Small scale denial of service attacks

Dissertation thesis

Vít Bukač

Brno, 2015
Declaration

Hereby I declare that this paper is my original authorial work, which I have worked out by my own. All sources, references and literature used or excerpted during elaboration of this work are properly cited and listed in complete reference to the due source.

Vít Bukač

Advisor: prof. RNDr. Václav Matyáš, M.Sc., Ph.D.
Acknowledgements

I am very grateful to my supervisor Vashek Matyáš for his guidance and support. I especially appreciate that he always found time for face-to-face consultations and that he was always willing to provide me with his valuable feedback.

I thank the members of the research teams that I worked with: Lukáš Němec, Radim Oštádal, Zdeněk Říha, Vlasta Šťavová, and Petr Švenda. It was an honor and a privilege to work with them and learn both from them and with them.

My thanks also go to all other colleagues from the Centre for Research on Cryptography and Security, most notably to Marek Kumpošt, Martin Stehlík and Marek Sýs. They helped create a friendly, inspirational and creative environment where I happily worked. On top of that, we all had a lot of fun, let’s not forget that either.

My current and former coworkers Václav Lorenc, Peter Raymond, Mike Schaefer, and Pavel Tuček were always very tolerant and supportive of my research. The experience and the knowledge that I gained from them shaped my ideas and opinions in subtle, yet important ways.

Special thanks go to my parents. Without their support, this thesis would have never been created. They helped me rise whenever I felt down and they kept me on the ground whenever I overextended.

Last thanks go to my Granny. She never even considered the possibility that I could fail.
Abstract

A denial of service (DoS) attack is the most common attack against the data availability on internet. It is an attempt to make a computer resource unavailable to its intended users. A distributed denial of service (DDoS) attack occurs when multiple attack sources collaborate to achieve this goal.

Flooding attacks are the most prominent class of (D)DoS attacks. During a flooding DoS attack, attacking hosts are sending a huge volume of unsolicited network traffic to the victim, which results in the victim being overwhelmed. While most attention of both academia and industry is given to the analysis of large-scale volumetric DoS attacks, statistics have shown that more than 80% of DoS attacks worldwide have bandwidth less than 1 Gbps. These low power attacks can be executed by botnets, standalone DoS tools or DDoS-for-hire services (also DDoS as a service, DDoSaaS).

This dissertation provides a comprehensive analysis of the threat posed by standalone DoS tools and widely available DDoSaaSs. Also, a framework for DoS attacks augmentation is presented.

In case of botnets, outgoing DoS attacks recognition can be used for behavioral detection of malware-infected hosts. We explore both the founding ideas and the state-of-the-art research on host-based intrusion detection systems (HIDSs). Separate sections are devoted to the protection against tampering and to the HIDS evasion techniques that are employed by attackers. Existing research trends are highlighted, and possible future directions are suggested.

Research on denial of service attack detection is complicated due to scarcity of reliable, widely available and representative contemporary input data. Labelled DoS attack datasets are especially rare. Therefore, 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, which were used in well-known attack campaigns. We provide a classification of different attack traffic features, including utilized evasion techniques and encountered traffic anomalies.

The maturity of DoS attack provision market demonstrated itself in the appearance of DDoS-for-hire services. These services are intended for a wide audience as they allow executing DoS attacks without any technical knowledge on the side of customer. We analyze the threat of DDoS-for-hire services to low and medium power cloud-based servers or home users. We aim to investigate popularity and availability of such services, their payment models, subscription pricing, complexity of the generated attack traffic and performance.
A progress of every attack against a computer system can be described by a well-known and widely used kill chain methodology. The kill chain specifies that every attack against a computer system goes through several clearly identifiable phases. The phases have a fixed order and each phase can take place only if all previous phases were successfully completed. We analyze the kill chain of DDoS-for-hire services and compare it to kill chain of advanced persistent threats. We discuss specifics of kill chain when a DoS attack is carried out by two distinct parties, i.e., service owner and customer.

Flooding DoS attacks frequently have unexpected impacts. In order to design proper detection and mitigation systems, it is crucial to discover existing limits and potential new variants of contemporary attack types. However, both the development of new DoS attacks and manual assessment of DoS attacks resiliency are tedious processes. We created a new framework for rapid evolution of denial-of-service attacks by genetic algorithms. The framework can be used to identify limits of current DoS attacks as well as to evaluate the resiliency of a particular software application against a selected DoS attack class. The framework is virtualization-based and provides a high degree of automation.
Keywords

denial of service, DoS, DDoS, DDoSaas, traffic features, dataset, DDoS-for-hire, intrusion detection, network security
# Contents

1 Introduction ................................................................. 1  
1.1 Denial of Service attacks ............................................. 1  
1.1.1 DoS attacks taxonomy ............................................. 1  
1.1.2 Testing environment ............................................. 2  
1.1.3 Detector placement ............................................. 3  
1.2 Problem statement ................................................... 4  
1.3 Contributions ....................................................... 5  
1.4 Structure of the thesis ............................................... 7  
2 Host-based intrusion detection systems ............................. 9  
2.1 HIDS overview ........................................................ 10  
2.2 Standalone network traffic monitoring HIDS ..................... 12  
2.3 Collaborative HIDS .................................................. 15  
2.4 Cloud-based HIDS .................................................... 17  
2.5 HIDS security ......................................................... 18  
2.6 Evasion techniques .................................................. 20  
2.7 Summary ............................................................... 22  
3 Traffic features of common standalone DoS attack tools ........ 24  
3.1 Related work .......................................................... 26  
3.1.1 (D)DoS attack traffic features .................................. 29  
3.2 Experiment ............................................................. 31  
3.2.1 Environment ........................................................ 31  
3.2.2 DoS tools selection .............................................. 31  
3.2.3 Measurement ..................................................... 32  
3.3 Analyzed traffic features ............................................ 34  
3.3.1 Traffic burst behavior ........................................... 34  
3.3.2 Flow count ....................................................... 37  
3.3.3 Flow parallelity ................................................... 38  
3.3.4 IP spoofing ........................................................ 41  
3.3.5 Average packet size ............................................. 42  
3.3.6 HTTP requests per flow ....................................... 43  
3.3.7 HTTP request URIs ............................................. 44  
3.3.8 HTTP User-Agent ................................................ 46  
3.3.9 Flow packet count ............................................... 49  
3.4 Discussion .............................................................. 51  
3.4.1 Traffic features and aggregation ............................... 51
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>Conclusions and future work</td>
<td>94</td>
</tr>
<tr>
<td>7.1</td>
<td>Future work</td>
<td>95</td>
</tr>
<tr>
<td>A</td>
<td>Bibliography</td>
<td>97</td>
</tr>
<tr>
<td>B</td>
<td>Standalone DoS tools traffic features</td>
<td>112</td>
</tr>
<tr>
<td>B.1</td>
<td>List of author’s publications</td>
<td>114</td>
</tr>
<tr>
<td>B.1.1</td>
<td>Book chapter</td>
<td>114</td>
</tr>
<tr>
<td>B.2</td>
<td>International conferences and workshops</td>
<td>114</td>
</tr>
<tr>
<td>B.3</td>
<td>National conferences and journals</td>
<td>115</td>
</tr>
<tr>
<td>B.4</td>
<td>Technical report</td>
<td>115</td>
</tr>
</tbody>
</table>
1 Introduction

We live in an information society. We are used to obtain required information in a timely fashion in order to make knowledgeable and responsible decisions. Inability to collect information may lead to financial losses, cultural conflits or even human injuries. In this thesis, we focus on denial of service attacks, a common class of attacks against the availability of resources in computer networks.

1.1 Denial of Service attacks

A denial of service attack (DoS attack) is an attempt to make a computer resource unavailable to its intended users. A distributed denial of service attack (DDoS attack) occurs when multiple attack sources collaborate to achieve this goal. Most DDoS attacks employ the IP spoofing to conceal identities of attacking machines [BBH09].

1.1.1 DoS attacks taxonomy

DoS attacks can be classified by many criterias. The following taxonomy is loosely derived from the work of Mirkovic and Reiher [MR04].

- **Vulnerability (Semantic):** Vulnerability attacks exploit a specific feature or an implementation bug of some protocol or of an application installed at the victim in order to consume excess amounts of its resources.

- **Flooding (Brute-force):** Flooding attacks are performed by initiating a vast amount of seemingly legitimate transactions or by sending a huge volume of unsolicited and unexpected traffic. Exhausted resources may include available computational time, available operating memory, free space in various buffers and status tables or available incoming bandwidth. A non-exhaustive list of existing flooding DoS attacks and their classification by key protocol into HTTP-based, TCP-based and UDP-based attacks is provided in Table 5.1.

Flooding attacks can be then treated in two groups – attacks at the network layer and at the application layer:

**Network layer:** Properties of network and transport layer protocols are exploited.
1. Introduction

- **Low rate DoS attack (shrew attack):** Low rate DoS attacks attempt to deny bandwidth to TCP flows while sending packets at a sufficiently low average rate to evade the detection [KK06].

- **Pulsing attacks:** Compromised hosts send out short bursts of attack packets to the victim instead of generating a constant packet flood during pulsing attacks [LC05].

- **Reflection:** A reflector is any IP host that will return a packet if sent a packet. Attackers orchestrate the hosts under their control to send the spoofed traffic purportedly coming from the victim to reflectors. The result is that the flood at the victim arrives from a significantly higher number of sources, an exceedingly diffuse flood likely clogging every single path to the victim from the rest of the internet [Pax01].

- **Amplification:** An amplification attack is a type of the reflection attack in which reflectors’ responses are larger than queries. Therefore the volume of the attack traffic from the source to the victim is multiplied [VE06].

**Application layer:** Application layer protocols are exploited. Connection establishments on the network layer and the transport layer are required. Application-layer requests originating from compromised hosts may be difficult to distinguish from those generated by legitimate users [XY09].

1.1.2 Testing environment

Numerous ways of the DoS attack experimenting have been presented in the literature. Most notable approaches are listed.

- **Testing in a real environment:** Although results can be very precise, due to disruptive effects of DoS attacks the testing in real environments is rarely performed. Also results from different networks are not comparable and tests on a global scale often require an agreement of multiple parties.

- **Testing in a testbed environment:** Testbed environments provide a sufficient fidelity, but experiments have shown results are not comparable between different testbeds [CFS06].

- **Traffic simulation:** Simulations enable an easy and fast creation of topologies, however, scaling is an ongoing issue. Simulated nodes possess an infinite CPU
and bus capacity, which can interfere with results especially when complex DDoS attacks are simulated [CFS06].

- **Packet traces replay**: Results are comparable and tests can be repeated. However, well-known documented up-to-date packet traces are sparse. Alternatively a proprietary set of packet traces can be generated using the overlay methodology [AH11], but it is hard to determine whether the background traffic is attack-free and sharing of proprietary traces is complicated because of privacy concerns.

