL-3] [LO4-SL-4] [LC-G-1] [LC-G-2] [LC-G-3] [SL-G-1] [SL-G-2] [SL-G-3] [TH-P-1] [TH-P-2] [UD-G-1] [UD-G-2] [UD-P-1] [UD-P-2] [UD-G-3]
Stability	[BD-T-1] [BD-T-2] [FF-G-1] [FF-G-3] [JA-G-1] [JA-G-2] [JL-G-2] [JL-G-3] [JL-T-1] [JL-T-3] [LC-T-1] [LC-T-2] [SD-T-1] [SD-T-2] [SD-T-3] [XO-T-1] [XO-T-2] [XO-T-3] [XO-T-4]
Planks	[LO1-G-1] [LO1-G-2] [LO1-G-3] [LO2-G-1] [LO2-G-2] [LO2-G-3] [LO2-G-4] [LO2-T-4] [LO3-G-1] [LO3-G-2] [LO3-G-3] [LO4-G-1] [LO4-G-2] [LO4-G-3] [SF-T-1] [SF-T-2]
Anomaly	[BAD-G-1] [HU-G-1] [JL-G-1] [JL-T-2]

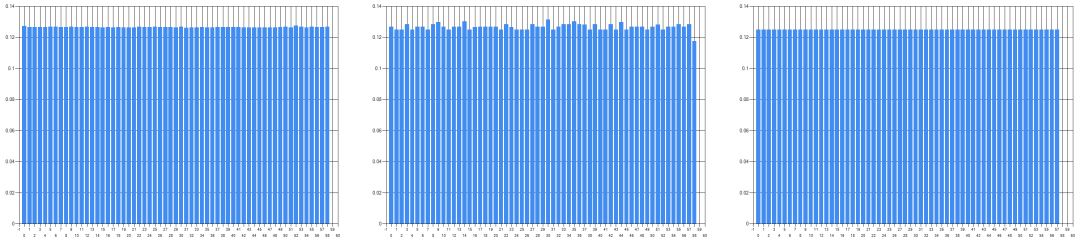
4.4.4 NS, ECE, CWR, URG to all TCP segments ratio

None of the tools in the set has created TCP segments with NS, CWR, ECE or URG flag set. The reasons may be these flags are not as widely used as the other core flags, therefore it would be possible to filter out attack traffic trivially.

4.5 Average flow duration

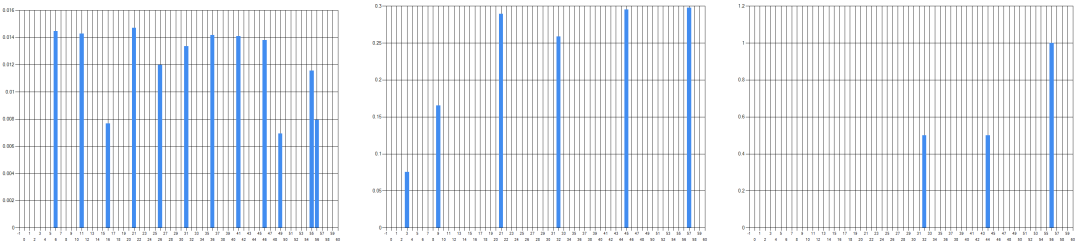
Duration is an important and an easily measurable property of the flow. Similar duration of many flows may be just another indicator of mutual resemblance. Also, there

Figure 16: RST segments to all segments ratio – Stability example



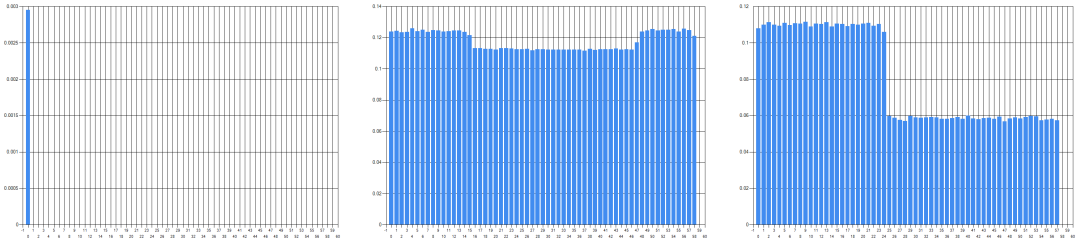
(a) [FF-G-1] (b) [JA-G-2] (c) [XO-T-2]

Figure 17: RST segments to all segments ratio – Planks example



(a) [HU-G-1] (b) [LO3-G-2] (c) [SF-T-1]

Figure 18: RST segments to all segments ratio – Anomaly example



(a) [BAD-G-1] (b) [JL-G-1] (c) [JL-T-2]

exist several key values of flow duration that can help estimate the outcome of the flow. For example, flows taking 130 seconds might have been closed after server timeout. Flow duration has been utilized for detection of DDoS attacks by Tao et al. in [TYP+09]. Siaterlis and Maglaris measure the number of flows with duration less than 10 ms [SM05]. Oppositely, Galtsev and Sukhov consider flows with duration longer than 300 seconds [GS11]. Braga et al. measure the average duration of flows in order to decrease false positives when there is a small number of packets exchanged [BMP10].

We provide summary of flow measurement in Table 32. Full results are available in Tables 33, 34 and 35. Majority of TCP-based attacks produce very short flows. Unsurprisingly, HTTP-based attacks that use malformed HTTP requests also generate flows with short average duration. We can observe there are configurations with very small relative standard deviation (LO-T, SL, TH). Small variance in flow duration for these configurations is the manifestation of batch flow establishment. Long flows are characteristic for slow attacks and HTTP attacks with multiple requests over each flow. LO-T is the only example of TCP-based attack that spans over period of more than one second.

We are convinced that flow duration metric has a decent potential for detection of outgoing DDoS attacks. It is measured effectively with NetFlow, applicable to all types of attacks and previous works have shown its practical useability. We especially highlight its convenience for detection of TCP-based attacks with high number of flows. For these types of attacks, flow duration measurement can become a quick and reasonably precise DoS detection input feature. Poor detection swiftness is an obvious handicap for the detection of DoS attacks with long flows. However, we still perceive the useability as a signature-based metrics that can be used to identify flows that were closed after a key time period. Such key time period is 130 seconds in our case, which seems to be a default timeout interval of IIS 7.0 webserver.

