Analyzing traffic features of common standalone DoS attack tools

Vit Bukac and Vashek Matyas

Centre for Research on Cryptography and Security
Faculty of Informatics
Masaryk University
Brno, Czech Republic
bukac@mail.muni.cz, matyas@fi.muni.cz

Abstract. Research on denial of service (DoS) attack detection is complicated due to scarcity of reliable, widely available and representative contemporary input data. Efficiency of newly proposed DoS detection methods is continually verified with obsolete attack samples and tools. 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 provide a classification of different attack traffic features, including utilized evasion techniques and encountered anomalies. We also propose a new research direction for the detection of DoS attacks at the source end, based on repeated attack patterns recognition.

Keywords: denial of service tools, input features, traffic characteristics

1 Introduction

Even though denial of service (DoS) attacks are steadily gaining on popularity among both cyber criminals and security researchers, there are only few studies collecting *thorough and truly representative characteristics* of DoS attack traffic. We observe a serious discrepancy between tools that are used by attack perpetrators and the tools that are used for testing DoS 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.

Understanding how attacks evolve is a necessary step towards the design of appropriate DoS attack detection and mitigation systems. Conversely, academic concepts are notoriously evaluated with obsolete and in practice already forgotten tools, most notably TFN, TFN2k, Shaft, Trinoo, Knight, mstream and Stacheldraht that all date back to year 2000. We still encounter numerous research works that present these tools as representatives of modern DoS attacks, even in respectable periodics (e.g., [2, 7, 26]). These tools no longer reflect contemporary real DoS attacks. DoS attacks went through an incredible development not only in terms of overall performance, but also in terms of attack properties. A brand new class of slow application DoS attacks has emerged and gained on popularity.

Simultaneously, contemporary labeled DoS attack datasets are sparse. Both DARPA and KDD99 datasets are still being used, despite being 15 years old and not representing current state in networks. For example, DoS category in KDD99 contains back, land, neptune, pod, smurf and teardrop attacks, none of which are seen in the wild any more. Other available datasets are produced by projects such as CAIDA, MAWI or ONTIC. However, these datasets either do not provide attack labeling or suffer from little DoS attack variability.

Our research focused on state-of-the art DoS tools, their DoS traffic properties, employed evasion techniques and further tools characteristics. Network traffic profiles of standalone DoS tools will help design detection methods that are based on valid assumptions. Also, by creating a database of attack tools, it will be possible to estimate what classes of DoS attacks can be detected by each proposed method.

Our traffic traces contain only attack traffic. Each trace file is labeled with the name of DoS tool that was used to generate the traffic, attack type and any attack configuration options. Therefore, our traces are suitable for evaluation of DoS detection and mitigation systems through attractive *overlay methodology*. Overlay methodology combines the separate attack traffic traces with the background traces from an arbitrary environment. This widespread methodology allows to recognize the ground truth and precisely determine the false positives rate and false negative rate of the evaluated detection system [5].

Our analysis will assist best to researchers focusing on *source-end DoS detection* solutions, such as D-WARD [17]. Mirkovic et al. argue that detecting DoS attacks directly at the source computers or first-mile routers brings benefits such as congestion avoidance, small collateral damage, easy traceback and the possibility to use sophisticated detection strategies due to more available resources [16]. Source-end solutions also bring significant advantages when applied in cloud environments, software-defined networking or untrustworthy networks. We believe the importance of source-end detection is proven by the development direction of host-based antimalware products. Security companies are gradually introducing more and more network analysis modules to their products, including DDoS detection modules, such as in the Symantec Endpoint Protection. Capability to detect outbound DoS attacks coupled with originator system process identification is a viable behavioral malware detection

This paper is supported by more technical details available in our technical report [8]. Our dataset of all PCAP traces and used DoS tools is freely available at [1]. Our paper presents the following contributions:

1. This is the first comparative study aimed exclusively at traffic properties of DoS attack tools. We overview existing state-of-the-art standalone DoS attack tools, their attack traffic properties and used evasion techniques. We provide conclusive evidence that no two DoS tools (of those we examined) generate DoS attack traffic with the same properties.
2. Identification of network traffic features that are suitable for the source-end DoS attack detection. We evaluated the importance of selected features for various classes of DoS attacks. We reject traffic randomization as an universal

answer to the DoS detection evasion. We propose a new area for detection of DoS attacks based on recognition of unique repeating patterns.
3. Support for the use of these traffic traces for evaluation of DoS intrusion detection systems in academic research thorough overlay methodology.

The remainder of this paper is organized as follows. Section 2 reviews relevant work in the traffic analysis of DoS tools. Section 3 describes our experiment and the selection of tools for analysis. Section 4 supplies our raw observations of the selected DoS traffic properties. Section 5 summarizes our results and highlights the impact of our analysis on DoS attacks detection. Section 6 concludes the paper.

2 Related work

Exploration of detailed properties of DoS attacks in the wild received limited to none interest from academia. This may be because of an assumption that DoS attacks cannot notably alter their properties, otherwise they would have to sacrifice performance or increase visibility. On the other hand, some of the most prominent state-of-the-art DoS tools are occasionally examined by freelance security specialists or companies dealing with DDoS protection solutions. Such analyses are often thorough and descriptive, but lack a mutual comparison and frequently skip deriving general concepts.