### 1.1.3 Detector placement

The deployment of the DoS attack detection system at the victim end is historically the most common. The victim has the highest motivation to mitigate attacks and the attack impact is most easily observed. However, victim end countermeasures cannot help in case of a truly damaging attack that overwhelms access links to the victim. Also, existing countermeasures (e.g., rate limiting, traffic filtering) are frequently performed on a per-subnet basis. Therefore, legitimate users from the attacker’s subnet may suffer the denial of service effect as a result of the defense’s collateral damage. Identification of attack sources is nearly impossible without the cooperation of intermediate networks.

An intermediate network is any network, which is traversed by the attack traffic en route from the source to the victim. Detection and defense at the intermediate network lower the global congestion and to a certain level offer a capability to identify source attack nodes. However, a deployment incentive is an open question since intermediate networks usually benefit little from the ability to stop ongoing attacks. Also, network devices on high rate backbone links reserve a vast majority of their resources to routing and switching purposes.

Finally, outgoing DoS attacks can be detected directly at the source hosts or at first mile routers. Source end detectors can prevent congestion, allow for an easy identification of attacking hosts and can apply complex detection algorithms. Detection algorithms may be more resource-demanding than in case of the victim detectors, because the source end detectors analyze less data and the combined computational power of all detectors is higher than the computational power of a limited set of victim detectors. However, source end detectors observe only a portion of attack traffic and they do not have access to the internal victim state, which makes the recognition of attack traffic difficult. The problem of deployment incentive is similar as with detectors for intermediate networks.
1. Introduction

1.2 Problem statement

This thesis deals mainly with the following problems:

1. **Contemporary DoS attacks are not sufficiently examined by the research community.** DoS attacks in the wild are too different from attack samples that are used for research purposes. DoS attacks are quickly evolving and constantly changing, while the academic community is slow to react. 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 attacks actively used in real environment contains advanced attacks such as DNS amplification, slow attacks and application-layer attacks. Conversely, academic concepts are notoriously evaluated with obsolete and in practice already extinct attack traffic, such as those produced by tools like TFN, TFN2k, Shaft, Trinoo, Knight, mstream and Stacheldraht that all date back to year 2000. Such attack traffic is no longer seen in the wild. Therefore, it cannot be used to validate the effectiveness of DoS mitigation systems against contemporary attacks.

2. **The market with DDoS attacks has matured into a DDoS-as-a-Service model, which has not been deeply examined.** Reasonably strong DoS attacks are now available to be executed by technically unsavvy users for a very low price. The impact of this evolution on DoS attacks proliferation and hazards must be analyzed.

3. **Contemporary labelled DoS attack datasets are sparse.** Datasets with labelled attack traffic and benign traffic are necessary for evaluation of false positive, true positive, false negative and true negative values of newly proposed detection and mitigation systems. DARPA 1999 and KDD 1999 datasets are still being used, despite being more than 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 types variability.

4. **There is no automated framework for evaluating the resistance of network applications to DoS attacks.** End applications are frequently key weakpoints during DoS attacks. By overwhelming the application itself, an attacker can successfully cause a denial of service effect, even if the hosting server is still available. A framework for an automated testing of advanced DoS attacks against network applications is needed to identify vulnerabilities in application design and/or implementation.
During our research we also touched upon the question of experiment results comparability between a virtual environment and a physical environment. Chertov et al. already presented such problem when experimenting with a large traffic volume that pushes physical devices to their limit [CFS06]. However, we note that the differences may be observed even in very simple experiments, where researchers intuitively expect comparable results. We briefly discuss this problem in Chapter 6.5.

1.3 Contributions

In this thesis we address the issue of small scale denial of service attacks.

1. **We perform a thorough analysis of DoS attack traffic that is generated by standalone DoS tools.** Standalone DoS tools are good representatives of contemporary DoS attacks seen in the wild. We collect traffic samples from these tools in a sandbox virtual environment without background noise traffic. We provide a classification of different attack traffic features, including utilized evasion techniques and encountered traffic anomalies. We also propose a new research direction for the detection of DoS attacks at the source end, based on repeated attack patterns recognition. The full results were published as a technical report in [Buk14]. Selected parts were also published as a research paper in [BM15].

2. **We map the full ecosystem of DDoS-for-hire services.** DDoS-for-hire services are a next evolutionary step in DoS attack monetization. We analyzed multiple data sources related to DDoS-for-hire services, such as the attack traffic generated by them, their leaked source codes, content of their leaked databases or statistical information about their webpages. The aggregate data was used to investigate popularity and availability of DDoS-for-hire services, their payment models, subscription pricing, complexity of the generated attack traffic and attack performance. The kill chain of a typical attack is presented. The chapter findings were published in [BSN15]. The kill chain section is based on papers [BRN15] and [BLM14].

3. **We create a new contemporary labelled DoS attack traces dataset and share it for use by other researchers.** The dataset is composed of DoS attack traffic traces that were collected during the research on both standalone DoS tools and DDoS-for-hire services. Associated data includes standalone DoS tools binaries and scripts that were used for attack traffic analysis.

4. **We design a novel framework that can be used for automated evaluation of DoS attacks effectiveness, as well as for vulnerability assessment of existing network applica-
1. Introduction

We present the framework modular architecture and workflow. The framework was employed in two scenarios: searching for a new potent HTTP GET flooding attack variant by changing the order and values of HTTP header fields, and vulnerability assessment of existing web servers against a new proposed SSL slow handshake DoS attack. Preliminary results from this chapter were published in [BOS+15].

5. We survey the state-of-the-art research on network monitoring host-based intrusion detection systems. Existing research trends are highlighted, and possible future directions are suggested. The presented findings were published as parts of extensive HIDS surveys in [BTD12] and in [BM14].

Our contributions are supported by the following list of the author publications that were peer-reviewed and published in the proceedings of international conferences, workshops and monographies:


1. Introduction


1.4 Structure of the thesis

The structure of the thesis is as follows:

- Chapter 2 surveys both the founding ideas and the state-of-the-art research on host-based intrusion detection systems (HIDSs) that employ network traffic monitoring. HIDSs are divided into standalone, collaborative and cloud-based categories. Seminal research findings and ideas are presented and supplied with comments. Separate sections are devoted to the protection against tampering and to the HIDS evasion techniques that are employed by attackers.

- In Chapter 3 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 and discuss the traffic features that are promising for the detection of DoS attacks at the source end. We propose a novel research area based on recognition of repeating patterns in DoS attack traffic.

- Chapter 4 analyzes the threat of DDoS-for-hire services to low and medium power cloud-based servers or home users. We provide characteristics of attack properties of attacks generated by these services. We also present findings that were discovered in leaked databases of these services, as well as their leaked representative source codes.

- Chapter 5 provides a detailed information about the DDoS-Vault dataset. We provide technical details about included traces and tools.

- Chapter 6 presents a novel framework for a rapid evolution of denial-of-service attacks by genetic algorithms. We provide technical overview of the framework.
We present the results of two testing scenarios, the HTTP GET requests flooding attack enhancement scenario and SSL slow attacks resistance scenario. We also briefly discuss challenges of experimenting in virtual and physical environments.

The last chapter concludes the thesis and presents directions for future work. It is followed by a list of references.
2 Host-based intrusion detection systems

An intrusion detection system (IDS) is a device or a software application that monitors events occurring at a host and/or in a network, identifies malicious activities or policy violations and produces reports for further analyses.

Numerous intrusion detection systems taxonomies have been proposed in the past. Lazarevic et al. designate five criteria to be used to classify IDSs: information source, analysis strategy, time aspects, architecture and response [LKS05]. A more detailed taxonomy that also includes classification by alerts and adds further subcategories is outlined by Sabahi and Movaghar [SM08a].

We recognize two types of IDSs based on the analysis strategy: signature-based IDSs (i.e., misuse-based) and anomaly-based IDSs:

- **Signature-based** systems compare observed events with known patterns of malicious activities. They can effectively detect existing threats and, provided signatures were constructed efficiently, have a low false positives rate.

- **Anomaly-based** systems compare the actual behavior with pre-created profiles of normal behavior and observe any deviations. Anomaly-based systems can detect previously unknown attacks, but the identification of these attacks can be very vague. A training period without any ongoing intrusions is often required to construct normal profiles. Some anomaly-based systems are adaptive, therefore they can update their normal profiles in a reaction to the changing properties of the observed system.

Commercially-available solutions can incorporate both detection approaches. For example, common antivirus software is predominantly signature-based, but can also include a heuristic analysis that falls into the anomaly-based analysis strategy. Surveys of various intrusion detection analysis strategies as well as comparison of their strengths and weaknesses were provided by Murali and Rao [MR05b] and by Chandola et al. [CBK09].

Scarfone and Mell distinguish four categories of IDSs based on information source, i.e., the type of events they monitor and the ways in which they are deployed: network-based IDS (NIDS), host-based IDS (HIDS), wireless IDS and a network behavior analysis system [SM07]. Sometimes we can also encounter the term “hybrid IDS” that denotes a system that combines two or more IDS categories.

In this chapter we focus strictly on host-based intrusion detection systems. The content of this chapter is based on our survey of HIDSs that was published in [BTD12] and later expanded in [BM14].
2. Host-based intrusion detection systems

2.1 HIDS overview

A host-based IDS is a program that monitors characteristics of a single host and events occurring within this host to identify and stop suspicious activity. In practice, the HIDS usually monitors the behavior of running processes, enforces the integrity of critical system files and registry keys, performs complex log analyses, and monitors the host network traffic.

A HIDS survey was published by Vigna and Kruegel [VK05]. Compared to them, our work explores the research effort of approximately the last five years, most notably the widespread of virtualization technologies, the emergence of cloud-based intrusion detection solutions and the whole direction of HIDSs research in situation when network-based IDSs are generally more preferred both by the academic community and by the knowledgeable security specialists.

HIDSs should be regarded as an important complementary part of the overall intrusion detection architecture. They supply the following benefits:

- **Semantic information**: HIDSs are close to the protected resources, in the best position to observe the behavior of the operating system or its applications. From the perspective of network attacks, the source host holds important information regarding the identification of the type of the attack and of the attack originator (e.g., name, path and a cryptographic hash of the program which produced the traffic, login of the user currently working with the computer, packets samples, exact timestamps).

- **Network traffic interpretation**: If we can solve the ground truth problem, the network traffic data is more precise than in case of NIDSs. Host-based IDSs interpret packets in the same way as the communicating applications. The majority of network traffic IDS evasion techniques (e.g., packet payload obfuscation, interpretation discrepancies) are not useable against HIDSs [PN98]. Moreover, a HIDS can inspect traffic which was end-to-end encrypted on the network layer or the transport layer (e.g., by TLS or IPsec), because it has access to the plaintext at the application layer.