5 HTTP behavior

5.1 HTTP requests success

Success rate of client HTTP request is an interesting metric to use. While occasional errors can be expected, most standard HTTP requests are resolved successfully in the longterm. Following events can alter the HTTP request success and failure ratio:

Table 32: Flow duration – Summary

0 – 100 ms	[ADR-P-1] [DS-T-1] [DS-T-2] [FF-G-1] [FF-G-2] [FF-G-3] [GB3-G-1] [GB5-G-1] [GB5-G-2] [HO-G-1] [HO-G-2] [HO-G-3] [HDT-P-2] [HF-P-1] [HU-G-1] [JA-G-1] [JA-G-2] [JA-G-3] [JL-G-1] [JL-G-2] [JL-G-3] [JL-T-1] [JL-T-2] [JL-T-3] [LC-T-1] [LC-T-2] [SD-T-2] [UD-G-1] [UD-P-1] [XO-T-1] [XO-T-2] [XO-T-3] [XO-T-4]
101 ms – 1 s	[BD-T-1] [BD-T-2] [SD-T-1] [SD-T-3] [SF-T-1] [SF-T-2] [SL-G-1] [UD-G-2] [UD-P-2]
1 s – 10 s	[LO1-G-1] [LO1-G-2] [LO1-G-3] [LO2-G-1] [LO2-G-2] [LO2-G-3] [LO2-G-4] [LO3-G-1] [LO3-G-2] [LO3-G-3] [LO4-G-1] [LO4-G-3]
> 10 s	[AD-G-1] [AD-G-2] [BAD-G-1] [BAD-G-2] [HDT-P-1] [HDT-P-3] [HDT-P-4] [LO1-T-1] [LO1-T-2] [LO1-T-3] [LO1-T-4] [LO2-T-1] [LO2-T-2] [LO2-T-3] [LO3-T-1] [LO3-T-2] [LO3-T-3] [LO3-T-4] [LO4-G-2] [LO4-R-1] [LO4-R-2] [LO4-SL-1] [LO4-SL-2] [LO4-SL-3] [LO4-SL-4] [LO4-T-1] [LO4-T-2] [LO4-T-3] [LC-G-1] [LC-G-2] [LC-G-3] [SL-G-2] [SL-G-3] [TH-P-1] [TH-P-2] [UD-G-3]

- Target server overwhelmed. If the DDoS attack is successful and the target is gradually overwhelmed, one of the possible outcomes is an increase in error responses from server.
- HTTP request poorly constructed. Many attack tools in our analysis were designed for maximal sending performance, often even at the expense of following standards. Missing or invalidly populated header fields were observed. Filtering for flows with high failure rate will reveal destination IP addresses, that are having connection issues. Subsequent flow protocol validation may reveal if the failure is caused by poorly constructed request.
- Network failure. Failure of internet uplink will result in lost connectivity to the majority of servers in Internet. This possibility can be partially distinguished from focused DDoS attacks, because during the attack the legitimate traffic is not exhibiting altered behavior.

Table 36 shows our observations about the success rate of HTTP DoS tools. We reiterate and emphasize that tools and the victim server were tuned in order to keep server

Table 33: Flow duration Part 1

Config ID	Avg	Max	Min	SD	RSD	Count
[ADR-P-1]	0.0014	0.0114	0.0000	0.0005	36%	938
[AD-G-1]	28.2488	53.5429	2.9546	35.7713	127%	2
[AD-G-2]	28.0777	53.1523	3.0031	35.4608	126%	2
[BAD-G-1]	29.4374	58.6171	0.2577	41.2663	140%	2
[BAD-G-2]	58.6889	58.8207	58.5570	0.1865	0%	2
[BD-T-1]	0.1093	0.1119	0.0004	0.0034	3%	1061
[BD-T-2]	0.1093	0.1405	0.0003	0.0035	3%	1068
[DS-T-1]	0.0006	0.0045	0.0000	0.0009	160%	3190
[DS-T-2]	0.0004	0.0036	0.0000	0.0006	143%	3250
[FF-G-1]	0.0006	0.0154	0.0003	0.0005	72%	88449
[FF-G-2]	0.0006	0.0208	0.0001	0.0004	60%	90438
[FF-G-3]	0.0006	0.0266	0.0002	0.0003	49%	91621
[GB3-G-1]	0.0012	0.0385	0.0000	0.0032	260%	3009
[GB5-G-1]	0.0020	0.048	0.0000	0.0066	327%	3363
[GB5-G-2]	0.0023	0.0528	0.0000	0.0070	312%	3342
[HO-G-1]	0.0358	0.2823	0.0150	0.0162	45%	290
[HO-G-2]	0.0383	0.6452	0.0166	0.0312	81%	580
[HO-G-3]	0.0369	0.4548	0.0151	0.0241	65%	530
[HDT-P-1]	40.2030	40.2156	40.1905	0.0069	0%	400
[HDT-P-2]	0.0085	0.386	0.0009	0.0424	500%	400
[HDT-P-3]	34.4402	59.3839	0.1963	16.4264	48%	400
[HDT-P-4]	15.9257	57.0776	0.0007	18.7415	118%	400
[HF-P-1]	0.0073	0.7182	0.0000	0.0526	718%	939
[HU-G-1]	0.0402	25.1578	0.0000	0.5952	1481%	1798
[JA-G-1]	0.0010	0.0066	0.0000	0.0004	38%	26296
[JA-G-2]	0.0010	0.005	0.0000	0.0003	27%	533
[JA-G-3]	0.0015	0.028	0.0000	0.0007	49%	26244
[JL-G-1]	0.0051	0.1806	0.0000	0.0055	109%	52449
[JL-G-2]	0.0009	0.0152	0.0002	0.0008	93%	54539
[JL-G-3]	0.0008	0.0157	0.0000	0.0007	81%	56522
[JL-T-1]	0.0008	0.0901	0.0000	0.0014	189%	82553

Table 34: Flow duration Part 2

Config ID	Avg	Max	Min	SD	RSD	Count
[JL-T-2]	0.0042	0.0472	0.0000	0.0055	130%	20456
[JL-T-3]	0.0007	0.0167	0.0000	0.0006	81%	71313
[LO1-G-1]	2.3076	10.3934	0.0003	1.4452	63%	17387
[LO1-G-2]	1.7054	6.593	0.0003	0.9439	55%	60877
[LO1-G-3]	2.8775	23.6393	0.0000	4.9639	173%	67230
[LO1-T-1]	55.1059	57.841	52.3495	1.7078	3%	10
[LO1-T-2]	53.7746	58.4762	50.5862	2.4658	5%	10
[LO1-T-3]	56.0803	58.2867	54.5122	1.5607	3%	5
[LO1-T-4]	56.0613	57.4233	54.8967	0.9772	2%	5
[LO2-G-1]	1.8304	5.6529	0.0000	1.6133	88%	16127
[LO2-G-2]	5.9598	45.541	0.0000	12.8220	215%	20934
[LO2-G-3]	3.2682	54.5414	0.0000	2.4853	76%	17279
[LO2-G-4]	4.1139	8.5306	0.0000	2.3841	58%	1849
[LO2-T-1]	53.9854	58.4887	49.5677	2.9930	6%	10
[LO2-T-2]	54.5070	58.8819	50.0166	3.0145	6%	10
[LO2-T-3]	56.4958	58.4759	54.4940	1.5642	3%	5
[LO2-T-4]	0.0003	0.0005	0.0002	0.0001	52%	5
[LO3-G-1]	2.4553	10.4998	0.0010	1.4827	60%	17281
[LO3-G-2]	5.4782	25.0519	0.0000	3.6445	67%	3725
[LO3-G-3]	5.5278	28.7546	0.0018	4.6426	84%	1858
[LO3-T-1]	55.3803	58.0563	52.6357	1.6760	3%	10
[LO3-T-2]	55.7212	58.4092	52.9784	1.6750	3%	10
[LO3-T-3]	56.6614	58.0005	55.5723	0.9376	2%	5
[LO3-T-4]	56.8953	58.2339	55.8207	0.9316	2%	5
[LO4-G-1]	3.1134	13.7684	0.0000	2.1474	69%	8764
[LO4-G-2]	14.4561	38.6941	0.0023	11.6941	81%	1837
[LO4-G-3]	3.7483	18.2816	0.0015	2.8575	76%	3691
[LO4-R-1]	56.5123	58.4393	44.5326	2.4885	4%	50
[LO4-R-2]	44.1579	56.8907	34.9020	10.6802	24%	15
[LO4-SL-1]	25.7346	33.2687	0.0007	12.0514	47%	500
[LO4-SL-2]	30.4306	30.6527	30.1989	0.1361	0%	75