The tools listed in this paper are extensively and routinely used by hacktivists to manifest their political opinions and by technically unsavvy users to harass other users. Bartolacci et al. describe the practice of "kicking", when online gamers use simple DoS tools to degrade their oponent's network connection or even force them out of the game [6].

Onut and Ghorbani argue there is a general lack of research on input features [19]. They investigated the effectiveness ranking of 673 network features for the detection of network attacks. Their evaluation concludes that for the detection of DoS attacks the best features are related to ICMP protocol. For TCP-based attacks, they emphasize the importance of SYN packet statistics and flow statistics. Another DARPA dataset traffic features analyzing paper was presented by Kabiri and Zargar [15]. They note the SYN flag presence, classification fields and protocol fields as most influential. Slightly enhanced DARPA 2000 dataset was analyzed by Zi et al. [27]. Their list of top 5 preferable features (in decreasing order) is TCP SYN occurrence, destination port entropy, entropy of source port, UDP protocol occurrence and packet volume. Unfortunately, the results based on DARPA and KDD datasets has been repeatedly criticized for not being a good representative sample of actual traffic in a network (e.g., [13, 21]).

Thing et al. [23] performed a detailed source code analysis of selected then popular bots for distributed DoS (DDoS) attacks, namely Agobot, SDBot, RBot and Sybot. Authors emphasize the importance of randomization in creating a packet, which is a view we share. Given the source code availability, this analysis is very descriptive with a deep understanding of inner works of each tool, but the

analysis does not provide a high-level overview of the traffic in real environments, study is not comparative and the scope is limited.

Traffic features that are significant for old TFN2k DoS tool traffic are examined by Dimitris et al. [11]. They put emphasis on the presence of SYN and URG flags, while simultaneously noticing that TTL and Window sizes provide almost no information. Conversely, our results indicate that the URG flag is not used by contemporary DoS tools anymore, probably because of its relative rarity, which would make the attack traffic easily identifiable [8].

Another study aimed at properties of DDoS bots has been performed by Edwards and Nazario [12]. The study focuses on families of DDoS botnet malware controlled predominantly from the Chinese IP space. An exhaustive summary of bot communication protocols is provided. Attacks supported by each bot are listed along with a high-level attack type taxonomy. However, from the perspective of attack traffic characteristics, only few unique properties of chosen bots are discussed and description of the traffic is overly general, without sufficient details to be used as an input in design of DDoS detection systems.

Slow DoS attacks form a class of stealthy attacks where attack hosts aim to allocate all available resources of the server for themselves, effectively denying the service for other hosts. Slow attacks require small bandwidth, are very stealthy and consist of fully established TCP connections. Cambiaso et al. classify slow attacks into four groups: pending requests DoS, long responses DoS, multi-layer DoS and mixed attacks [9]. Several representatives of slow DoS attacks have been discovered already, most notable being Slowloris [14] and Slow HTTP POST [10].

Basic properties of DDoS traffic are frequently listed with DDoS botnet analyses, such as the analysis of Dirt Jumper botnet [3] or Miner botnet [20]. Due to their primary focus on botnet properties, these studies only rarely provide sufficient technical details about the generated DDoS traffic. Although an overall description helps to understand the basic idea of an attack, missing technical details make it impossible to use this data as an input source for creation of new DDoS detection methods. Simultaneously, any estimate of effectiveness of existing DDoS detection methods against these attacks is difficult and unreliable.

3 Experiment

3.1 DoS tools selection

The full list of analyzed tools, versions, respective sources, supported attack types and tool identifiers used in later text is provided in Table 1. We are convinced that this list accurately represents the types of standalone DoS tools that can be currently encountered during real attacks.

Firstly, we selected a subset of existing standalone DoS tools based on their popularity and capabilities of attacking generic web servers. Arbor Networks Worldwide Infrastructure survey of 2014 notes that 78% of respondents have been targeted with various types of the HTTP GET flood, 55% with the HTTP POST flood, 43% with Slowloris attack, 38% with the LOIC DoS tool or its

variants, 27% with the Apache Killer tool, 23% with the HOIC DoS tool or its variants and 19% with the SIP Call-control flood. Among trailing attack types and tools are SlowPost, THC, nkiller, Hulk, RUDY and Recoil [4]. Secondly, we focused on tools that were used or allegedly used during publicized DDoS campaigns (OpUSA, OpIsrael, OpMyanmar). Thirdly, respected security companies often publish lists of DoS tools that are either popular or present a new step in development of DoS tools such as a by Curt Wilson of Arbor Networks [25].

We excluded any tools that are exclusive for a specific target application (e.g., Apache Killer) and tools that do not directly communicate with the target (e.g., DNS amplification attack tools). Lastly, we included several tools that are a popular choice on hacker forums (e.g., GoodBye, Janidos) or are created as open source in public software repositories (e.g., HTTP DoS Tool) or that take an extraordinary approach in causing a DoS effect (e.g., AnonymousDoS). Tools were selected in order to represent a full spectrum of existing types of TCP and HTTP DoS attacks.

Table 1. Selected tools and supported attacks.