- **Best effort**: We often encounter situations in which other IDS categories are not useable (e.g., notebooks connecting through an untrusted Wi-Fi, 4G mobile devices). In some cases even if a malicious activity was detected, network administrators may not be willing to share that information with the affected users. Then we must rely on HIDSs to provide at least a minimal level of protection.
2. Host-based intrusion detection systems

Oppositely, HIDSs also bring serious disadvantages that originate in placement of HIDS agents in a potentially untrustworthy environment:

- **Reliability of input**: Malicious data traffic may be intentionally hidden from the operating system or masked as belonging to a different process (e.g., by a rootkit).
- **Access control dependency**: A HIDS agent may be shut down if the malware obtained administrative rights to the computer.

Even though HIDSs bring unique benefits, during the last several years a significantly greater research effort has been devoted to the development of network-based IDSs. Jiankun Hu gives two main reasons: real time and computing resource restraint, and networking factor [Hu10]. We add another three reasons: the ground truth problem, deployment and management issues, and single machine attack class limitation.

The final list is as follows:

- **Real-time and computing resource restraint** [Hu10]: An intrusion should be detected during or immediately after it happened. However, the traditional HIDS techniques (e.g., log analysis, offline integrity checking) bring undesirable delays and/or require a large amount of computing resources. NIDSs usually detect intrusions in real-time.

- **Networking factor** [Hu10]: Nowadays, most applications are network-based. Over the years these applications became a primary attack vector against end hosts. Therefore, there is a strong tendency to protect the network applications in a well-arranged centralized manner.

- **Ground truth problem**: Information supplied to and from a HIDS could be forged or altered by an attacker who took control of an underlying operating system. Existing common privilege control mechanisms allow the administrator to modify every aspect of the system, including the kernel configuration and the code stored in programmable hardware. Also, an attacker with administrator privileges can alter logs to hide any traces of malicious actions. Contrary, NIDSs are usually considered to be hard to compromise.

- **Deployment and management issues**: Current networks are heterogeneous, comprising of hosts with different capabilities and different operating systems. Devising a tool which could be applied across the whole network is a difficult process. Managing dozens or hundreds of HIDSs is both organizationally and technically more challenging than managing several NIDSs.
2. Host-based intrusion detection systems

- **Single machine attack class limitation**: Attacks that are manifested over multiple computers (e.g., horizontal port scans) might not be detected when our view of the events is limited to a single host. Network-based IDSs can correlate events from the entire network and detect such attacks accordingly. An alternative approach is to employ a collaborative HIDS that can benefit from exchanging messages and sharing views with other hosts in the network.

We believe that the importance of the real-time and computing resource restraint will decrease, because current fast hardware allows security researchers to develop new complex detection techniques with required properties. The ground truth problem is also well understood and steadily researched both by the academic community and the private sector. Deployment and management issues are sometimes reduced by the central management consoles. For example, most enterprise-level antivirus software vendors provide server applications with an aggregate view over all installed instances. Unfortunately, these administrative consoles are vendor-specific and are not compatible with products from other vendors. Single machine class limitation is an inherent problem. In global view, we believe HIDS should always seek cooperation with the already existing intrusion detection architecture, contribute to it and benefit from it.

2.2 Standalone network traffic monitoring HIDS

A network traffic HIDS monitors incoming and outgoing packets for signs of unsolicited data flows. Data encryption on the network layer or the transport layer does not affect HIDS capability, because it can obtain the access to the decrypted payload at the application layer. The gathered traffic is interpreted by a HIDS similarly as by client applications, opposite to discrepancies that are typical for network IDSs.

When a host becomes incorporated in a botnet it can be misused for a variety of malicious activities (i.e., performing denial of service attacks, distributing spam, online fraud). These activities cause changes in the host network behavior. Even when the bot does not exhibit any malicious activity, it still communicates with its command and control servers on the internet. Upon discovering that the host participates in any of these specific data flows, we can assume the host was compromised and should be quarantined and examined.

HIDSs often perform the deep packet inspection of data traffic. The average end host connection speed in 2014 Q4 is less than 5 Mbit/s [Aka15]). The average connection speed was less than 25 Mbit/s in all listed countries. At these speeds, the deep packet inspection is possible even without dedicated HW modules. A random packet sampling technique can be used to keep an acceptable CPU load even during traffic peaks.
Bot infected hosts often request commands and updates from botnet command and control servers. Modern malware often exploits HTTP traffic over TCP port 80 for its communication, because this port is usually open at firewalls on the path. Xiong et al. present a HIDS that parses the outgoing HTTP traffic for signs of intrusions and permits or denies the traffic according to a whitelist [XMS+09]. Each HTTP request is processed independently of the requesting browser in case the browser was compromised. A source domain is identified for each HTTP object in the response. If the domain is already on the whitelist, the object is allowed. Otherwise the user is queried whether he or she explicitly requested the object and if so, the domain is added to the whitelist. On one hand, an experiment has shown that users tend to visit a limited set of IP addresses, but regularly. Authors claim that such result supports the usability of the presented HIDS. On the other hand, a common webpage contains objects from several dozen domains (e.g., advertising sites, social networks, user satisfaction monitoring sites). HIDS users, who are often inexperienced in terms of IT security, cannot be realistically asked to decide whether each of these domains should be allowed.

Whitelists are used also by Takemori et al. [TNTM08], who suggested comparing host outgoing traffic with whitelists in order to discover the computer-originated malicious data flows. Initially, whitelists are populated with IP addresses of well-known services (e.g., DNS servers, patch servers, antivirus servers) and of computers in a local network. After the first installation of the operating system a few days learning period takes place, during which users cannot work with the computer and intrusions must not take place. During this period the computer-originated outgoing packets are monitored and their destination domain names are added to the whitelists. After this quite lengthy learning period, packets that do not match any whitelist entry are dropped. To ensure that user-originated traffic is not affected, the traffic is allowed through during a short time after each interaction between the user and the computer (e.g., a keyboard operation). In large enterprise environments with multiple heterogeneous systems the whitelist maintenance can be problematic. A solution could be an automatic updating service that collaborates with well-known security vendors.

Kwon et al. assume the host compromised by a bot can be discovered from an outgoing traffic [KLL11]. They describe two properties that differentiate bot and human processes in a host machine. First condition is whether the behavior was initiated by a user. It can be decided from interaction with input devices and types of Windows GUI reports. Second condition is whether the behavior is malicious. Two types of malicious behavior are classified and used for detection: distributed denial-of-service attacks (DDoS attacks) and spamming. DDoS attacks are recognized by an incoming to outgoing packet count asymmetry and spamming hosts are identified by the quantity and the periodicity of the mail traffic. If both conditions hold for a packet, the responsible process is
reliably identified as malicious by analyzing correlations between API calls and attack traffic, even if process' port-binding information is hidden from HIDS. We highlight this ability because it simplifies incident response and root cause analysis. As such, it makes the proposed system applicable in a high security enterprise environment.

Not-a-bot (NAB) system mitigates network attacks by an automatic validation of the user-originated traffic [GBMR09]. For each request, the request originator is automatically determined. If the originator is the user or an application running on user behalf, the request is allowed, otherwise the request is blocked. Decision on whether the request comes from the user is based on user’s interactions with a computer. After each keystroke or mouse movement there is a short time window during which requests are allowed through. Allowed requests are attested with a digital signature. The attesting module cannot be altered because its integrity is protected by a Trusted Platform Module. The attestation is responder-specific, content-specific and challenger-specific. Attested requests are analyzed by verifiers in network who may take appropriate actions. NAB can be used with existing network protocols, however, client applications require modifications for NAB to be supported.

We acknowledge that large volume attacks (e.g., DDoS attacks, spamming campaigns, click fraud) can be successfully mitigated with the user-initiated traffic recognition. However, attacks with low traffic volume can be performed during the short period after each user operation, therefore being considered user-originated.

Laurens et al. present the DDoSniffer tool for the detection of outgoing TCP connection attacks and bandwidth attacks in [LSD09]. Connection attack is defined as an attack that violates the typical behavior of TCP connections by generating connections with four packets or fewer (e.g., TCP SYN DDoS). Bandwidth attack is defined as an attack with a connection ratio (i.e., the number of incoming packets to the number of outgoing packets ratio) larger than four. The tool monitors the client data traffic. If a packet begins a TCP connection, new records are created in the New-connection table and the Connection table. Packet count of each connection is monitored. If the packet count of a particular connection exceeds four, the appropriate record is deleted from the New-connection table. An alarm is raised if a) the IP spoofing is taking place, b) the number of records in the New-connection table exceeds the predefined threshold NEWCONN, c) at least one connection from the Connection table has the connection ratio greater than R. Values of NEWCONN and R were decided empirically and are constant (i.e., NEWCONN=200, R=4). DDoSniffer’s detection capability is limited only to a subclass of existing DDoS attacks. Fixed thresholds might not reflect the actual network where DDoSniffer is deployed, especially when a P2P traffic is present [HAKK10].
2.3 Collaborative HIDS

Standalone HIDSs can provide a basic protection for individual computers. However, in an enterprise environment this approach might not be sufficient. Standalone HIDSs cannot provide data correlation, they lack the global overview of the infrastructure and usually cannot be centrally managed efficiently. Also, different types of IDSs are differently sensitive to various intrusions. HIDSs are not suitable for a detection of large-scale DDoS attacks, horizontal port scans or spamming campaigns. On the other hand, NIDSs have difficulties with identifying malware intrusions, privilege abuses or focused data thefts.

A collaborative intrusion detection system (CIDS) can be employed to address the shortcomings of standalone HIDSs. Nodes in CIDS exchange relevant audit data (raw data, alerts or both) and share information about intrusions and attackers. Data sharing and correlation are intended to improve both the true positives rate and the false positives rate.

Zhou et al. presented an extensive survey of collaborative intrusion detection systems in [ZLK10]. In the following part of our survey related to the collaborative IDS we take a rather different overall approach. We limit ourselves strictly to the cooperating HIDSs and we focus on exploring how the collaborative intrusion detection could be employed and what problems could be encountered in typical deployment scenarios.

Recent research on collaborative intrusion detection systems aims primarily to design generic frameworks that could incorporate both host-based IDSs and network-based IDSs. Integration could be either direct or indirect. Direct IDS integration is tighter, usually centralized, and within products of one vendor. Indirect integration is usually based on a security information and event management (SIEM) system [SM07].