Table 35: Flow duration Part 3

Config ID	Avg	Max	Min	SD	RSD	Count
[LO4-SL-3]	22.7357	33.2744	0.0006	13.7977	61%	500
[LO4-SL-4]	30.4195	30.6387	30.2008	0.1356	0%	75
[LO4-T-1]	55.8061	58.4824	53.0576	1.6750	3%	10
[LO4-T-2]	55.6772	58.3595	52.9320	1.6728	3%	10
[LO4-T-3]	56.9255	58.2771	55.8384	0.9399	2%	5
[LC-G-1]	54.4606	54.7849	54.1363	0.4586	1%	2
[LC-G-2]	54.7416	54.7418	54.7414	0.0003	0%	2
[LC-G-3]	55.1862	55.1862	55.1862	—	—	1
[LC-T-1]	0.0193	0.425	0.0001	0.0394	204%	27394
[LC-T-2]	0.0161	0.4228	0.0003	0.0174	108%	6869
[SD-T-1]	0.1941	8.4353	0.0000	0.6478	334%	119161
[SD-T-2]	0.0504	0.0773	0.0001	0.0063	13%	115101
[SD-T-3]	0.4076	16.2237	0.0000	1.2419	305%	115552
[SL-G-1]	0.4172	0.5274	0.3943	0.0236	6%	500
[SL-G-2]	30.2226	30.2758	30.1919	0.0171	0%	500
[SL-G-3]	50.2318	50.272	50.1888	0.0190	0%	100
[SF-T-1]	0.1100	21.0025	0.0000	1.4384	1308%	464
[SF-T-2]	0.1100	21.0323	0.0000	1.4393	1308%	464
[TH-P-1]	56.8418	57.916	54.8748	0.6904	1%	256
[TH-P-2]	57.7713	58.6353	56.0571	0.6791	1%	64
[UD-G-1]	0.0704	0.4216	0.0026	0.0953	135%	3729
[UD-G-2]	0.2117	0.8378	0.0014	0.0441	21%	2125
[UD-P-1]	0.0085	0.6873	0.0007	0.0466	551%	3710
[UD-P-2]	0.2088	0.8745	0.0021	0.0458	22%	3780
[UD-G-3]	22.5169	43.3079	0.0034	13.7701	61%	2152
[XO-T-1]	0.0006	0.0066	0.0001	0.0004	65%	100082
[XO-T-2]	0.0006	0.011	0.0001	0.0004	67%	102668
[XO-T-3]	0.0006	0.0068	0.0002	0.0004	68%	101496
[XO-T-4]	0.0006	0.0066	0.0003	0.0004	64%	102510

stable and not overwhelmed. The produced results were the consequence of adhering to standards both by the attacker and by the victim.

Special case is Slowloris. This tool is designed to tie available server connections by slowly sending long HTTP request. During our defined sample window this HTTP request is never finished, therefore the response is not observed. Normally, Microsoft IIS server reacts by closing the connection with RST segment.

Interesting is also the development of LOIC. Version 1.0.4.0 produces HTTP requests for which IIS responds with error 400 Bad request. Later versions are corrected and the response is properly received. Oppositely, JavaLOIC do not implement this modification and produces minimal HTTP requests, that cause error 400.

Table 36: HTTP requests success

Ok	[AD-G-1] [AD-G-2] [BAD-G-1] [BAD-G-2] [HO-G-1] [HO-G-2] [HO-G-3] [HDT-P-1] [HDT-P-2] [HDT-P-3] [HDT-P-4] [HU-G-1] [LO2-G-1] [LO2-G-2] [LO2-G-3] [LO2-G-4] [LO3-G-1] [LO3-G-2] [LO3-G-3] [LO4-G-1] [LO4-G-2] [LC-G-1] [LC-G-2] [LC-G-3] [UD-G-1]
Malformed connection, closed due to timeout	[SL-G-1] [SL-G-2] [SL-G-3]
Error 40x	[ADR-P-1] [FF-G-1] [FF-G-2] [FF-G-3] [GB3-G-1] [GB5-G-1] [GB5-G-2] [HF-P-1] [JA-G-1] [JA-G-2] [JA-G-3] [JL-G-1] [JL-G-2] [JL-G-3] [LO1-G-1] [LO1-G-2] [LO1-G-3] [LO4-G-3] [TH-P-1] [TH-P-2] [UD-G-2] [UD-G-3] [UD-P-1] [UD-P-2]

5.2 HTTP requests per flow

Number of outgoing HTTP requests per the number of flows for a single destination IP address can be considered a decent detection metric. The normal traffic consists both of TCP flows with only one HTTP request and of TCP flows that carry multiple HTTP requests along with respective responses. Therefore on average, the number of HTTP requests exchanged over destination port 80 is higher than the number of TCP flows with this destination port. This important characteristic is only rarely emulated by DoS tools. Volume-based attack tools produce many HTTP requests and their distribution

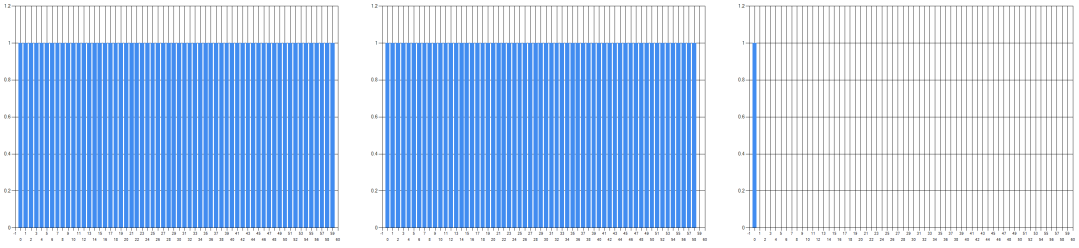
between flows is often very straightforward, as can be seen in Table 37 and Figure 19. FF and HO are classic examples of tools with many flows, one HTTP request each. LO Recoil attack is a slow attack with never finished HTTP header, therefore only the first part of each segmented HTTP request is marked.

- One per flow. Each established TCP flow is closed after at most one HTTP request is sent from the attacker to the victim. The ratio between the number of HTTP requests and the number of TCP flows carrying HTTP protocol messages convergates to 1.
- Multiple per flow. Established TCP flows can carry one or more separate HTTP requests and respective responses. Of the tested tools, none has exhabitied such behavior with chosen configurations.
- Infinite per flow. TCP flows carrying attack HTTP requests are never closed under normal circumstances and the request sending has not been observed to be stopping during our analysis. The ratio between the number of HTTP requests and the number of TCP flows carrying HTTP protocol messages during each interval is much higher than 1 and is usually copying packet rate curve.