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

Standalone tools are common inspirations for botnets. Even though most botnets rely on common volume-based attacks, such as generic HTTP GET flood, HTTP POST attack or TCP SYN attack, succesful new attacks are occasionally incorporated as well. For example, since the first release of the Slowloris HTTP client in June 2009, the Slowloris attack code has been included in advanced DDoS bots such as Mariposa, Skunkx or SpyEye. Similarly, a slow POST attack known from the Torshammer tool has been added to the Solar botnet and the R-U-D-Y attack to the Cyclone botnet. Although we usually observe a delay between the creation of a new proof-of-concept tool and full weaponization, support for new attacks is indeed added to botnets. Also, while standalone DoS tools are mostly free and public, bot binaries may be cracked and therefore unreliable, may be missing crucial components or may not be available at all. Obtaining reasonable botnet DoS traffic samples under pre-defined conditions and with non-interfering background traffic might be extremely difficult. Therefore, we believe this paper will also be beneficial for research on contemporary botnet capabilities.

3.2 Environment

The virtual environment was used in order to minimize the influence of real intermediate network on measurements. Also, virtual machine snapshots allow returning to a conjoint initial stable state. Therefore, any measurements on a restored snapshot are not affected by artifacts from previous measurements (e.g., keep-alive packets sent by either side). Our virtual environment was built on a single physical server with Core i7 CPU and 16 GB RAM.

We created a simple point-to-point virtual network between two virtual machines. The attacker VM had the Windows 7 operating system and the victim was the IIS 7.0 webserver on the Windows Server 2008 R2. Firewalls on both machines were configured to allow all incoming traffic from the shared network. Default settings for other subnets were kept. Except for DoS attack tools and the operating system itself, no other legitimate network traffic was knowingly produced. Tools were executed through the Administrator account with UAC enabled.

Our analysis was performed in a controlled virtual environment with no background traffic. Background traffic was omitted in order to gain as clear view of ideal attack conditions as possible. Applying legitimate background traffic would invalidate our results for scenarios with background traffic differing from the one we generated. Also, from the perspective of source end DoS attack detection, the impact of background traffic is diminishing. A reasonable assumption is that the source host is sending the attack traffic towards only one victim. Therefore, any source end DoS detection system can be considering traffic of each source IP and destination IP pair separately.

Background traffic can only alter time distribution of traffic (sections 4.1, 4.2 and 4.3) and only for highly susceptible, usually low-volume, tools. Internal properties of flows (e.g., HTTP request URI, flow packet count) cannot be altered by background traffic at all (sections 4.4, 4.5 and 4.6). Given the placement

of source end detectors directly on sending hosts or on first-mile routers, the complexity of intermediate networks or the number of attacking hosts is similarly irrelevant.

We used the CNN.com webpage from 11/19/2012 19:39 UTC, renamed to index.htm, as a testing target page. A popular existing webpage was selected in order to mimic real conditions under which DoS tools are launched. Saved webpage has 109 files and the total size is 3.3 MB including images.

3.3 Measurement

We review DoS attack tools from the viewpoint of source-end detection. While DoS mitigation systems are usually deployed on the victim side, the source-end side is more sound for the purpose of understanding the attack. Focusing on the source end enables deep and very precise understanding of inner works of tested tools without disturbances caused by an intermediate network.

Each tool has been tested with various configurations. The first configuration of each tool has been set with default tool settings if such exist. Configurations were chosen in order to test primarily settings that can alter the form of produced network traffic. We did not distinguish between successful and unsuccessful attacks. 60 and 300 second traffic samples were obtained for every tool configuration. The 300 second limit was chosen in order to track at least several iterations of even the most stealthy slow attacks. Oppositely, most DoS tools demonstrated their full traffic properties within first 15 seconds. Due to difficulties with packet recording at high packet transmission speeds, the measurement was focused on tool capabilities and traffic features, not performance comparison. Even though attack volume/performance is one of the cornerstones of victim end DDoS defense, its use in source end detection is problematic, mostly due to limited client bandwidth that is commonly saturated with legitimate network traffic.

DoS tools ran from a common initial state. Both outgoing and incoming network traffic was recorded with the dumpcap tool from the Wireshark suite directly at the attacker VM. We then performed our analyses offline on the collected PCAP files. Analyses 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 processed packet by packet, network flows were reconstructed and flow statistics were computed (e.g., simultaneous flow count, packets per flow). We define flow as 5-tuple: source IP address, destination IP address, source TCP/UDP port, destination TCP/UDP port, protocol.

Graphs on the following pages represent values of respective metrics each second of the first minute of the attack. Tables contain tool IDs of tools representing each category. When an ID is found in multiple categories, the actual behavior is dependent on chosen tool settings. Identifiers GB and LO represent all versions of the respective tool.

4 DoS traffic properties

4.1 Traffic burst behavior

Traditionally, DoS attacks were believed to produce an 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 of low-rate attacks is increasing as well [4].

Division of tools into classes by the packet rate shows that we can encounter both volume-rich tools and tools that produce hardly any traffic. Byte rate and packet rate values are especially interesting for tools that do not enable the attack intensity to be specified. For the vast majority of configurations the changes of byte rate value in time correspond to the changes of packet rate value. Note that the tool IDs are provided in Table 1 above.

In our set, a clear majority of tools employs an immediate full attack strength approach. Exceptions are LO and JL that may have an initiation period up to 10 seconds long (Fig. 4). 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. Packet rates of many DoS tools in our set exhibit a burst behavior. We divide observed burst types into four types. Attribution of tools to each burstiness type is provided in Table 2.

Full burstiness: The attack traffic is delivered only in bursts. Minimal or no traffic is exchanged between bursts (Fig. 3). Full burstiness is also very popular with slow attacks, often probably due to guidance by an internal clock.

Regular peaks: Produced network traffic is very stable except for regular repeating anomalies (Fig. 1).