Aussibal and Gallon proposed a distributed detection platform with an advanced alert processing system, capable of evaluating the severity of alerts [AG10]. The system consists of probes and detection entities. Probes are common IDSs, both HIDSs and NIDSs. Detection entities serve as alert aggregators. Each anomaly detected by a probe is cross-referenced with the Common Vulnerability Scoring System (CVSS) and is assigned a score. If the probe is compatible with the Common Vulnerabilities and Exposures dictionary (CVE), the CVE information is added to the alert message. Alert messages are collected at detection entities. For each observed network flow the sum of alert scores is computed and if the sum exceeds a pre-set threshold, the flow is classified as “ALERT” and all other detection entities are notified. It is imperative to evaluate the reliability of each probe for each class of detectable attacks when using already existing IDSs. A rigorous comparative testing is required for IDSs to collaborate efficiently, benefiting from individual strengths of probes and eliminating their weaknesses.
A distributed IDS that utilizes mobile agents is proposed by Liu and Li [LL08]. The IDS incorporates both HIDSs and NIDSs. There are four types of agents – “host agents”, “net agents”, “manage agents” and “mobile agents”. Host agents reside at a host, collect data and detect any possible host intrusions. Net agents detect intrusions within the network traffic. Reports from host agents and network agents are collected at manage agents. When a suspicious, but not necessarily intrusive, activity is detected, a manage agent spawns a new mobile agent. Mobile agent is a specialized piece of code that visits all hosts where the suspicious activity was reported, collects related contextual information and possibly generates alerts. All information is summarized and transmitted to the manage agent that in turn may use it in the self-learning process. This detection method is inherently reactive, yet it can benefit from rich contextual data collected over multiple hosts.

RepCIDN, a Collaborative Intrusion Detection Network both for HIDSs and NIDSs with a partially-decentralized schema is proposed by Pérez et al. [PMPG13]. RepCIDN uses the signature detection model and supports a dissemination of signatures. The schema is enhanced with a reputation system that uses a Wise Committee (WC). WC members are several most trustworthy NIDSs. WC permission is required for any intra-domain information sharing. The trustworthiness of each node in the security domain is decided by WC members on the basis of interactions it had with the system in the past. Inter-domain communication is forwarded through wise committee leaders in a P2P fashion.

An active probing mechanism to detect the ARP spoofing, malformed ARP packets and the ARP denial-of-service attack in a local network without the need for a central entity is suggested by Barbhuiya et al. [BRR+11]. Received ARP requests and responses are verified by broadcasted confirmation requests. If IP and MAC addresses were already considered trustworthy, ARP packets are accepted. Otherwise a simple verification process is performed during which all hosts are queried. If the attacker is present in the network and attempts the ARP spoofing, query responses from hosts are not uniform and the spoofing is detected. Since ARP spoofing can be easily hindered by network switches, the proposed protocol is suitable especially for wireless ad hoc networks, for which it provides a good protection against ARP-related attacks with little communication overhead.

Zhou et al. designed a distributed detection scheme where end hosts are organized in a P2P network by the Chord protocol [ZXX09]. The P2P network enables a high scalability and provides a high performance. Insider attacks are countered with a node trust mechanism. Each source host has an agent installed. Changes in the amount of packets and in the traffic volume sent to a particular destination are analyzed by a non-parameter CUSUM algorithm ([Pag54]). On finding suspicious flows, the host sends a
2. Host-based intrusion detection systems

*request to the rest of P2P network.* If the host receives a request it calculates the similarity of the anomaly between own data traffic and the requesting node’s traffic. If a similarity threshold is exceeded an alert packet is sent to the victim node. Victim node decides if the attack is in progress based on reports from multiple nodes. This scheme is prone to false negatives under heavy attacks. When access routes to the victim are being saturated a successful packet delivery from source agents to the victim cannot be guaranteed. The request-response mechanism can create high volumes of an undesired data traffic when pulsing attacks are employed or the system is deployed on a large scale.

2.4 Cloud-based HIDS

We differentiate between two primary classes of HIDSs in cloud environment: infrastructure HIDSs and cloud client HIDSs. *Infrastructure HIDSs* are placed at physical servers or inside cloud virtual machines and are intended to protect the server infrastructure of the cloud. Although infrastructure HIDSs are consistent from the technical point of view, their use is influenced by the specifics of cloud environment, frequently the separation of responsibilities. *Cloud client HIDSs* are subscribers of in-cloud intrusion detection services (intrusion detection as a service, IDSaaS). They serve primarily as simple data collection units that may optionally perform data preprocessing and filtering. Audit data is then transmitted to a central high availability detection engine placed in the cloud. The biggest limitation of IDSaaS is its inability to detect intrusions in case of a failure of the link between the cloud service provider and the customer. Such failure may be inadvertent, but also the result of a deliberate (D)DoS attack.

The Cloud Security Alliance white paper [CSA11] states that in addition to traditional functions, IDSs in cloud are required to provide virtualization layer workload monitoring, hypervisor and virtual machine manager integrity monitoring and virtual machine image repository monitoring.

Cloud IDSs can be deployed at the system layer, the platform layer and the application layer. Correlation of audit data from each layer may improve the accuracy of detection. IDSs managed by cloud providers can detect attacks spanning over multiple customers or detect attacks on some customers using data received from other customers. Simultaneous usage of a single physical server by multiple customers is also common and could be considered a security threat by some organizations. Moreover, privacy-sensitive data aggregated at the IDS might subject to laws or internal organization policies. Enforcing these policies in cloud may be hard, especially in situations when an organization does not know the precise physical location where its data is stored [MKL09].
2. Host-based intrusion detection systems

Although cloud providers offer their own security services to customers, these services often lack necessary transparency and may not comply with organization's regional regulations or internal procedures. Moreover, security management and responsibilities in cloud are divided between the cloud service provider and the customer. Customer’s security specialists do not have a full control over the underlying virtual machine or its physical infrastructure. Trust boundaries and areas of responsibility must therefore be established.

A comprehensive paper on cloud infrastructure security was published by Vaquero, Rodero-Merino and Morán [VRMM11]. Modi et al. presented a survey of intrusion detection techniques in cloud [MPB+13].

A generic collaborative framework for bot detection is presented by Takemori et al. [TFSN09]. Victims of attacks report their IP addresses and timestamps of attacks to a central authority. Remaining nodes periodically download the list of victims and compare it with own outgoing packets. If there is a match, a compromised node knows it has been misused. Further inspection of compromised hosts allows discovering command and control servers even if the attack traffic itself has spoofed source IP addresses. Takemori’s approach requires a massive collaboration to be truly functional. Storing the traffic history on the local host might raise privacy concerns when the host is shared among multiple users.

Houmansadr et. al propose a cloud-based system for a complex detection of malware in smartphones [HZB+11]. A network proxy is established between internet and the cell network. The smartphone environment is emulated in the cloud. All incoming traffic is duplicated by the proxy and sent simultaneously to the smartphone and to the emulation platform. High cloud performance allows complex IDS engines that are unsuitable for resource-constrained mobile devices to be used for a real-time in-depth analysis. Each smartphone has a lightweight agent installed that collects user and sensor input, sends it to the emulation engine and optionally performs the intrusion response. Contrary to previous approaches the user input is replicated in real time, not in batches. Therefore, the cloud replica is always kept synchronized with little required bandwidth.

2.5 HIDS security

The question of tamper-resistance should be raised during every HIDS-related discussion. Host-based intrusion detection systems reside directly at the protected hosts, therefore they are vulnerable when an attacker takes control of the underlying operating system. A HIDS can be shut down, its analysis engine could be influenced, critical
files changed or deleted, input data altered or output alerts dropped. Ensuring reliable input and output in a potentially hostile environment is a challenging task.

Molina and Cukier define the HIDS resiliency as “the probability that the HIDS will not be subverted in the event of an attack against the system under supervision” [MC09]. They argue that the resiliency is closely linked with the independency. HIDS should be most independent of the supervised system, because shared system elements can serve as attack vectors. The HIDS resiliency is defined as a quantitative, attack-dependent metric, whereas the HIDS independency is defined as a qualitative attack-independent metric. A sample independence analysis of Samhain integrity checker over Gentoo Linux is provided.

Laureano et al. suggested that HIDSs could be protected in a virtual environment [LMJ07]. Processes and events are monitored inside a virtual machine, but the analysis is performed by a HIDS that is placed on an underlying physical machine. The HIDS is separated from the attacker, but it still possesses all knowledge about the protected system.

VMwall, an application-level firewall for Xen virtual environments with a high tamper resistance is presented by Srivastava and Giffin [SG08]. VMwall’s function is not affected even if the attacker takes control of the protected virtual machine. VMwall utilizes the virtual machine isolation and the virtual machine introspection for a secured monitoring of the network traffic of protected virtual machines. A kernel module intercepts packets destined to and coming from the protected VM and decides if they are forwarded. A user agent correlates packets with processes running inside the VM. Both the kernel module and the user agent are placed in a secured VM. Data structures which are necessary for VMwall function are secured with existing kernel integrity protection mechanisms.

Parno et al. argue that network devices devote a lot of their precious resources to reconstructing the state information that is already known to end hosts [PZP12]. Proliferation of TPM-equipped computers and secure smartphones encourages us to use these trusted elements of end hosts to support the host trustfulness. They designed an architecture where information from trusted clients is collected by trusted verifiers and verifiers make recommendations to network filters how to react on the traffic. Clients have a minimal-size hypervisor incorporated. The hypervisor ensures a secure boot of the client and that the agent application was not modified. Once clients authorize verifiers, clients can cryptographically attest their traffic by a hardware-based cryptographic attestation. Any change to the protective hypervisor layer makes the authentication token inaccessible, forbidding the client to further authenticate its traffic. Network filters allow, block or inspect the network traffic based on the recommendations from verifiers.
2.6 Evasion techniques

Evasion techniques are modifications to existing attacks that are allowing these attacks to proceed undetected by intrusion detection systems. Usually, these modifications only marginally affect the core function of the attack, but simultaneously they significantly change its external properties. Evasion techniques were first described by Ptacek and Newsham [PN98] as an approach to evade detection by a NIDS. Ptacek and Newsham distinguished “insertion”, which involves injection of packets to the NIDS, “evasion”, which involves rejection of packets by the NIDS and “denial-of-service”, which represents an attempt to make the NIDS unable to function. Since then, the scope of evasion techniques widened to include also evasion of host-based IDSs. Evasion techniques can be tailored to an IDS product, a signature database file or even to a specific IDS instance.

We identify five evasion approaches:

- **Code morphing**: Malware signature is masked by compression, encryption, extraordinary encoding or polymorphic coding.

- **Behavior morphing**: The actual attack behavior is modified to distinguish from expected malicious behavior (e.g., order of operations during the attack is shuffled; malicious operations are performed by separate processes seemingly without mutual relations).

- **Timing**: IDSs inherently observe intrusions in a distinguishable time window. By spreading the attack in sufficiently long time the limits of what is considered a “normal” behavior are not crossed. Also, an anomaly IDS that is constantly updating its normal profile can be made to consider malicious activities harmless.

- **Denial of service**: The ability of IDSs to detect or report intrusions is denied. HIDS or some of its components may be shut down, modified or overwhelmed.