Table 37: HTTP requests per flow

One per flow	[ADR-P-1] [FF-G-1] [FF-G-2] [FF-G-3] [GB3-G-1] [GB5-G-1] [GB5-G-2] [HO-G-1] [HO-G-2] [HO-G-3] [HDT-P-1] [HDT-P-2] [HDT-P-3] [HDT-P-4] [HF-P-1] [HU-G-1] [JA-G-1] [JA-G-2] [JA-G-3] [JL-G-1] [JL-G-2] [JL-G-3] [LO1-G-1] [LO1-G-2] [LO1-G-3] [LO2-G-1] [LO2-G-2] [LO2-G-3] [LO2-G-4] [LO4-G-1] [LO4-G-2] [LO4-G-3] [LO4-R-1] [LO4-R-2] [LO4-SL-1] [LO4-SL-2] [LO4-SL-3] [LO4-SL-4] [SL-G-1] [SL-G-2] [SL-G-3] [TH-P-1] [TH-P-2] [UD-G-1] [UD-G-2] [UD-P-1] [UD-P-2] [UD-G-3]
Multiple per flow	
Infinite per flow	[AD-G-1] [AD-G-2] [BAD-G-1] [BAD-G-2] [LC-G-1] [LC-G-2] [LC-G-3]

Figure 19: HTTP request per flow count



(a) [FF-G-2] (b) [HO-G-2] (c) [LO4-R-2]

5.3 HTTP request method

Request method indicates the action that is required from the server. RFC 2616 defines 8 methods: OPTIONS, GET, HEAD, POST, PUT, DELETE, TRACE, CONNECT. In practice, most HTTP requests use GET or POST method. We can expect only these two most common methods to be used during DDoS attacks. Employing rare request method would simplify both the harmful traffic detection and filtering. From the source end DDoS detection perspective, measuring the ratio between GET and remaining methods may be used, especially for the detection of POST flooding. Excessive amount of POST requests are worth close investigation.

None of the tools in our analysis have used other method than GET or POST. Requests produced by tools were also always homogenous, without switching between different methods during the attack.

Table 38: HTTP requests method

GET	[AD-G-1] [AD-G-2] [BAD-G-1] [BAD-G-2] [FF-G-1] [FF-G-2] [FF-G-3] [GB3-G-1] [GB5-G-1] [GB5-G-2] [HO-G-1] [HO-G-2] [HO-G-3] [HU-G-1] [JA-G-1] [JA-G-2] [JA-G-3] [JL-G-1] [JL-G-2] [JL-G-3] [LO1-G-1] [LO1-G-2] [LO1-G-3] [LO2-G-1] [LO2-G-2] [LO2-G-3] [LO2-G-4] [LO3-G-1] [LO3-G-2] [LO3-G-3] [LO4-G-1] [LO4-G-2] [LO4-G-3] [LO4-R-1] [LO4-R-2] [LO4-SL-3] [LO4-SL-4] [LC-G-1] [LC-G-2] [LC-G-3] [SL-G-1] [SL-G-2] [SL-G-3] [UD-G-1] [UD-G-2] [UD-G-3]
POST	[ADR-P-1] [HDT-P-1] [HDT-P-2] [HDT-P-3] [HDT-P-4] [HF-P-1] [LO4-SL-1] [LO4-SL-2] [TH-P-1] [TH-P-2] [UD-P-1] [UD-P-2]

5.4 HTTP requests URIs

We are convinced URI monitoring one of the most important metrics that can be used to verify the presence of an outgoing DDoS attack in a given traffic sample. Observing repeated similar URIs either within one HTTP flow or within multiple flows with very similar characteristics justifies raising an alert for a security operator to investigate manually. There are several methods that can be employed in order to make comparison of HTTP URIs difficult. Tables 39 and 40 show a detailed listing of URI modifications observed during our analysis.

Table 39: HTTP requests URIs

Same HTTP requests among multiple flows	[ADR-P-1] [FF-G-1] [FF-G-2] [FF-G-3] [GB3-G-1] [GB5-G-1] [GB5-G-2] [HO-G-1] [HO-G-2] [HO-G-3] [HDT-P-1] [HDT-P-2] [HDT-P-3] [HDT-P-4] [HF-P-1] [JA-G-1] [JA-G-2] [JA-G-3] [JL-G-1] [JL-G-3] [LO1-G-1] [LO1-G-2] [LO1-G-3] [LO2-G-1] [LO2-G-2] [LO2-G-3] [LO2-G-4] [LO3-G-1] [LO3-G-2] [LO3-G-3] [LO4-G-1] [LO4-G-2] [LO4-G-3] [LO4-R-1] [LO4-R-2] [LO4-SL-1] [LO4-SL-2] [LO4-SL-3] [LO4-SL-4] [LC-G-1] [LC-G-2] [LC-G-3] [SL-G-1] [SL-G-2] [SL-G-3] [TH-P-1] [TH-P-2] [UD-G-1] [UD-P-1] [UD-G-3]
Completely different URLs	[JL-G-2] [UD-G-2] [UD-P-2]
Same base URL, different parameters	[AD-G-1] [AD-G-2] [BAD-G-1] [BAD-G-2] [HU-G-1]

- URI string set. The tool attacks not just one URI, but a predefined set of URIs. Using a set may slightly downgrade the attack efficiency, because not only the most resource demanding page is chosen to be the target, but also several other. Moreover, if chosen URIs are not as critical as the primary target, selected HTTP requests may be blocked before reaching the intended target. URI string set modification can be overcome by grouping the flows by destination address. Even if the threshold is set per-URI, the attack may be detected if the set is small. Enlargening the set signifies efficiency drop.

- Page crawling. The tool starts with an initial URI and gets more URIs by parsing the links in the HTTP response. Page crawling may seem like an efficient method, but its implementation is difficult. Current webpages frequently host links that lead to other webservers and following them would result in attack traffic being spread among many more targets. Therefore, the tool would have to allow to configure limitations for crawling. Also, the impact on tool performance might be significant. Attack tools do not process the response and doing so would require much more resources on the attacker machine. Crawling can be employed to create an URI set which is subsequently used. In such situation, listed limitations for string sets apply. In our analysis, no tool employed page crawling nor predefined URI set.

- Parameter change. The base domain and file path remains constant, but full URI is made unique by adding unique parameter values. This method allows to specify the precise target, while maintaining a decent level of stealthiness. Unique parameter values also render webpage caching servers between the attacker and the victim useless, therefore make the attack mitigation more difficult. Obvious countermeasure is to only consider the base path. While it is not often possible for the purposes of attack mitigation, it is a solid approach for the purposes of an outgoing DDoS attack detection. Figures 20 and 21 provide a sample comparison of HTTP request count when full URIs are considered and when parameters are dropped. Dropped parameters result in constant unique HTTP request count equal to 1.