One-time extreme: At one point of the tool run, often at the beginning of the attack, the traffic characteristics are significantly different from the rest (Fig. 2).

None: The tool does not produce traffic that has observable bursts in packet rate.

Although according to our knowledge the 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 a burst behavior, recognition of repeated occurrences of bursts and on similarity comparisons of these bursts.

Table 2. Traffic burstiness.

Full burstiness	HDT, HU, LO4, SL, SF
Regular peaks	BD, HO, LO, LC, UD
One-time extreme	AD, BAD, DS, GB, HDT, TH
None	ADR, FF, HF, JA, JL, LO, LC, SD, UD, XO

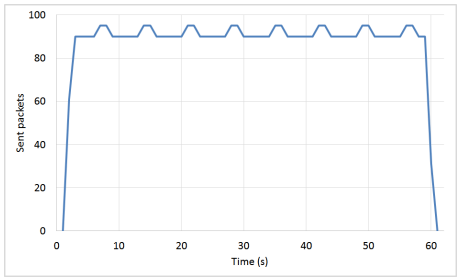

Fig. 1. BD packet count.

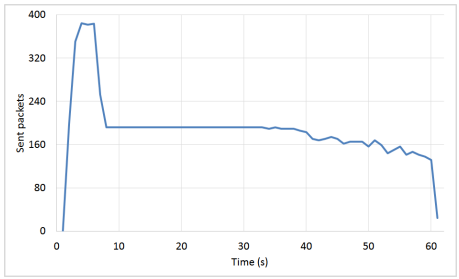

Fig. 2. GB5 packet count.

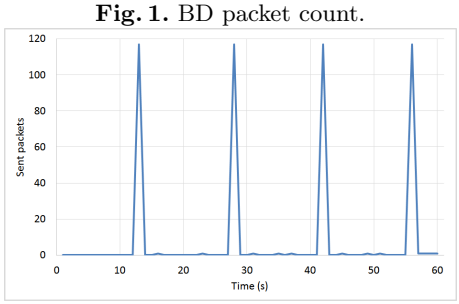

Fig. 3. SF packet count.

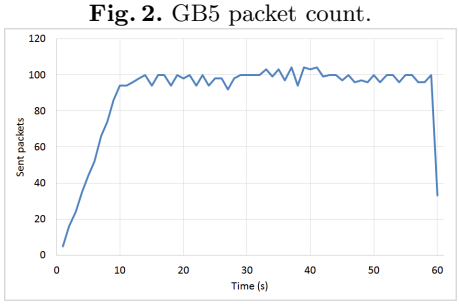

Fig. 4. LO2 packet count.

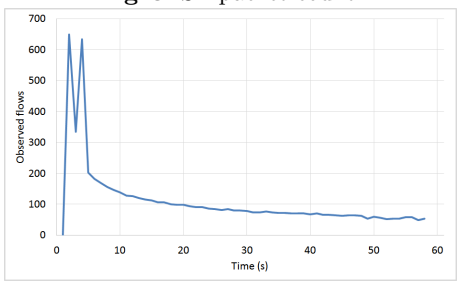

Fig. 5. DS flow count.

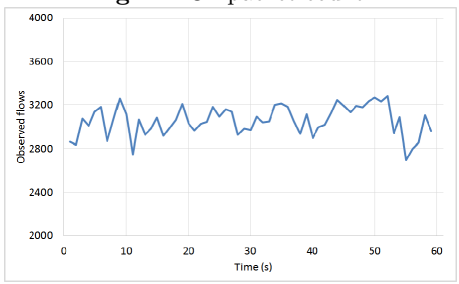

Fig. 6. FF flow count.

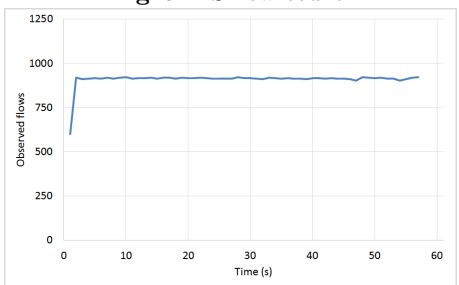

Fig. 7. JA flow count.

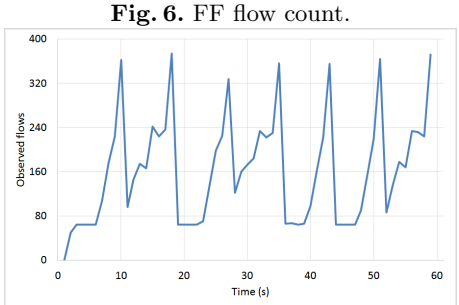

Fig. 8. LO2 flow count.

4.2 Flow count

Attacker establishing many connections towards a victim is one of the most common assumptions about DoS attacks. Reasoning behind this 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. JL, SD and XO 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 used to reach such limit (e.g., HU, FF), especially when executed several times in parallel. Oppositely, without regards to tool 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 stealthy for source-end intrusion detection systems that are based on flow count analysis.

Another important aspect of DoS traffic is the change in the number of flows in time. We classify configurations by the number of flows that were observed during initial 60 seconds of the attack. Four flow count patterns have been recognized. Attribution of tools to each flow count type is provided in Table 3.