- **Duality of interpretation**: The attacker exploits differences between how the input is interpreted by the IDS and by the application. IDS is persuaded that the input is either benign or it is corrupted and should be discarded.

A framework for an evaluation of evasion techniques for botnets is presented by Stinson and Mitchell [SM08b]. The cost associated with evading IDSs consists of two components: implementation complexity and botnet utility. Implementation complexity represents the difficulty of making modifications to the botnet in order to evade detection, ranging from a simple command selection to extensive source code modifications. Botnet utility specifies the impact of modifications on the ability of the botnet to
perform its activities. Modifications could influence diversity of attacks, time required to launch an attack, botnet size, attack rate and synchronization level. Authors suggest designing such detection methods where the evasion techniques against them negatively affect botnet capabilities.

Shadow attacks, a new class of duality of interpretation evasion attacks against behavior-based malware detectors, are proposed by Ma et al. [MDL+12]. Behavior-based detectors usually analyze system calls to detect anomalies. Shadow attacks employ partitioning of sequences of malicious system calls into multiple processes. Therefore, unless the detector can correlate system calls from all shadow processes, the global system calls sequence remains undetected. To conceal their relationship, shadow processes can communicate through ordinary channels (e.g., sockets, memory sharing, file sharing, remote network coordination) or through special covert channels. A compiler-level tool AutoShadow for an automatic transformation of an arbitrary malware source code to a shadow malware executable is created.

Another two new classes of duality of interpretation attacks against signature-based malware detectors are presented by Jana and Shmatikov [JS12]. Chameleon attacks exploit differences between the heuristics used by detectors to determine the type of file and those used by end hosts. The type of the file presented to the detector is incorrectly recognized and therefore an inappropriate signature matching is performed, without regard to specific requirements of the file type (e.g., an archive is considered to be a monolithic file). Werewolf attacks exploit differences between how executables and application-specific formats are parsed between detectors and actual applications or operating systems. Even though the file is recognized, the detector’s parsing ability is usually limited in comparison to the application’s parser. Both classes of attacks require only the file metadata to be modified, allowing a simple masquerading of an arbitrary malware file. Authors conclude that host-based IDSs could be more resistant to some types of chameleon and werewolf attacks, especially when on-access file scanning is employed and when HIDSs are closely integrated with the application (e.g., application itself provides access to an already parsed file).

We believe that at least some classes of attack tools that employ shadow, werewolf or chameleon evasion techniques could be detected by HIDSs capable of analyzing the host’s network traffic. Adapting the network behavior of tools responsible for example for (D)DoS attacks, port scans or malware spreading is rather difficult task. Modification could also lead to a lower effectiveness of the attack tools, in compliance with recommendations from Stinson and Mitchell [SM08b]. Although this approach is predominantly followed by NIDSs, we encourage adopting this technique also for HIDSs, bearing in mind both HIDS advantages and disadvantages.
2.7 Summary

In this chapter, we explored the current state of network traffic monitoring host-based intrusion detection systems. Requirements for an optimal HIDS were formulated. State-of-the-art research on standalone network traffic monitoring HIDSs, collaborative HIDSs and cloud-based HIDSs was presented. Overall characteristics of each class were presented and related issues were examined in detail. The HIDS ground truth problem was presented. Techniques that can be used to evade HIDS were discussed, as well as techniques used to increase HIDS resistance against tampering.

In our view, the future of HIDSs lies mainly in smartphones, tablets and other general-purpose mobile devices. These devices regularly connect to untrusted networks, communicate with possibly infected peers in their range and often contain both valuable personal and enterprise data.

We are convinced that host-based intrusion detection systems should always be considered an important part of an overall intrusion detection architecture. On the other hand, HIDSs may also work separately, without the support, and still provide a decent level of protection. HIDSs should not aim to replace network-based IDSs, but rather serve as a complementary tool. In this architecture, their tasks will be to confirm and stop the intrusion, identify attack vectors and help restore the secured state. The ability to identify attack vectors may be even enhanced to the point where HIDS output can be used as a guideline where to find a forensics evidence of the malicious activity.

We have identified two main research trends across all HIDSs: the utilization of virtualization technologies and the shift towards a real-time detection. Achieving the real-time detection is a necessary step towards functional host-based intrusion prevention systems. It is also a logical progression towards an efficient collaboration with NIDSs. Virtualization can affect virtually every aspect of HIDSs, with attack detection, management simplification and tamper resistance being the most notable. However, in most cases changes must be made to the virtualization layer or a custom-based hypervisor must be created. This may limit the usability and flexibility of virtualization-based solutions. Virtualization-based approaches are indeed functional; however, usually they can be used only for VMs hosted in datacenters. Another important problem to address is how to masquerade the virtual environment as a real environment. Otherwise, the malware that determines it is running in the virtual machine could alter its behavior and evade detection.

Network traffic analysis HIDSs deal with the same problems as NIDSs. We are convinced that aggregating host contextual information with a view of host network traffic provides a new promising research challenge. The challenge is emphasized with collaborative systems. Designing protocols and schemes for intrusion data exchange with
regards to privacy requirements, limited node trust and frequent node movements will be particularly demanding. Existing systems that incorporate both HIDSs and NIDSs usually put more trust on NIDSs and HIDSs serve as an additional source of information. It is a logical solution and we expect to see more similar models. We do not expect a wide deployment of purely collaborative network analysis HIDSs. They are suitable only for corporate networks, which is a traditional domain of NIDSs and they do not bring enough benefits to overcome this fact.

Cloud-based intrusion detection is a fostering concept. Low client resources requirements combined with minimal user knowledge requirements predetermine the cloud-based IDS to become very popular, particularly among home users and small companies that cannot afford own IT security specialists. As with collaborative systems, privacy and trust issues must be addressed. Using cloud-based IDS could be somewhat problematic on mobile devices with pay-per-use internet connection, because the IDS can generate potentially huge network traffic. We believe the number of cloud-based IDS services will grow and so will their diversity.

Interesting findings are linked with user-friendliness. Since HIDSs directly impact common users, user-friendliness is a determining factor in HIDSs popularity. Our survey has shown that keeping reasonable memory requirements is an important goal of almost every HIDS, and CPU and memory consumption is usually indeed low. On the other hand, many solutions have a very high false positives rate (i.e., 3-5% percent). We are convinced that a system that disturbs users from their ordinary work is unacceptable and cannot be deployed in real environment. Especially, large organizations with hundreds of client computers do not want their security specialists to waste time with repeating false alarms. The same situation is with HIDSs that require some form of human input (i.e., solve a puzzle, confirm change) unless they are carefully balanced.

During security experiments, the emphasis is put on the false positives rate and the false negatives rate. Our informal observations confirm the findings of Killrouhy and Maxion [KM11], who claim that only around 50% of security research papers use comparative experiments. We hypothesize that the reason is the unavailability of shared test data, but also the unavailability of actual HIDS implementations. Therefore, we make our dataset available, which is another contribution of this thesis presented in Chapter 5.
3 Traffic features of common standalone DoS attack tools

Even though denial of service 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.

Approximately 70% of all observed denial of service attacks exhibit attack traffic bandwidth less than 1 Gbps, according to a report from Radware [HGBEO14]. Almost 40% of all attacks do not even exceed 10 Mbps. Attacks of such power can be created with a limited number of machines that do not have to be synchronized in a botnet. One possible way is to use standalone DoS tools.

Standalone DoS tools support a wide variety of attacks. They may be released as proof-of-concept implementations of recently discovered attack types (e.g., Slowloris, RUDY). The tools are either freely released, possibly as open source, or made available at hacking forums. They are usually easy to use, but they do not support a central remote management and require a manual configuration by an operator. Moreover, standalone tools are common inspirations for botnets. 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 attack traffic samples under pre-defined conditions and with non-interfering background traffic might be extremely difficult.

Standalone DoS tools can be considered ideal traffic generators for research on DoS attacks thanks to their wide availability and their support for contemporary DoS attacks. However, 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 by academia. The list of tools and techniques actively used in real environment contains advanced tools such as LOIC, HOIC or Slowloris. Conversely, academic concepts are notoriously evaluated with obsolete and in practice already forgotten tools, most notably TFN, TFN2k [BT00], Shaft [DLD00], Trinoo [Dit99a], Knight, mstream [Mur99] and Stacheldraht [Dit99b] 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 periodicals (e.g., [AMG+12, BKBK13, YKP+13]). These tools
no longer reflect contemporary real DoS attacks. Compared to the listed tools that are used in academia, real world DoS attacks went through an incredible development.

These are the key changes:

- **Attack infrastructure.** While old tools use exclusively agent–handler infrastructure with spoofed IP addresses, contemporary attacks employ also direct non-spoofed communication between the attacker and the victim (e.g., for application layer attacks), amplification or botnets without handler layer.

- **Attack types.** Brand new classes of flooding application layer attacks, slow attacks and reflection attacks has emerged and gained on popularity. Attack classes such as ICMP flooding or Smurf attacks are no longer used, because either underlying vulnerabilities were fixed or these attacks are too visible and easy to mitigate for current detection systems.

- **Attack properties.** Specific of these tools may no longer be valid. For example, TFN2k calculates incorrect packet checksums and inserts zero in the TCP header length field.

- **Overall performance.** Contemporary DoS attack tools can generate a significantly higher traffic volume.

They changed in terms of overall performance, attack types popularity, but also in terms of attack properties.

In this chapter we focus 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. The results presented in this chapter were published in [BM15]. Further technical details can be found in our technical report [Buk14].

We analyze the standalone DoS attack tools from perspective of source-end detection due to reasons listed in Chapter 3.1. Source-end detection can take place at first-mile router (see Chapter 3.1) or directly at the source host (see Chapter 2.2). Our analysis will assist best to researchers focusing on source-end DoS detection solutions, such as D-WARD [MR05a].
3. Traffic features of common standalone DoS attack tools

3.1 Related work

Overall requirements and characteristics of DDoS attack detection and defense systems near the source are given by Mirkovic et al. in [MPR03]. They argue that truly enormous DDoS attacks cannot be countered with the victim end defense by itself, because anti-DDoS mechanisms may be overwhelmed themselves leaving the source end defense to be focused on. 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 resources available to these source-end devices cumulatively.

We add that 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.

Advantages:

- **Congestion avoidance**: Restraining attack streams near the source preserves internet resources, which would be depleted by the attack traffic.
- **Small collateral damage**: Rate limiting or traffic filtering near the source negatively affects minimum legitimate users.
- **Easier traceback**: Being close to the source facilitates an easier attacker traceback and a simplified post-attack investigation.
- **Sophisticated detection strategies**: Routers closer to sources are likely to relay less traffic than intermediate routers and can therefore dedicate more of their resources to the DDoS defense.