  - Per-thread random parameter. LO pursues an interesting policy. For each process thread a random string is generated and subsequently the string is appended to URIs of all HTTP requests produced by this thread. This policy is reasonable from the performance point of view. Also, victim-based DDoS detection systems can be initially overcome if the attack is sufficiently distributed. Over time however, the constant parameter may be used filter out attack flows by routers and firewalls on the path between the attacker and the victim. Per-thread parameters are also useless against source-end DDoS detection systems. Example of a [LO3-G-2] request is
    "GET index.htm?LFMLND".

  - Timestamp parameter. Each request has a timestamp information that makes it unique. Timestamps we observed had Unix timestamp format. Example of a [BAD-G-2] request is
    "GET index.htm?id=1383307797379&msg=We%20Are%20Legion".
  - Random key=value parameter. HU adds a structure that mimics standard URI parameters. Random key name and random key value are separated with "=" sign. Example HTTP request is
    "GET index.htm?TJSXF=SQGUISOS".

- Random URI. URI may be fully randomly generated. That presents a challenge for the attack detection and mitigation, but the attack effectiveness is severely degraded. A huge majority of responses is error 400, therefore the web server does not saturate its outgoing bandwidth and also do not devote as much computational power to retrieve the response. Alternatively, only a parameter part of the URI may be random. In such case, limitations listed for parameter changes apply.

Table 40: HTTP requests URIs modifications

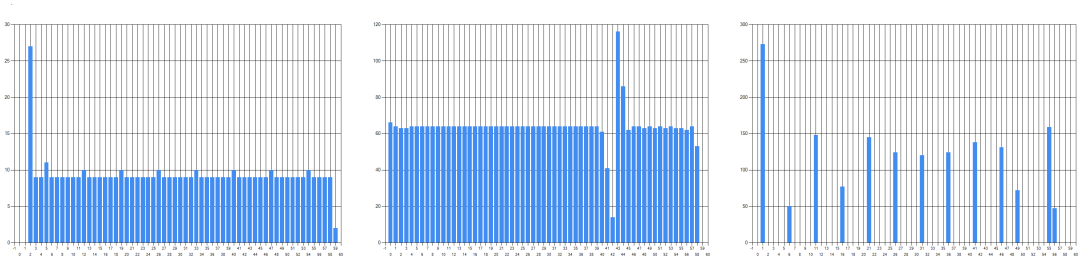
Random URI	[JL-G-2] [UD-G-2] [UD-P-2]
Per-thread random parameter	[LO3-G-2] [LO3-G-3] [LO4-G-2] [LO4-G-3]
Timestamp parameter	[AD-G-1] [AD-G-2] [BAD-G-1] [BAD-G-2]
Random key=value parameter	[HU-G-1]

5.5 HTTP header fields

5.5.1 User-Agent

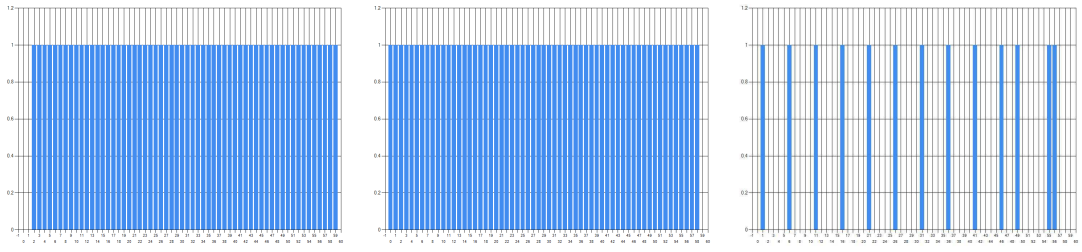
User-Agent (UA) field in HTTP header serves as an identifier of the client sending the request. It's purpose is mainly to allow to provide different content for different situations, e.g. the server may respond with webpage suitable for small displays if the request comes from a mobile device browser. User-Agent string is selected by the creator of the client application. Theoretically, it is possible for creators of malware and hacking

Figure 20: HTTP request URIs – Unique HTTP request count



(a) [AD-G-1] (b) [BAD-G-2] (c) [HU-G-1]

Figure 21: HTTP request URIs – Unique HTTP request without parameters count



(a) [AD-G-1] (b) [BAD-G-2] (c) [HU-G-1]

tools to mimic UA strings of popular software. However in practice, UA strings have been succesfully used to track generic malware [MV11]. We have identified three filling methods for UA field. Classification of DDoS tools by the methods is given in Table 41.

- Missing UA. Even though UA string is not a required HTTP header field, it can be found in the vast majority of HTTP requests. Many DDoS tools in our set are however so focused on requests per second rate that the UA field is not included. Tracking the number of HTTP requests destined for the same IP address and measuring the percentage of requests without UA field can become a decent anomaly detection method.

- Static UA. UA field is filled with a static string, depending on the tool in question. Most of the tools mimic existing web browsers, which makes the UA field unsuitable for detection purposes. However, there exist also tools which either allow to insert an arbitrary static string (e.g., [HDT-P-3]) or which fill the field with easily distinguishable string (e.g., [JA-G-1]). In these situations, UA field can be used both for signature detection and traffic clustering.

- Dynamic UA. The field is filled with a string from a limited set or the string is randomly generated, possibly from basic building blocks. Dynamic UA aim to confuse victim-based detection systems, which cannot subsequently cluster the traffic by UA strings. However, this technique makes the tool very visible for source-end based detection systems. A simple threshold scheme for the maximal number of unique UAs, implemented easily for example with cumulative Bloom filters, can detect the presence of such tool on the host.

5.5.2 Referer

Referer HTTP header field contains the address of a website from which the URI in current request was linked. This field is intended mostly for the purposes of advertising and user profiling. As can be seen in Table 42, huge majority of DDoS tools do not include this header in the request. This approach is not unexpected, because inserting a static referer string could make requests clustering easier while inserting dynamic referer string presents a non-trivial problem.

Configurations [BAD-G-1] and [BAD-G-2] always use a static string "http://127.0.0.1/?a=e". This string is probably default value for ZzeePhpExe

Table 41: HTTP requests User-Agents

User-agent field missing	[ADR-P-1] [GB3-G-1] [GB5-G-1] [GB5-G-2] [HO-G-1] [HO-G-2] [HO-G-3] [HF-P-1] [JL-G-1] [JL-G-2] [JL-G-3] [LO1-G-1] [LO1-G-2] [LO1-G-3] [LO2-G-1] [LO2-G-2] [LO2-G-3] [LO2-G-4] [LC-G-1] [LC-G-2] [LC-G-3] [UD-G-1] [UD-P-1] [UD-G-3]
User-agent field static	[AD-G-1] [AD-G-2] [BAD-G-1] [BAD-G-2] [FF-G-1] [FF-G-2] [FF-G-3] [HDT-P-1] [HDT-P-2] [HDT-P-3] [HDT-P-4] [JA-G-1] [JA-G-2] [JA-G-3] [LO3-G-1] [LO3-G-2] [LO3-G-3] [LO4-G-1] [LO4-G-2] [LO4-G-3] [SL-G-1] [SL-G-2] [SL-G-3]
User-agent field dynamic	[HU-G-1] [TH-P-1] [TH-P-2] [UD-G-2] [UD-P-2]

compiler this tool was built with. [HU-G-1] employes much more sophisticated technique. Each referer is unique. It consists of an URI from a predefined set (e.g., "www.usatoday.com/search/results", "www.google.com") followed by a key "q" with a random value consisting of 5-10 capital letters. Example is "http://www.usatoday.com/search/results?q=AVBPW". Although it is possible to collect all URIs in the set and subsequently to construct a regular expression that will filter only traffic produced by HU, we consider this solution very time consuming and inflexible for practical use. Referer field populating by HU is an example of advanced DoS HTTP request construction with focus on making the detection as difficult as possible.