Stability: Most tools exhibit only minor changes while the long term trend remains steady, e.g., FF or JA. While minor fluctuations can be expected (Fig. 6), the flow rate is extremely stable for most tools in an ideal closed environment (Fig. 7). This fact is emphasized in case of tools that require the operator to specify the flow rate prior to attack, be it directly as request per second ratio (e.g., BAD) or inderectly by the number of attack threads (e.g., LC, LO).

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

Decreasing count: Several tools such as DS (Fig. 5), GB and HDT tend to decrease the number of observable flows, even if the victim has not been made unavailable. The reason may be a poor design of the tool or inherent attack characteristics, especially in the case of slow attacks.

Increasing count: Although an attacker is expected to attempt using 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. A 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, except for a short initialization period at the beginning of the attack.

Table 3. Flow count change.

Stability	AD, ADR, BAD, BD, FF, HF, HO, HDT, JA, JL, LC, LO, SD, XO
Pulsing	HDT, HU, JL, LO, SF, SL, UD
Decreasing count	DS, GB, HDT

4.3 Flow parallelity

Results of flow parallelity measurements support our observations from the flow count measurement. The level of flow parallelity generally decreases with the decreasing flow count. Our observations show that a true parallelity is not 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 the flow sampling interval. With a decreasing interval, thresholds for DoS detection via simultaneous flows count should be lowered in order to maintain detection accuracy, as the count of seemingly simultaneous flows will decrease. In contrast, the count of truly simultaneous flows would remain constant. Attribution of tools to each flow parallelity type is provided in Table 4.

All simultaneous: Flows that are initiated in a 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. Attacks with spoofed source IP address has been inserted into this group (e.g., SF).

Mostly simultaneous: Flows are closed after a prolonged time, usually by the victim after the connection timeout runs out. Many flows are open at the same time. Flow duration usually 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, usually between several hundred milliseconds and several seconds.

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 a very short duration.

Table 4. Flow parallelity.

All simultaneous	AD, BAD, LC, SF, SL
Mostly simultaneous	DS, GB, HDT, HU, LO4, TH, UD
Long-term consecutive	LO
Mostly consecutive	ADR, BD, FF, HF, HO, JA, JL, UD, XO

4.4 HTTP requests per flow

Number of outgoing HTTP requests per flow for a single destination IP address can also be considered a decent detection metric. Normal non-DoS 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 HTTP attack tools produce many HTTP requests and their distribution between flows is often very straightforward, as can be seen in Table 5.

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 converges to 1 (Fig. 9). Special case are slow attacks based on slow sending of HTTP header. Although these attacks take a long time, each flow contains only one HTTP request message that is slowly constructed.

Multiple per flow: Established TCP flows can carry one or more separate HTTP requests and respective responses. Of the tested tools, none has exhibited 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. The ratio usually copies the packet rate curve (Fig. 10).

Table 5. HTTP requests per flow.

One per flow	ADR, FF, GB, HDT, HF, HO, HU, JA, JL, LO, SL, TH, UD
Infinite per flow	AD, BAD, LC

4.5 HTTP request URIs

We are convinced that the HTTP uniform resource identifier (URI) monitoring can be used as one of the most important metrics to verify the presence of an outgoing DoS attack in a given traffic sample. Observing repeated similar URIs either within one HTTP flow or within multiple flows with very similar characteristics is a strong indication of internal relationship and possible evidence of an outgoing DoS attack. Even though simply storing of all observed URIs is inefficient, performance problems can be solved, for example, with counting Bloom filters. Our analysis shows that from the perspective of source end DoS detection, most DoS tools target only a very limited number of URIs. Observing

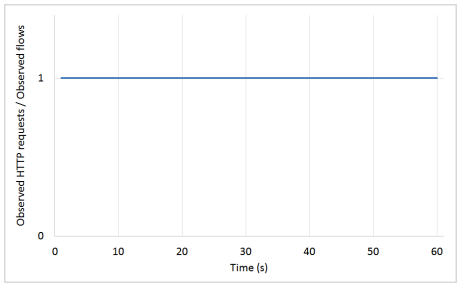

Fig. 9. FF HTTP requests per flow.

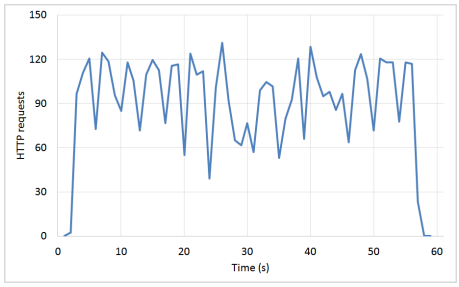

Fig. 10. LC HTTP requests per flow.

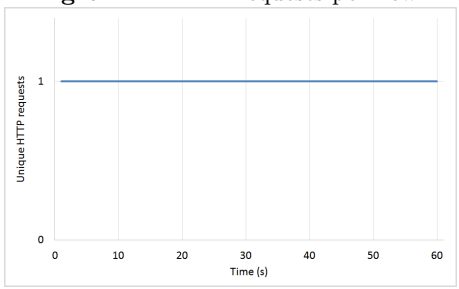

Fig. 11. JA unique HTTP request count.

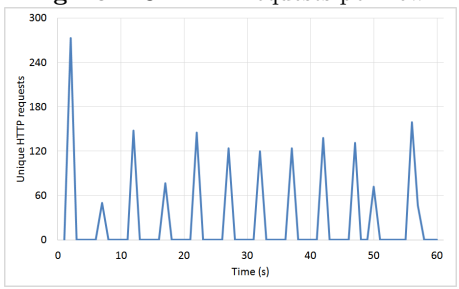

Fig. 12. HU unique HTTP request count.