Disadvantages:

- **Defense effectiveness**: Attack packets are very similar to benign packets. Traditional detection methods are based on input/output connection counts, the traffic volume or incoming/outgoing packet rates. However, these methods may be inoperative since only a fraction of the attack traffic is observable at the source.
3. Traffic features of common standalone DoS attack tools

- **Source-end response must be selective.**

- **Deployment incentive**: Deployment of tools that are able to detect outgoing DDoS attacks presents expenditures for the owner of the source network, yet the protection is supplied to the victim. Therefore a low deployment cost and a low number of false alarms are highly desirable properties of the source end defense. Benefit for the source end may be for example the identification of hosts that were incorporated into a DDoS attacks performing botnet.

Related work on attacks detection directly at the source host is presented in Section 2.2. DoS detectors with sensors placed on the first-mile router are listed below.

A fast stateless SYN flood detection method for first mile or last mile routers was proposed by Wang et al. in [WZS02]. SYN floods are detected by monitoring statistical changes in numbers of SYN and FIN packets. Under normal conditions, numbers of SYN and FIN or RST packets are equal. However, during the attack, the victim is unable to respond to a majority of SYN packets so the ratio is changing. The non-parametric CUSUM algorithm is used to pinpoint the existence and the time of change.

Mirkovic and Reiher present D-WARD, a DDoS detection and response solution for source network routers. D-WARD monitors packet flows of each host in its network [MR05a]. Each TCP flow with a *sending rate significantly higher than a receiving rate* is marked as an attack. Destination hosts, to which a high number of simultaneous connections was open from the network, are identified as victims of a subnet spoofing attack. UDP attacks are detected on a per-protocol basis, i.e., DNS, NTP and media streaming. In addition to the attack detection, the observation component gathers per-connection statistics and periodically compares them to a legitimate connection model. Rate limiting is applied to malicious data flows. D-WARD can be deployed as a stand-alone system or as a source end component of a distributed defense. An improvement of D-WARD is presented by Kang et al. in [KZJ05]. Abnormal changes in the flow sending rate to response rate ratio are sought by the non-parametric CUSUM algorithm. Applying the sequential change point detection to the existing system leads to a decrease of false positives rate and to an increase of detection rate.

Multi-feature extraction and detection method for highly distributed DDoS attacks was suggested by Kang et al. in [KLZL07]. TCP header flags field, IP ID field and S-D-P triplet (source IP address, destination IP address, destination port) are considered as input features. Received data flow is split into fixed length parts. Observation subsequences based on input features are extracted from each part. Each subsequence is processed with the Multi-stream combined Hidden Markov Model and its output probability is calculated. If the probability is smaller than a threshold of the output, the subsequence is marked as “questionable”. If the ratio between the number of questionable
subsequences and the number of all subsequences is higher than a threshold, a DDoS attack in progress is reported.

Collaborative TCP SYN attack detection system was presented by Xiao et al. in [XCH06]. The scheme allows for a recognition of the IP spoofing in a spoofed network from unexpected TCP SYNACK responses. Client detectors are deployed on edge routers of the innocent network. Client detectors provide alert notifications to a server detector, which is employed by the protected server. The Counting Bloom filter stores occurrences of SYN and SYNACK segments. Each time a SYN packet originates in the source network, respective positions of the filter are incremented. Each time a SYNACK packet is incoming, respective positions are decremented. Initial values of positions are set to 0. If the value of any position drops below 0, a new suspicious alarm is reported. Alarms are then evaluated according to a history of occurrences of source IP addresses (i.e., victim server addresses). If the computed score exceeds a predefined threshold an attack is reported to the appropriate server detector. However, a willingness to deploy the described system remains questionable. The owner of the network receives no benefit from its operation, not even the knowledge of misuse of its own hosts. On the other hand, their reputation may be damaged if the system produces false positive alarms.

The same SYN / SYNACK phenomenon is observed from the actual attack source network by Nashat et al. in [NJH08]. The Counting Bloom filter classifies incoming SYNACK packets as firsts or as retransmissions by their acknowledgement number. The difference in numbers of received SYN and SYNACK packets is then normalized and the CUSUM algorithm calculates the point in time when a TCP SYN attack began. Unfortunately, it means that each packet’s payload must be examined and the TCP sequential number must be retrieved. It could lead to an increased router CPU load as the TCP sequential/acknowledgement number is usually not present in the NetFlow data [Cla04] and has to be gathered by a deep packet inspection.

Both previous methods can be successfully used only against DDoS attacks, which employ the IP spoofing. Application layer attacks and trivial network layer attacks cannot be detected.

Flow statistics are gathered by a detection and response system for perimeter routers, as proposed by Malliga et al. in [MTJ08]. Authors assume that source nodes open many new simultaneous connections to the victim during the attack. Detection has two phases. In the learning phase the number of connections opened from each source host is observed. A threshold information entropy as well as its lower and upper bounds are calculated for every IP address. When the current entropy value falls outside bounds, then an attack in progress is assumed and suspicious flows are rate limited.

Cheng et al. define the IP Address Feature Value (IAFV) algorithm for source network routers. The algorithm reflects abrupt traffic changes, flow dissymmetry, source
3. Traffic features of common standalone DoS attack tools

IP addresses distribution and target IP addresses concentration features of the network traffic [CYL+09]. Changes of a network flow state in time are described by IAFV time series. DDoS attack detection is equivalent to a classification of IAFV time series. A support vector machine classifier with several empirically decided parameters is trained and used to identify DDoS attacks.

3.1.1 (D)DoS attack traffic features

Exploration of detailed properties of DoS attacks in the wild received limited to no 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 chapter are extensively and routinely used by hacktivists to manifest their political opinions and by technically unsavvy users to harass other users. For example, Bartolacci et al. describe the practice of “kicking”, when online gamers use simple DoS tools to degrade their opponent’s network connection or even force them out of the game [BLP14].

Onut and Ghorbani argue there is a general lack of research on input features [OG07]. They ranked 673 network features by the their effectiveness 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 [KZ09]. 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. [ZYW10]. Their list of top 5 preferable features (in decreasing importance) is TCP SYN occurrence, destination port entropy, entropy of source port, UDP protocol occurrence and packet volume. Unfortunately, the results based on DARPA [DAR98] and KDD [Lic13] datasets has been repeatedly criticized for not being a good representative sample of actual traffic in a network (e.g., [EVP11, SSTG12]).

Thing et al. [TSD07] 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
3. Traffic features of common standalone DoS attack tools

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. [DIE04]. 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 [Buk14].

Another study aimed at properties of DDoS bots has been performed by Edwards and Nazario [EN11]. 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.

Basic properties of DDoS traffic are frequently listed with DDoS botnet analyses, such as the analysis of Dirt Jumper botnet [AV12] or Miner botnet [PGP12]. 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 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.

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 are focused only on the tools themselves, not deriving general concepts. Examples are analysis of LOIC [Ver11] and HOIC [Bar12].

Several new classes of denial of service attacks have been discovered in the last couple of years. We observe a smart use of old concepts as well as slow increase in application layer attacks.

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 TCP connections with completed three-way handshake. Cambiaso et al. classify slow attacks into four groups: pending requests DoS, long responses DoS, multi-layer DoS and mixed attacks [CPA12]. Several representatives of slow DoS attacks have been dis-
covered already, most notable being Slowloris [Han09] and Slow HTTP POST [CB10].

Kuzmanovic and Knightly presented a new class of low-rate TCP attacks that are able to throttle TCP flows to a small fraction of their ideal rate, while transmitting at a sufficiently low average rate to elude detection [KK03]. The attack exploits an inherent tradeoff induced by a mismatch of defense and attack timescales by sending short bursts of traffic in fixed time intervals. Another variant of a pulsing attack was proposed by Luo and Chang [LC05]. The attack manipulates TCP flows to frequently enter the fast recovery state or the timeout state.

Instead of attacking servers directly, the attacker may decide to target access links to them. During the Coremelt attack, bots send traffic just to each other and not towards the victim [SP09]. Bots are saturating the targeted network link with only legitimate traffic, therefore making the detection extremely difficult. The more advanced Cross-fire attack aims to employ a low number of bots to flood selected few network links between bots and victim servers with low-intensity flows [KLG13]. The aggregation of flows results in overwhelming the links, therefore legitimate connections can no longer transmit data towards the victim servers. In both cases, the attack is undetectable by the victim servers, because they never receive any traffic from the attack hosts.

### 3.2 Experiment

We collected a representative set of standalone DoS tools, executed them in a controlled environment and recorded the traffic properties of attacks that they generate.

#### 3.2.1 Environment

The standalone DoS attack tools were tested in a closed virtual environment. Attack traces were collected directly at the source attack hosts. No background traffic was generated in order to gain a clear record of attack traffic. The collection process is described in detail in Chapter 5.2.

#### 3.2.2 DoS tools selection

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 [Arb14]. 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 [Wil12].

We excluded the tools that are exclusive for a specific target application (e.g., Apache Killer). Unfortunately, we were unable to find any representatives of reflection or amplification attacks among standalone DoS tools, therefore these attack classes were excluded as well. 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.

The full list of analyzed tools, versions, respective sources, supported attack types and tool identifiers used in later text is provided in Table 3.1. We are convinced that this list accurately represents the types of standalone DoS tools that can be currently encountered during real attacks.

3.2.3 Measurement

We review DoS attack tools from the viewpoint of the 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 internal principles of attacks from the 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. The other configurations were chosen in order to test primarily the settings that can alter the form of produced network traffic. We did not distinguish between successful and unsuccessful attacks. Traffic samples with 60 second and 300 second attack duration 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. 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 the limited client bandwidth that is commonly saturated with legitimate network traffic.
3. Traffic features of common standalone DoS attack tools

Table 3.1: Standalone DoS tools that we collected and analyzed. We list their supported attacks as well as the sources where these tools were encountered.