5.5.3 Accept-Encoding and Accept-Language

Two remaining HTTP header fields occasionally encountered at requests generated by DoS tools are Accept-Encoding and Accept-Language. Accept-Encoding determines what types of encoding are understood and preferred by the client application, Accept-Language field specifies output in what language is acceptable for the client application respectively. We did not discover any interesting anomalies regarding the usage of these two fields. Tables 43 and 44 show that even though these fields are utilized occasionally, most tools create HTTP requests without them.

Accept-Language field for AD and BAD is filled with value "en-US\r\n". Conversely, HO uses "en\r\n". Both these values are common and valid. All tools ex-

Table 42: HTTP requests header fields – Referer

Referer field missing	[AD-G-1] [AD-G-2] [ADR-P-1] [FF-G-1] [FF-G-2] [FF-G-3] [GB3-G-1] [GB5-G-1] [GB5-G-2] [HO-G-1] [HO-G-2] [HO-G-3] [HDT-P-1] [HDT-P-2] [HDT-P-3] [HDT-P-4] [HF-P-1] [JA-G-1] [JA-G-2] [JA-G-3] [JL-G-1] [JL-G-2] [JL-G-3] [LO1-G-1] [LO1-G-2] [LO1-G-3] [LO2-G-1] [LO2-G-2] [LO2-G-3] [LO2-G-4] [LO3-G-1] [LO3-G-2] [LO3-G-3] [LO4-G-1] [LO4-G-2] [LO4-G-3] [LC-G-1] [LC-G-2] [LC-G-3] [SL-G-1] [SL-G-2] [SL-G-3] [TH-P-1] [TH-P-2] [UD-G-1] [UD-G-2] [UD-P-1] [UD-P-2] [UD-G-3]
Referer field populated	[BAD-G-1] [BAD-G-2] [HU-G-1]

cept HU contain an expected string "gzip, deflate\r\n" in their Accept-Encoding HTTP header field. HU fills the field with a reserved word "identity\r\n". This codeword tells the receiving server that any encoding is acceptable for the client.

Table 43: HTTP requests header fields – Accept-Encoding

Accept-Encoding field missing	[ADR-P-1] [FF-G-1] [FF-G-2] [FF-G-3] [GB3-G-1] [GB5-G-1] [GB5-G-2] [HO-G-1] [HO-G-2] [HO-G-3] [HDT-P-1] [HDT-P-2] [HDT-P-3] [HDT-P-4] [HF-P-1] [JL-G-1] [JL-G-2] [JL-G-3] [LO1-G-1] [LO1-G-2] [LO1-G-3] [LO2-G-1] [LO2-G-2] [LO2-G-3] [LO2-G-4] [LO3-G-3] [LO4-G-1] [LO4-G-2] [LO4-G-3] [SL-G-1] [SL-G-2] [SL-G-3] [TH-P-1] [TH-P-2] [UD-G-1] [UD-G-2] [UD-P-1] [UD-P-2] [UD-G-3]
Accept-Encoding field valid	[AD-G-1] [AD-G-2] [BAD-G-1] [BAD-G-2] [HU-G-1] [JA-G-1] [JA-G-2] [JA-G-3] [LO3-G-1] [LO3-G-2]

Table 44: HTTP requests header fields – Accept-Language

Accept-Language field missing	[ADR-P-1] [FF-G-1] [FF-G-2] [FF-G-3] [GB3-G-1] [GB5-G-1] [GB5-G-2] [HDT-P-1] [HDT-P-2] [HDT-P-3] [HDT-P-4] [HF-P-1] [HU-G-1] [JL-G-1] [JL-G-2] [JL-G-3] [LO1-G-1] [LO1-G-2] [LO1-G-3] [LO2-G-1] [LO2-G-2] [LO2-G-3] [LO2-G-4] [LO3-G-1] [LO3-G-2] [LO3-G-3] [LO4-G-1] [LO4-G-2] [LO4-G-3] [LC-G-1] [LC-G-2] [LC-G-3] [SL-G-1] [SL-G-2] [SL-G-3] [TH-P-1] [TH-P-2] [UD-G-1] [UD-G-2] [UD-P-1] [UD-P-2] [UD-G-3]
Accept-language field valid	[AD-G-1] [AD-G-2] [BAD-G-1] [BAD-G-2] [HO-G-1] [HO-G-2] [HO-G-3]

6 Summary

6.1 Attacks diversity

Network traffic generated by tools in our set present a variety of DoS attacks. Even though it was possible to classify attacks by the basic concept, every attack was unique in some regard.

6.2 Traffic features

Although almost every traffic feature that we measured yielded some results, none proved to be sufficient alone for the detection of DoS attacks in the source-end network. Every feature can detect only a subset of existing DoS attacks. We believe that an aggregation of multiple features is necessary to be used for a general detection. Standalone features suffer from false positives, but more importantly, have an unexceedable limit of false negatives rate. Different classes of DoS attacks have different properties and none of the traffic features could be applied to all. We support the approach taken for example by [SM05], [OL07] and [TYP+09] who collect multiple feature values and subsequently compute their aggregate importance.

Serious consideration must be given not only to the computational efficiency of features' values analysis, but also to an efficient collection of input values. Features included in NetFlow standard are therefore preferred. However, as our results show, this limited set of flow-based statistics and network layer features may not be sufficient for the reliable confirmation of some classes of DoS attacks. In order to balance the complexity of collection and processing of some features and potentially huge amounts of packets/flows for analysis, sampling and filtering of suspicious flows may be employed prior to the analysis. We believe that the analysis process separated into several stages as proposed, for example, by Wang et al. [WWWS12] is promising.

Traditional metrics such as high bitrate and high packetrate are by themselves not reliable options for source-end detection. By definition, slow attacks are hardly detectable via metrics focused on high volumes. Also, many tools enable to specify the attack performance so it is possible to find a configuration which cannot be detected through volume-based metrics.

6.3 Repeating patterns

Most important observation of this work is that standalone DoS attack tools *traffic comprises of repeating operations*. Every attack has a basic construction unit which is iterated in time, creating a series of similar operations. Although some characteristics of operations may change with each iteration, most defining properties are constant. Construction units may have a form of flows with distinct characteristics in case of TCP-based attacks or HTTP requests and according responses in case of HTTP-based attacks.

Example 1 – [BD-T-1]. The traffic comprises of separate attack flows. Each flow is to be considered an operation. Each flow has the same packet count, packet size distribution and is carrying TCP segments. Each flow has the same TCP flags composition. The flow is always established via a correct 3-way handshake and terminated by the attacker with TCP FIN segments, which is followed by TCP RST segment from the victim. None of the TCP segments transmits any payload. All of the TCP header options fields of packets in one flow have the same values as the equivalent packets in other flows. All flows have a very short duration, 99 % of them takes between 0.1 and 0.12 seconds. None of the packets in the flow is fragmented, has the TTL value altered or is using a spoofed IP address.