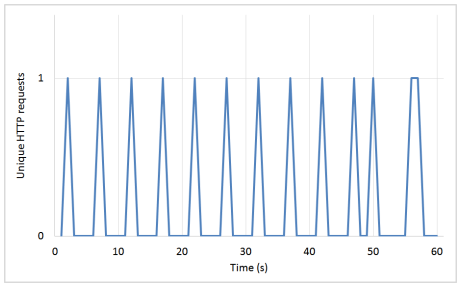

Fig. 13. HU requests without parameters.

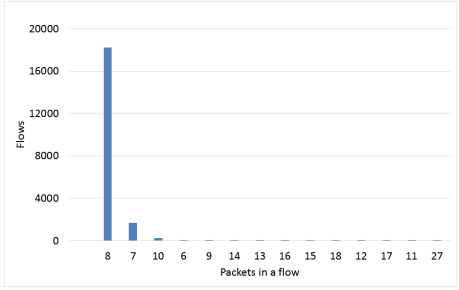

Fig. 14. JL flow packet count distribution.

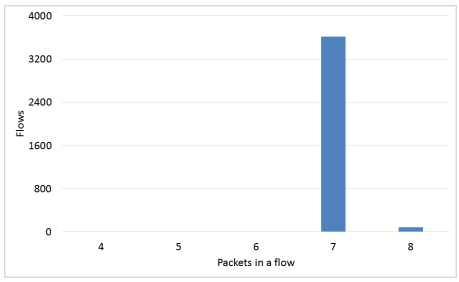

Fig. 15. UD flow packet count distribution.

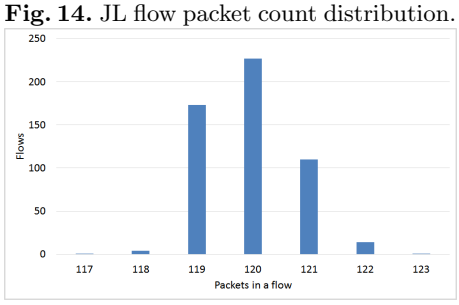

Fig. 16. HO flow packet count distribution.

such HTTP requests exceeding predefined threshold is a sufficient signal of an outgoing DoS attack in progress.

There are four basic techniques how DoS tools may process URIs. Attribution of tools to each of these techniques is provided in Table 6.

URI string set: The tool targets not just one URI on a selected victim server, but a predefined set of URIs. Using a set may slightly downgrade the attack efficiency, as not only the most resource demanding page is chosen to be the target, but also several others. If only one URI is accessed by the tool, the count of unique HTTP requests in time is equal to 1 (Fig. 11). It is also not uncommon for tools to not allow the change of the target URI at all (i.e., a basic value such as index.htm is employed).

Page crawling: The tool starts with an initial URI and gets more URIs by parsing the links in the HTTP response. None of the tools in our analysis employed the page crawling.

Parameter change: The base domain and file path remain constant, but full URI is made unique by adding unique parameter values. Unique parameter values render webpage caching servers between the attacker and the victim useless, therefore make the attack mitigation more difficult. Figures 12 and 13 show the difference between capturing full URIs and without parameters.

Random URI: URI may be fully randomly generated. That presents a challenge for attack detection and mitigation, but 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 so much computational power to retrieve the response.

It should be noted that URI frequency monitoring is unreliable metric when the webpage in question is limited to only a few pages. Therefore a combination with other metrics, such as suspicious User-Agent string monitoring, is necessary.

Oppositely, an overly large number of hard-coded URIs negatively impacts the attack power. Although accessing a large number or URIs makes intermediate caching less effective, the attacker also partially sacrifices his attack potential. Different URI requests require different volume of resources to process. With the suggested approach not only resource-demanding requests (e.g., DB searches, form submits), but also generic requests are sent towards the victim, lowering the attack effectiveness.

Table 6. HTTP request URIs.

URI string set	ADR, FF, GB, HDT, HF, HO, JA, JL, LO, LC, SL, TH, UD
Parameter change	AD, BAD, HU
Random URI	JL, UD

4.6 Flow packet count

Packet count is one of the most important properties of every flow. We believe that 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. TCP attack tools produce traffic where all closed flows have exactly the same packet count (disregarding possible TCP retransmissions). We believe that when applied to high flow (count) tools (e.g., SD, XO, JL), this metric can be both very precise and computationally efficient. Configurable precision can be devised from how many flow counts must be correctly predicted in order to consider those flows being part of a DoS attack.

The purpose of normal traffic is to transmit data between communication participants. In terms of TCP, 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 closed connection with only two or less packets sent by the flow initiator could have not transmitted any data. Oppositely, TCP-based attacks usually transfer few packets per flow, aiming to exploit TCP rather than transmit data.

As is shown in Table 7, majority of tools can produce homogenous traffic from the point of flow packet count. For example, HTTP POST flooding attack by UD generates flows with vast majority having 7 packets after closure (Fig. 15). Oppositely, HO is one of the tools whose traffic does not provide any recognizable flow packet count (Fig. 16). Excluded were tools whose connections were never closed during the first minute of the attack (AD, BAD, HF).

Table 7. Flow packet count distribution.