<table>
<thead>
<tr>
<th>Name</th>
<th>Version</th>
<th>Source</th>
<th>Tool ID</th>
<th>Attacks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Anonymous DoSer 2.0</td>
<td></td>
<td>OpUSA, OpMyanmar</td>
<td>AD</td>
<td>HTTP</td>
</tr>
<tr>
<td>AnonymousDOS</td>
<td></td>
<td>Representative</td>
<td>ADR</td>
<td>HTTP</td>
</tr>
<tr>
<td>BanglaDOS</td>
<td></td>
<td>Representative</td>
<td>BAD</td>
<td>HTTP</td>
</tr>
<tr>
<td>ByteDOS 3.2</td>
<td></td>
<td>OpIsrael, OpUSA</td>
<td>BD</td>
<td>SYN, ICMP</td>
</tr>
<tr>
<td>DoS 5.5</td>
<td></td>
<td>Representative</td>
<td>DS</td>
<td>TCP</td>
</tr>
<tr>
<td>FireFlood 1.2</td>
<td></td>
<td>OpMyanmar</td>
<td>FF</td>
<td>HTTP</td>
</tr>
<tr>
<td>Goodbye 3.0</td>
<td></td>
<td>OpUSA, ArborNetworks</td>
<td>GB3</td>
<td>HTTP</td>
</tr>
<tr>
<td>Goodbye 5.2</td>
<td></td>
<td>OpUSA, ArborNetworks</td>
<td>GB5</td>
<td>HTTP</td>
</tr>
<tr>
<td>HOIC 2.1.003</td>
<td></td>
<td>OpUSA, OpMyanmar</td>
<td>HO</td>
<td>HTTP</td>
</tr>
<tr>
<td>HULK 1.0</td>
<td></td>
<td>OpUSA, InfoSec</td>
<td>HU</td>
<td>HTTP</td>
</tr>
<tr>
<td>HTTP DoS Tool 3.6</td>
<td></td>
<td>Representative</td>
<td>HDT</td>
<td>slow headers, slow POST</td>
</tr>
<tr>
<td>HTTPFlooder</td>
<td></td>
<td>OpUSA</td>
<td>HF</td>
<td>HTTP</td>
</tr>
<tr>
<td>Janidos -Weak ed.-</td>
<td></td>
<td>ArborNetworks</td>
<td>JA</td>
<td>HTTP</td>
</tr>
<tr>
<td>JavaLOIC 0.0.3.7</td>
<td></td>
<td>OpUSA, OpMyanmar</td>
<td>JL</td>
<td>TCP, UDP, HTTP</td>
</tr>
<tr>
<td>LOIC 1.0.4.0</td>
<td></td>
<td>OpUSA, OpMyanmar</td>
<td>LO1</td>
<td>TCP, UDP, HTTP</td>
</tr>
<tr>
<td>LOIC 1.0.7.42</td>
<td></td>
<td>OpUSA, OpMyanmar</td>
<td>LO2</td>
<td>TCP, UDP, HTTP</td>
</tr>
<tr>
<td>LOIC 1.1.1.25</td>
<td></td>
<td>OpUSA, OpMyanmar</td>
<td>LO3</td>
<td>TCP, UDP, HTTP</td>
</tr>
<tr>
<td>LOIC 1.1.2.0b</td>
<td></td>
<td>OpUSA, OpMyanmar</td>
<td>LO4</td>
<td>TCP, UDP, HTTP, ReCoil, slowLOIC</td>
</tr>
<tr>
<td>Longcat 2.3</td>
<td></td>
<td>Hacker forums</td>
<td>LC</td>
<td>TCP, UDP, HTTP</td>
</tr>
<tr>
<td>SimpleDoSTool</td>
<td></td>
<td>Representative</td>
<td>SD</td>
<td>TCP</td>
</tr>
<tr>
<td>Slowloris 0.7</td>
<td></td>
<td>OpIsrael, OpUSA</td>
<td>SL</td>
<td>HTTP</td>
</tr>
<tr>
<td>Syn Flood DOS</td>
<td></td>
<td>OpUSA</td>
<td>SF</td>
<td>SYN</td>
</tr>
<tr>
<td>TORSHAMMER 1.0b</td>
<td></td>
<td>OpIsrael, InfoSec</td>
<td>TH</td>
<td>HTTP</td>
</tr>
<tr>
<td>UnknownDoser 1.1.0.2</td>
<td></td>
<td>Hacker forums</td>
<td>UD</td>
<td>HTTP</td>
</tr>
<tr>
<td>XOIC 1.3</td>
<td></td>
<td>InfoSec</td>
<td>XO</td>
<td>TCP, UDP, ICMP</td>
</tr>
</tbody>
</table>
DoS tools were executed 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 a flow as a 5-tuple, similarly to NetFlow protocol: 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 per second within the first minute of the attack. Tables contain tool IDs of tools representing each category. If an ID is found in multiple categories, the actual behavior is dependent on the chosen tool settings.

3.3 Analyzed traffic features

3.3.1 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 [Arb14].

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 allow 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.

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. 3.4). 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 3.2.

**Full burstiness:** The attack traffic is delivered only in bursts. Minimal or no traffic is exchanged between bursts (Fig. 3.3). Full burstiness is also very popular with slow attacks, often probably due to guidance by an internal clock.
3. Traffic features of common standalone DoS attack tools

Figure 3.1: BD packet count.

**Regular peaks:** Produced network traffic is very stable except for regular repeating anomalies (Fig. 3.1).

**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. 3.2).

**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>
<thead>
<tr>
<th>Traffic burstiness</th>
<th>HDT, HU, LO4, SL, SF</th>
</tr>
</thead>
<tbody>
<tr>
<td>Full burstiness</td>
<td>BD, HO, LO, LC, UD</td>
</tr>
<tr>
<td>Regular peaks</td>
<td>AD, BAD, DS, GB, HDT, TH</td>
</tr>
<tr>
<td>One-time extreme</td>
<td>ADR, FF, HF, JA, JL, LO, LC, SD, UD, XO</td>
</tr>
</tbody>
</table>

Table 3.2: Traffic burstiness.
3. Traffic features of common standalone DoS attack tools

**Figure 3.2:** GB5 packet count.

**Figure 3.3:** SF packet count.
3. Traffic features of common standalone DoS attack tools

3.3.2 Flow count

An 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 3.3.

**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. 3.6), the flow rate is extremely stable for most tools in an ideal closed environment (Fig. 3.7). 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 indirectly 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. 3.8) or HDT, which alternates between calm no-traffic periods and periods of batch packet sendings.

**Decreasing count:** Several tools such as DS (Fig. 3.5), 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, an 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>
<thead>
<tr>
<th>Stability</th>
<th>AD, ADR, BAD, BD, FF, HF, HO, HDT, JA, JL, LC, LO, SD, XO</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pulsing</td>
<td>HDT, HU, JL, LO, SF, SL, UD</td>
</tr>
<tr>
<td>Decreasing count</td>
<td>DS, GB, HDT</td>
</tr>
</tbody>
</table>

**3.3.3 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
3. Traffic features of common standalone DoS attack tools

Figure 3.5: DS flow count.

Figure 3.6: FF flow count.
3. Traffic features of common standalone DoS attack tools

Figure 3.7: JA flow count.

Figure 3.8: LO2 flow count.
remain constant. Attribution of tools to each flow parallelity type is provided in Table 3.4.

**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 simultaneously. 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>
<thead>
<tr>
<th>All simultaneous</th>
<th>AD, BAD, LC, SF, SL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mostly simultaneous</td>
<td>DS, GB, HDT, HU, LO4, TH, UD</td>
</tr>
<tr>
<td>Long-term consecutive</td>
<td>LO</td>
</tr>
<tr>
<td>Mostly consecutive</td>
<td>ADR, BD, FF, HF, HO, JA, JL, UD, XO</td>
</tr>
</tbody>
</table>

### 3.3.4 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 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. Moreover, efficient IP spoofing attacks require an up-to-date list of available proxy servers or a large number of hosts some of which serve as proxies. Numerous methods for source end DoS detection based on the recognition of IP spoofing were proposed [XCH06, PYHR10] in the past.
3. Traffic features of common standalone DoS attack tools

The popularity of IP spoofing among standalone DoS attack tools is low, SYN Flood DoS being the only representative in our set. On the other hand, IP spoofing is heavily used by DDoS-for-hire services (see Chapter 4.3.3).

3.3.5 Average packet size

Continuous measurement of average connection packet size in time is a simple metric 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 messages 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 in attacker’s network. Table 3.5 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 3.6). 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 a 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>
<thead>
<tr>
<th>Table 3.5: Average packet size with limit – TCP configurations.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Average packet size =&lt; 64 B</strong></td>
</tr>
<tr>
<td><strong>Average packet size &gt; 64 B</strong></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table 3.6: Average packet size with limit – HTTP configurations.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Average packet size =&lt; 64 B</strong></td>
</tr>
<tr>
<td><strong>Average packet size &gt; 64 B</strong></td>
</tr>
</tbody>
</table>

### 3.3.6 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 3.7.

**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. 3.9). 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.
3. Traffic features of common standalone DoS attack tools

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. 3.10).

Table 3.7: HTTP requests per flow.

<table>
<thead>
<tr>
<th>One per flow</th>
<th>ADR, FF, GB, HDT, HF, HO, HU, JA, JL, LO, SL, TH, UD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Infinite per flow</td>
<td>AD, BAD, LC</td>
</tr>
</tbody>
</table>

3.3.7 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
3. Traffic features of common standalone DoS attack tools

storing all observed URIs is inefficient, performance problems can be solved, for example, with the 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 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 3.8.

**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. 3.11). 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 used the page crawling.

**Parameter change:** The base domain and file path remain constant, but the 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 3.12 and 3.13 show the difference between capturing full URIs and without parameters. When URI parameters are discarded, the
internal similarity of traffic demonstrates itself much more visibly.

**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 an 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. User-Agent string monitoring is studied in Chapter 3.3.8.

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 by the victim server. 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>
<thead>
<tr>
<th>URI string set</th>
<th>ADR, FF, GB, HDT, HF, HO, JA, JL, LO, LC, SL, TH, UD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parameter change</td>
<td>AD, BAD, HU</td>
</tr>
<tr>
<td>Random URI</td>
<td>JL, UD</td>
</tr>
</tbody>
</table>

### 3.3.8 HTTP User-Agent

User-Agent (UA) field in HTTP header serves as an identifier of the client sending the request. Its purpose is mainly to allow providing 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 tools to mimic UA strings of popular software. However, in practice UA strings have been successfully used to track generic malware [MV11]. We have identified three filling methods for UA field. Classification of DDoS tools by these methods is given in Table 3.9.

**Missing UA string:** 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, how-
3. Traffic features of common standalone DoS attack tools

Figure 3.11: JA unique HTTP request count.

Figure 3.12: HU unique HTTP request count.
ever, 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 string**: 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 are also tools which either allow inserting an arbitrary static string (e.g., [HDT-P-3]) or filling the field with an 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 string**: The field is filled with a string from a limited set or the string is randomly generated, possibly from basic building blocks. Dynamic UA string aims 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 the Cumulative Bloom filters, can detect the presence of such tool on the host.
3. Traffic features of common standalone DoS attack tools

Table 3.9: HTTP requests User-Agents.

| User-agent field missing | ADR, GB3, GB5, HO, HF, JL, LO1, LO2, LC, UD |
| User-agent field static  | AD, BAD, FF, HDT, JA, LO3, LO4, SL          |
| User-agent field dynamic | HU, TH, UD                                   |

3.3.9 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 the TCP protocol behavior rather than to transmit data.

As is shown in Table 3.10, 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 seven packets after closure (Fig. 3.15). Oppositely, HO is one of the tools whose traffic does not provide any recognizable flow packet count (Fig. 3.16). Excluded were tools whose connections were never closed during the first minute of the attack (AD, BAD, HF).