Example 2 – [AD-G-2]. Attacker opens a fixed number of simultaneous flows towards the victim. Repeated HTTP requests are sent over each flow. Each HTTP request is an operation. All packets with HTTP requests have the same length, TTL field value and are not fragmented. Header of every HTTP request contains the same fields with the same values. Referer field is always missing. Full URI comprises of a basic path and parameters. The path is similar across all flows. The parameter is numeric and is gradually rising, while the second parameter is a static string.

Example 3 – [LO3-G-2]. Attack comprises of many flows. Each flow is an operation. Each flow is opened with a correct 3WH, one HTTP request is sent towards the victim and the flow is closed. Similarities between different flows can be found on IP layer (e.g., packet sizes, packet count from attacker), TCP layer (e.g., TCP flag ratios, window size, TCP checksum) and HTTP layer (e.g., HTTP fields found in the header, URI, parameters).

Noise traffic can be filtered out once DoS operations are identified. Subsequently, traffic can be analyzed on high scale. Patterns such as packet rate burst behavior, flow count in time or flow paralellity are recognizable. Existing DoS detection methods can be applied to the filtered traffic with an increased accuracy.

Pattern recognition opens a new area of how detect outgoing DoS attacks at the source end. This novel approach presents challenges how to recognize construction units in a traffic that contains both benign traffic and malicious traffic, how to determine which unit properties are constant and how to apply chosen pattern matching in time efficiently. Benefits are high precision growing with each next correctly identified operation and possibility to detect yet unknown attacks. Since repeating patterns have been identified across all classes of attacks, it can become a basis of a very broad detection method.

6.4 Evasion techniques

Most tools do not support any type of detection evasion techniques. Even if supported, they are not enabled by default. Most frequent are various kinds of randomization (e.g., packet payload, HTTP request URI). Randomization is usually configurable only for packet fields chosen by tool creator. Therefore, effect of randomization can be negated if multiple input features are analyzed in conjunction. Exception to the rule is HOIC, which provides environment for randomization of almost all fields in HTTP header.

6.5 Experiment designs

The consequence of diversity of attacks and class-specific features is that features for DoS detection are not mutually comparable. We want to encourage researchers to always state what their basic assumptions about DoS attacks are and what features are processed. If possible, at least a rudimentary analysis of properties of both the training traffic and the real traffic that was used in the experiment should be provided. Efficiency measurement and methods comparison should be performed only within a scope of attacks that satisfies chosen assumptions.

6.6 Tools characteristics

Majority of tools does not require root privileges and so can be executed on computers at work, school or internet cafe. Basic work with tools does not require advanced knowledge about the victim or the type of attack. Most tools allow targeting only one victim at a time. This is an important observation for source-end detection, because statistics of multiple flows aimed at a single target can be included in detection. One tool usually supports 3 different types of attack at most. The actual attack configuration may vary

considerably. We encountered tools both with plenty options as well as tools that do not support any configuration except choice of target. Standalone DoS tools are not intended for automated use. Majority of them have a simple GUI without corresponding command-line interface.

7 Conclusions

This technical report provides a comprehensive analysis of attack traffic generated by existing and widely used standalone DoS attack tools. We analyze the attack traffic in perspectives of various traffic features. Traffic features are divided according to the ISO/OSI layer at which they can be measured. For each feature we divide tool behavior into classes, provide characteristics of each class and attribute DoS tools to these classes.

Our analysis shows that DoS attack traffic comprises of repeating operations. A close examination of attack traffic can always identify independent building blocks – operations. These operations are mostly similar with minimum variability. The presence and nature of operations can rarely be observed from the perspective of just one traffic feature. Usually, an aggregation of several traffic features is required for reasonable characterization of an operation. Since operations are omnipresent at all attack traffic samples in our analysis, we propose a new research area for the detection of DoS attacks at the source end that is based on repeated attack pattern recognition. Our future work will focus on efficient aggregation of several traffic features in order to be able to identify repeating DoS attack patterns in observed network traffic.

None of the DoS detection metrics in our analysis proved to be sufficient for the detection of all types of DoS attacks by its own. For example, volume-based DoS detection metrics cannot reliably detect slow attacks, while protocol validation metrics are unsuccessful during HTTP attacks comprising of legitimate requests. As a consequence, mutual comparison of detection systems can be performed only on types of attacks for which these metrics are suitable. Therefore, we urge researchers to include their assumptions about detectable attack types, their properties and possible detection evasion techniques in every research output/publication that is dealing with DoS attack detection.

Standalone DoS attack tools are widely available, trivial to use, can generate a full scale of existing DoS attacks and are regularly observed during real hacktivist campaigns. Some of them are being constantly developed, adding new attack types and

countermeasures against defense systems in the process. We are convinced that standalone DoS attack tools can be used both for understanding of modern DoS attacks and for testing new detection systems. We encourage the use of these tools, opposite to obsolete tools still prevailing in academia such as TFN2k, Trinoo or Shaft.

Acknowledgements

Authors wish to acknowledge support of the Czech research project VG20102014031, programme BV II/2 - VS.

References

[AMG+12] Esraa Alomari, Selvakumar Manickam, B. B. Gupta, Shankar Karuppayah, and Rafeef Alfaris. Botnet-based Distributed Denial of Service (DDoS) Attacks on Web Servers: Classification and Art. *International Journal of Computer Applications*, 49(7):24–32, July 2012. Published by Foundation of Computer Science, New York, USA.

[BKBK13] Monowar H. Bhuyan, H. J. Kashyap, D. K. Bhattacharyya, and J. K. Kalita. Detecting Distributed Denial of Service Attacks: Methods, Tools and Future Directions. *The Computer Journal*, 2013.

[BMP10] R. Braga, E. Mota, and A. Passito. Lightweight DDoS flooding attack detection using NOX/OpenFlow. In *2010 IEEE 35th Conference on Local Computer Networks (LCN)*, pages 408–415, Oct 2010.

[CYL+09] Jieren Cheng, Jianping Yin, Yun Liu, Zhiping Cai, and Min Li. DDoS attack detection algorithm using IP address features. In *Frontiers in Algorithmics*, pages 207–215. Springer, 2009.

[DA07] Ping Du and Shunji Abe. Detecting DoS attacks using packet size distribution. In *Bio-Inspired Models of Network, Information and Computing Systems, 2007. Bionetics 2007. 2nd*, pages 93–96. IEEE, 2007.

[DPV06] Alberto Dainotti, Antonio Pescapé, and Giorgio Ventre. Wavelet-based Detection of DoS Attacks. In *Global Telecommunications Conference*, 2006.

[EN11] Jeff Edwards and Jose Nazario. A survey of contemporary Chinese DDoS malware. In *Proceedings of the 21st Virus Bulletin International Conference*. Virus Bulletin Ltd, 2011.

[EYA12] Benishti Eyal, Balmas Yaniv, and Matan Atad. #OPISRAEL, November 2012. Threat Alert, Radware.