All flows the same packet count	ADR, BD, DS, FF, GB, HDT, JA, JL, LC, LO1, SD, SF, SL, UD, XO
Minimal differences	LO, SD, UD
Significant differences	HDT, HO, HU, JL, LO, SD, TH, UD

5 Discussion

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 can rarely launch several attacks simultaneously. Majority of tools does not require root privileges and therefore can be executed on computers at work, school or internet cafe. Basic operations with DoS tools do not require advanced knowledge about the victim or the type of attack. Most tools allow for 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.

5.1 Traffic features and aggregation

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

Although almost every traffic feature that we measured yielded some results, none proved to be sufficient on its own for the detection of DoS attacks in the source-end network. Every feature can detect only a subset of existing DoS attacks. Standalone features suffer from false positives, but more importantly, have an inherent limit of false negatives rate. Different classes of DoS attacks have different properties and none of the traffic features could be applied to all. Employing just one input feature for DoS detection results in an inability to detect many classes of attacks. Still prevalent assumptions about DoS traffic regarding traffic volume, flow composition or protocol compliance are obsolete and cannot be applied to DoS attacks in general, rather only to small DoS attack subclasses.

Therefore we believe that an aggregation of multiple features is necessary to be used for a general detection. We support the approach taken, for example, by [18, 22] that collect multiple feature values and subsequently compute their aggregate importance.

Serious consideration must be given not only to the computational efficiency of the detection, but also to an efficient collection of input values. Features included in the 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 (e.g., slow attacks cannot be detected with volume-based detection metrics). 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. [24] is promising.

Traditional metrics such as a high bitrate and a high packetrate are by themselves not reliable options for the 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.

5.2 Repeating patterns

Most important observation of this work is that standalone DoS attack tool *traffic comprises of repeating operations*. Every attack has a basic construction unit that 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.

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 increased accuracy.

Recognition of repeating patterns opens a new area of detecting outgoing DoS attacks at the source end. This novel approach presents challenges how to identify 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. For illustration, we provide example scenarios of this new approach to DoS detection.
Example 1 – BD. 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 flag composition. The flow is always established via a correct TCP 3-way handshake (3WH) and terminated by the attacker with the TCP FIN segment, which is followed by the TCP RST segment from the victim. TCP segments don't carry any payload. All of the TCP header option 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 take between 0.1 and 0.12 seconds. None of the packets has the time to live (TTL) value altered or is using a spoofed IP address.
Example 2 – AD. The 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 packets are not fragmented. Header of every HTTP request contains the same fields with the same values. The referer field is always missing. The 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.

5.3 Evasion techniques

Most standalone DoS 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 traffic properties randomization. Randomization is usually configurable only for the packet fields chosen by a tool creator. Therefore, the effect of randomization can frequently be negated if multiple input features/header fields are analyzed in conjunction.

A similar technique can be observed at URI randomization. Adding random parameters such as timestamps in Unix format (e.g., AD, BAD), random parameter values (e.g., LO) or even random parameters (e.g., HU) can be used both to evade simple DoS detection systems and to circumvent content caching between the attacker and the victim.

Randomization is a powerful weapon for attacker, but it is not almighty. Excessive or impromper randomization can be detrimental for the attacker by making his traffic more visible. For example, as noted above, attack tools commonly randomize User-Agent string of HTTP request header [8]. While this is reasonable for victim end detection systems, because User-Agent string cannot be used to classify attack traffic, it significantly raises suspicion of source end detection systems. Even more importantly, many attack traffic features cannot be randomized without severe degradation of attack performance (e.g., flow packet count for TCP SYN attack).

Employing evasion techniques for the network or transport ISO/OSI layer was rare. SF was the only tool in our set that employed IP spoofing. We assume that IP spoofing is not popular with these tools, because it enforces the use of only the most primitive attacks, such as SYN flood.

5.4 Future work

We perceive this work is a necessary prerequisite to our follow-up research on DDoS attack detection. Creation of this work was compelled by the lack of up-to-date traffic samples and sparse reliable information on traffic properties of contemporary DoS attacks. We are convinced that the persistent trend when DDoS detection methods are evaluated against well-understood, but ruefully outdated attack descriptions/attack tools, is inherently flawed. Even though the exact properties of each attack that we analyzed, varied, we have discovered a set patterns recurring among DoS tools from different creators. We believe these patterns will prevail for a longer time than simple attack signatures.

The key revelation is the presence of repeating operations in all analyzed DoS attack traffic. Therefore, we propose a new research area for the detection of DoS attacks at the source end that is based on repeated attack pattern recognition. We discuss overall DoS tool properties and employed detection evasion techniques. Since attack features are not mutually comparable due to inherent detection efficiency limitations, it is crucial that researchers include their DoS attack traffic assumptions and any possible evasion techniques in every research output/publication that is dealing with DoS attack detection.

Further research will be required to analyze why these patterns are prevalent. Possibly, this is because of focus of tools' creators on victim end defense. Even though thorough per-packet randomization is possible, it results in an increased load of the source host, brings implementation issues and most notably, it decreases an overall performance of the tool. We frequently encountered per-flow randomization or even randomization taking place only once when the tool was run. From the victim end perspective, this level of traffic randomization is usually sufficient, due to distributed nature of attacks. However, this behavior can be exploited by source end DDoS detection solutions, because it increases attack visibility near the source host.

The impact of randomization on detection metrics depending on the placement of detection sensors is another interesting area of further research.

Volumetric DoS attack traffic consists of repeated operations with minimal differences. We intend to explore the possibility of creating a grammar that would allow us to describe the attack traffic from the source host perspective in an easily understandable, yet precise notion. The grammar will give researchers a good understanding of what operations are common and how the attack traffic changes between different versions of one tool.

6 Conclusions

This paper encourages and supports the evaluation of new source end DDoS detection systems against contemporary DoS attacks. We have analyzed state-of-the-art standalone DoS tools that have been observed in real DoS attacks. We provided detailed properties of attack traffic and emphasized notable traffic anomalies from the perspective of source end DoS detection. Attack traffic is classified by each input feature and overall characteristics of each class are listed. Attack traffic traces are suitable for evaluation of DoS detection and mitigation systems through overlay methodology. More details about our experiments can be found in our technical report [8]. The traces are available for download at [1].

References

1. DDoS-Vault project. 2015. https://github.com/crocs-muni/ddos-vault/wiki.
2. 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. Foundation of Computer Science, New York, USA.
3. Marquez Andrade and Natalija Vlajic. Dirt Jumper: A New and Fast Evolving Botnet-for-DDoS. *International Journal of Intelligent Computing Research*, 3(3), December 2012.
4. Arbor Networks. Worldwide Infrastructure Security Report. volume IX, 2014.
5. Adam J. Aviv and Andreas Haeberlen. Challenges in Experimenting with Botnet Detection Systems. In *4th USENIX Workshop on Cyber Security Experimentation and Test (CSET '11)*, 2011.
6. Michael R. Bartolacci, Larry J. LeBlanc, and Ashley Podhradsky. Personal Denial Of Service (PDOS) Attacks: A Discussion and Exploration of a New Category of Cyber Crime. *Journal of Digital Forensics, Security and Law*, 9(1):19–36, 2014.
7. 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*, 57(4), 2013.
8. Vit Bukac. Traffic characteristics of common DoS tools. Masaryk University, April 2014. Technical report FIMU-RS-2014-02.
9. Enrico Cambiaso, Gianluca Papaleo, and Maurizio Aiello. Taxonomy of Slow DoS Attacks to Web Applications. In *Recent Trends in Computer Networks and Distributed Systems Security*, volume 335 of *Communications in Computer and Information Science*, pages 195–204. Springer Berlin Heidelberg, 2012.
10. Wong Onn Chee and Tom Brennan. H.....t.....t....p....p....o....s....t. In *OWASP AppSec DC 2010*. The OWASP Foundation, 2010.

11. Gavrilis Dimitris, Tsoulos Ioannis, and Dermatas Evangelos. Feature Selection for Robust Detection of Distributed Denial-of-Service Attacks Using Genetic Algorithms. In *Methods and Applications of Artificial Intelligence*, volume 3025 of *LNCS*, pages 276–281. Springer Berlin Heidelberg, 2004.
12. Jeff Edwards and Jose Nazario. A survey of contemporary Chinese DDoS malware. In *Proceedings of the 21st Virus Bulletin International Conference*, 2011.
13. Vegard Engen, Jonathan Vincent, and Keith Phalp. Exploring Discrepancies in Findings Obtained with the KDD Cup '99 Data Set. *Intelligent Data Analysis*, 15(2):251–276, April 2011.
14. Robert Hansen. Slowloris HTTP DoS. 2009. ha.ckers.org/slowloris/ (22/10/2014).
15. Peyman Kabiri and Gholam Reza Zargar. Category-based selection of effective parameters for intrusion detection. *International Journal of Computer Science and Network Security (IJCSNS)*, 9(9):181–188, 2009.
16. Jelena Mirkovic, Gregory Prier, and Peter Reiher. Source-End DDoS Defense. In *Second IEEE International Symposium on Network Computing and Applications, 2003. NCA 2003*, pages 171–178, 2003.
17. 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), March 2005.
18. 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.
19. Iosif-Viorel Onut and Ali A. Ghorbani. Features vs. Attacks: A Comprehensive Feature Selection Model for Network Based Intrusion Detection Systems. In *Information Security*, volume 4779 of *LNCS*, pages 19–36. Springer Heidelberg, 2007.
20. Daniel Plohmann and Elmar Gerhards-Padilla. Case study of the Miner Botnet. In *4th International Conference on Cyber Conflict (CYCON)*. IEEE, 2012.
21. Ali Shiravi, Hadi Shiravi, Mahbod Tavallaee, and Ali A. Ghorbani. Toward developing a systematic approach to generate benchmark datasets for intrusion detection. *Computers & Security*, 31(3), 2012.
22. 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.
23. Vrizlynn L. Thing, Morris Sloman, and Naranker Dulay. A Survey of Bots Used for Distributed Denial of Service Attacks. In *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.
24. Fei Wang, Hailong Wang, Xiaofeng Wang, and Jinshu Su. A new multistage approach to detect subtle DDoS attacks. *Mathematical and Computer Modelling*, 55(1–2):198 – 213, 2012.
25. Curt Wilson. Attack of the Shuriken: Many Hands, Many Weapons. 2012. Webpage, http://asert.arbornetworks.com/ddos-tools/ (29/05/2015).
26. 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):1005 – 1012, 2013.
27. Lifang Zi, J. Yearwood, and Xin-Wen Wu. Adaptive Clustering with Feature Ranking for DDoS Attacks Detection. In *4th International Conference on Network and System Security (NSS)*, pages 281–286, Sept 2010.