[GP01] Thomer M. Gil and Massimiliano Poletto. MULTOPS : a data-structure for bandwidth attack detection. In *Proceedings of the 10th conference on USENIX Security Symposium*, volume 10, 2001.

[GS11] Aleksey A. Galtsev and Andrei M. Sukhov. Network Attack Detection at Flow Level. In Sergey Balandin, Yevgeni Koucheryavy, and Honglin Hu, editors, *Smart Spaces and Next Generation Wired/Wireless Networking*, volume 6869 of *Lecture Notes in Computer Science*, pages 326–334. Springer Berlin Heidelberg, 2011.

[LKK+08] Keunsoo Lee, Juhyun Kim, Ki Hoon Kwon, Younggoo Han, and Sehun Kim. DDoS attack detection method using cluster analysis. *Expert Systems with Applications: An International Journal*, 34(3):1659–1665, April 2008.

[LMSD09] Vicky Laurens, Alexandre Miege, Abdulmotaleb El Saddik, and Pulak Dhar. DDoSniffer: Detecting DDoS Attack at the Source Agents. *International Journal of Advanced Media and Communication*, 3(3):290–311, July 2009.

[LO09] Georgios Loukas and Gülay Öke. Protection Against Denial of Service Attacks: A Survey. *The Computer Journal*, 53(7):1020–1037, August 2009.

[MR05] Jelena Mirkovic and Peter Reiher. D-WARD: a source-end defense against flooding denial-of-service attacks. *IEEE Transactions on Dependable and Secure Computing*, 2(3):216–232, 2005.

[MTJ08] S. Malliga, A. Tamilarasi, and M. Janani. Filtering spoofed traffic at source end for defending against DoS/DDoS attacks. In *2008 International Conference on Computing, Communication and Networking*, pages 1–5. IEEE, 2008.

[MV11] Darren Manners and Robert Vandenbrink. The user agent field: Analyzing and detecting the abnormal or malicious in your organization. 2011. SANS Institute.

[Nat13] National Cybersecurity and Communications Integration Center, Department of Homeland security. OpUSA: Potential Tools. Technical report, May 2013.

[Net12] Arbor Networks. Worldwide infrastructure security report. Technical report, 2012.

[NJH08] Dalia Nashat, Xiaohong Jiang, and Susumu Horiguchi. Router based detection for Low-rate agents of DDoS attack. In *2008 International Conference on High Performance Switching and Routing*, pages 177–182, May 2008.

[OG10] Sean-Philip Oriyano and Michael Gregg. *Hacker Techniques, Tools, and Incident Handling*. Jones and Bartlett Publishers, Inc., USA, 1st edition, 2010.

[oIA13] Office of Intelligence and Department of Homeland security Analysis. OpUSA: Criminal Hackers Planning Cyber Attacks Against US Websites. Technical report, May 2013.

[OL07] Gülay Öke and Georgios Loukas. A Denial of Service Detector based on Maximum Likelihood Detection and the Random Neural Network. *The Computer Journal*, 50(6), September 2007.

[Pax01] Vern Paxson. An analysis of using reflectors for distributed denial-of-service attacks. *ACM SIGCOMM Computer Communication Review*, 31(3):38–47, July 2001.

[PYHR10] Pyungkoo Park, HeeKyoung Yi, SangJin Hong, and JaeCheul Ryu. An Effective Defense Mechanism against DoS / DDoS Attacks in Flow-based routers. In *Proceedings of the 8th International Conference on Advances in Mobile Computing and Multimedia*, 2010.

[SA07] Amey Shevtekar and Nirwan Ansari. A Proactive Test Based Differentiation Technique to Mitigate Low Rate DoS Attacks. In *2007 16th International Conference on Computer Communications and Networks*, August 2007.

[SA11] Manjula Suresh and R. Anitha. Evaluating Machine Learning Algorithms for Detecting DDoS Attacks. In David Wyld, Michal Wozniak, Nabendu Chaki, Natarajan Meghanathan, and Dhinaharan Nagamalai, editors, *Advances in Network Security and Applications*, volume 196 of *Communications in Computer and Information Science*, pages 441–452. Springer Berlin Heidelberg, 2011.

[Sha13] Pavitra Shankdhar. DOS Attacks and Free DOS Attacking Tools, October 2013. Webpage, http://resources.infosecinstitute.com/dos-attacks-free-dos-attacking-tools/ (4/4/2014).

[SM05] Christos Siaterlis and Vasilis Maglaris. Detecting Incoming and Outgoing DDoS Attacks at the Edge Using a Single Set of Network Characteristics. In *10th IEEE Symposium on Computers and Communications (ISCC'05)*, 2005.

[SWH11] Sin-Seok Seo, Young J. Won, and James Won-Ki Hong. Witnessing distributed denial-of-service traffic from an attacker's network. In *Network and Service Management (CNSM), 2011 7th International Conference on*, pages 1–7, 2011.

[Tea12] TeamDangerHackers. Anonymous - #Op MYANMAR, December 2012. Webpage, https://www.youtube.com/watch?v=flNwuDge-EE (4/4/2014).

[TSD07] Vrizlynn L. Thing, Morris Sloman, and Naranker Dulay. A Survey of Bots Used for Distributed Denial of Service Attacks. In Hein Venter, Mariki Eloff, Les Labuschagne, Jan Eloff, and Rossouw Solms, editors, *New Approaches for Security, Privacy and Trust in Complex Environments*, volume 232 of *IFIP International Federation for Information Processing*, pages 229–240. Springer US, 2007.

[TYP+09] Ran Tao, Li Yang, Lu Peng, Bin Li, and Alma Cemerlic. A case study: Using architectural features to improve sophisticated denial-of-service attack detections. In *2009 IEEE Symposium on Computational Intelligence in Cyber Security*, March 2009.

[Unk12] Author Unknown. Anon_doxing #OPISREAL, 2012. Webpage, http://pastebin.com/9M0HLC3d (4/4/2014).

[Wil12] Curt Wilson. Attack of the Shuriken: Many Hands, Many Weapons, February 2012. Webpage, http://www.arbornetworks.com/asert/2012/02/ddos-tools/ (4/4/2014).

[WWWS12] Fei Wang, Hailong Wang, Xiaofeng Wang, and Jinshu Su. A new multistage approach to detect subtle DDoS attacks. *Mathematical and Computer Modelling*, 55(1):198 – 213, 2012. Advanced Theory and Practice for Cryptography and Future Security.

[WZS02] Haining Wang, Danlu Zhang, and Kang G. Shin. Detecting SYN flooding attacks. In *INFOCOM 2002. Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, volume 3, pages 1530–1539. IEEE, 2002.

[XCH06] Bin Xiao, Wei Chen, and Yanxiang He. A novel approach to detecting DDoS Attacks at an Early Stage. *The Journal of Supercomputing*, 36(3):235–248, June 2006.

[YKP+13] Jaehak Yu, Hyunjoong Kang, DaeHeon Park, Hyo-Chan Bang, and Do Wook Kang. An in-depth analysis on traffic flooding attacks detection and system using data mining techniques. *Journal of Systems Architecture*, 59(10, Part B):1005 – 1012, 2013. Advanced Smart Vehicular Communication System and Applications.