Table 3.10: 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                               |
3. Traffic features of common standalone DoS attack tools

Figure 3.14: JL flow packet count distribution.

Figure 3.15: UD flow packet count distribution.
3.4 Discussion

Most standalone DoS tools are single-purpose programs that are capable of only one type of attack. Moreover, tools that support multiple attack types can rarely launch several attacks simultaneously. Majority of standalone DoS 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 targeting only one victim at a time. This is an important observation for the source end detection, because statistics of multiple flows aimed at a single target can be included in detection.

3.4.1 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. For clarity and comparability, we provide all classifications in a two tables for TCP-based and HTTP-based features respectively in Appendix A.

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.
3. Traffic features of common standalone DoS attack tools

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 Öke and Loukas [OL07] or Siaterlis and Maglaris [SM05] who 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. [WWWS12] is promising.

Traditional metrics such as a high bitrate and a high packet rate 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 (e.g., LOIC, HOIC) enable to specify the attack performance so it is possible to find a configuration which cannot be detected through volume-based metrics.

3.4.2 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 a form of 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 a 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 what 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 do not 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 and TTL field value. 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.

Example 3 – LO3. The 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 then the flow is closed. Similarities between different flows can be found at the IP layer (e.g., packet sizes, packet count from attacker), the TCP layer (e.g., TCP flag ratios, window size, TCP checksum) and the HTTP layer (e.g., HTTP fields found in the header, URI, parameters).

3.4.3 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 the tool creator. Therefore, the effect of randomization can fre-
3. Traffic features of common standalone DoS attack tools

Consequently 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), URI values (e.g., LO) or even randomly created parameters (e.g., HU) can be used both to evade simple DoS detection systems and to circumvent the 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 their traffic more visible. For example, as noted above, attack tools commonly randomize the User-Agent string of a HTTP request header [Buk14]. While this is a reasonable technique to be used against the victim end detection systems, because the User-Agent string cannot be used to classify attack traffic, constantly increasing number of different observed User-Agent strings significantly raises suspicion of source end detection systems. Even more importantly, many attack traffic features cannot be randomized at all or only with a 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 basic SYN flood.

We emphasize that it is crucial for researchers to include their DoS attack traffic assumptions and any possible evasion techniques in every research output/publication that is dealing with DoS attack detection.

3.5 Summary

The work on this chapter 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 and/or attack tools, is inherently flawed.

We have classified 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. More details about our experiments can be found in our technical report [Buk14]. The collected traces were included in our dataset. The detailed description is provided in Chapter 5.

The key revelation is the presence of repeating operations in all analyzed DoS attack
Traffic. Even though the exact properties of each attack that we analyzed, varied, we have discovered a set of patterns recurring among DoS tools from different creators. We believe these patterns will prevail for a longer time than simple attack signatures. 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.
4 DDoS-as-a-Service

This chapter aims to provide a comprehensive analysis of the Distributed Denial of Service as a Service (DDoSaaS) phenomenon. We evaluate the threat that DDoSaaS poses to low to medium power cloud-based servers or home users. Our goal is to measure the performance of generated attacks and properties of attack traffic, investigate financial aspect of the services, evaluate service popularity and compare their source codes. The information we collected allows us to conduct a grounded assessment of DDoSaaS risks for cloud providers and common users. This chapter is based on our papers about DDoSaaS [BSN15] and cyber attack kill chains [BLM14, BRN15].

DDoSaaS is a next step in monetization of DoS attacks. Previously, managed DoS attacks were not publicly advertised and they had to be ordered at botnet owners through direct communication. That made hiring of DoS attacks available only to a small community of knowledgeable users. DDoSaaS aim to enable performing DoS attacks to anyone with an access to internet.

DDoSaaS usually present themselves as stress testing services (often called booters or stressers), willing to test the resistance of a chosen target to Distributed Denial of Service (DDoS) attacks. The services are accessed via websites that require prior registration. Common features of a DDoSaaS website are: it is in English, lists prices in US dollars and is easily retrievable through mainstream search engines. According to [SW], the websites are frequently accessed through aggregated booter lists (e.g., top10stressers.com, thebestbooters.com), hacker sites (e.g., hackforums.net, hackbulletin.com) or Skype resolvers that translate Skype nicknames to latest IP addresses (e.g., iskyperesolve.com, skypegrab.net).

The websites frequently contain a Terms Of Service page, where service operators disclaim any responsibility for damages caused by users of the service. However, service operators do not check whether a customer is ordering attacks on targets under customer’s supervision.

Most common DDoSaaS customers are online gamers who seek to gain a competitive advantage over their opponents [KM13]. When ordering an attack, the customer has to specify a target URL or an IP address, length of the attack (limited by paid subscription, see Section 4.6.2) and an attack type. These values are inserted into a web form on the stresser webpage and submitted to a back-end server. The server evaluates the request and orders attack servers to initiate an attack.

Services are very customer friendly. The main webpage contains dashboards with news for customers, such as newly available attack types or bandwidth increases and basic service statistics. A ticketing system is usually prepared for customers to report
bugs and any issues. Many services even claim 24/7 support via instant messaging channels and offer occasional promo actions, such as subscription discounts, free trials or an advanced subscription for the price of a lesser subscription.

The bandwidth available for attacks is usually advertised in the order of several gigabits per second or more (e.g., Anonymous Stresser 5 Gbit/s, Quantum Booter 15 Gbit/s). DDoSaaS employs a limited number of powerful servers that send attack traffic. The traffic is usually subsequently amplified by unsuspecting poorly configured intermediaries. DDoSaaSs also quickly adopt newly discovered attack methods. We have encountered sites offering attacks that were amplified through recently discovered vulnerabilities in Joomla content management system, Microsoft SQL Server or Simple Service Discovery protocol (SSDP).

The current modus operandi of DDoSaaS provides a good level of anonymity for both providers and technically knowledgeable users. Payments can be sent by anonymous cryptocurrencies (see Section 4.6.1), attack traffic has spoofed source IP addresses and webpages can be accessed through anonymization proxies.

4.1 Related work

The first academic paper that focused solely on DDoSaaS services was published by Karami and McCoy in 2013 [KM13]. Authors analyze the leaked database of the twBooter service and execute several simple attacks against their server. Key revelations are that the attack traffic is generated by servers, attack strength is sufficient to disrupt low to medium-sized web sites, primary service customers are gamers who prefer short attack lengths and most frequent targets are either game servers or game forums.

Yu et al. discuss the threat of DDoS attacks against cloud-based servers as a resource competition problem [YTGW14]. They observe that even though the cloud has enough resources to overcome DDoS attacks, the resources are not distributed as needed by customers. Specifically, virtual machine instances are usually reserved with fixed computational, memory and bandwidth limits. A DDoS attack may cause that these limits are exceeded, overwhelming the target instance.

DirtJumper botnet is analyzed by Büscher and Holz in [BH12]. They point out that the cybercrime ecosystem provides DDoS-for-rent services that customers may use without having to take care of the backend infrastructure. Prices range from 5 USD per hour to 50 USD per day. They argue that DirtJumper botnet operators used DDoS attacks to disrupt their competitors (e.g., hacking forums, malware distribution sites, botnets). We have observed similar situation with DDoSaaS. News at stresser homepages frequently mentioned that homepages were suffering or recently recovered from DDoS attacks.
Mirkovic and Reiher present a taxonomy of DDoS attacks in [MR04]. The criteria for DDoS attacks classification are degree of automation, exploited weakness to Deny Service, source address validity, attack rate dynamics, possibility of characterization, persistence of agent set, victim type and impact on the victim. In their taxonomy, attacks from DDoSaaS are assigned to these respective classes: DA-3: Automatic, EW-2: Brute-Force, SAV-1: Spoofed Source Address, with unpredictable attack rate dynamics (ARD), PC-1: RAVS-1 Filterable, PAS-1: Constant Agent Set, VT-4: Network Attacks and IV-1: PDR-1: Self-Recoverable.

The crimeware-as-a-service (CaaS) business model was investigated by Sood and Enbody [SE13]. In the CaaS model, roles for service creators and service operators are divided. The authors emphasize the importance of crime forums for advertising and e-currencies for exchange. Web Money is cited as an online payment system that is used extensively in the underground market. DoS attack order is described as a process when key communication between the seller and buyer takes place on an IRC channel.

Investigation into DDoS-for-hire services was highly publicized by articles that have been published by a well known computer security expert Brian Krebs (e.g., [Kre13, Kre14]), who was also fairly successful in tracking several service owners. Krebs argues that most stresser services are operated by US citizens who possess a limited knowledge, rely on PayPal payment system and hide their webpages behind the CloudFlare content delivery network. The author also points out that the source code of DDoSaaS web pages may be frequently reused.

Santanna et al. published two studies about DDoS-for-hire services [SvRDS+15, SDSP15]. In [SvRDS+15], authors analyze properties of 7 DNS-based and 2 CHARGEN-based DDoSaaS-generated attacks directed against a university network. Compared to their study: (1) We analyze the attacks from the perspective of a cloud-based server. (2) Our scope includes many more independent attacks with a greater variety of attack types. (3) We estimate the attack success rate and evaluate also the application layer traffic properties. Oppositely, [SvRDS+15] complements our work with the investigation into the geolocation of reflectors and the discussion of a competition between various DDoSaaSs.

Second paper by Santanna et al. focuses on the analysis of booter databases [SDSP15]. The paper provides an extensive overview of DDoSaaS user/customer behavior, which fits in with our analysis of economical aspect of DDoSaaS, as well as information about the user location. Our aggregated database with more sources and more records also confirms observations of Santanna et al. that most attacks are shorter than 10 minutes and UDP-based attacks are the most popular.
4. DDoS-as-a-Service

4.2 Dataset description

The list of domains that we investigated with respect to DDoSaaS contains 542 records. This list was constructed from searches for keywords such as booter, stresser, ddos-for-hire or ddosaas. These searches were run at Google search engine, YouTube and Hackforums.net site. In order to confirm that a particular domain is hosting a DDoSaaS website, at least one of the following conditions must have been fulfilled:

- The website behind the domain name is still accessible and belongs to a DDoSaaS service.
- A snapshot of the index webpage exists at a third party store (e.g., Google cache, CloudFlare cache, web.archive.org, etc.). The snapshot shows that the index webpage belonged to a DDoSaaS service.
- The domain name was mentioned on a hacker forum in a discussion about DDoSaaS services to be running and serving customers.

We confirmed 423 websites to be associated with DDoSaaS, 84 of which were accessible at some point during our investigation. We were able to create user accounts on 71 of them, to list supported payment methods from 82 and to list available subscription offers from 62 of them.

Most detailed information about the popularity of DDoSaaS can be extracted from leaked databases. Similarly, the internal working of DDoSaaS websites is best evaluated from the website source code. We have collected 53 archives of DDoSaaS website source code and separate 31 database files that have been released to sites such as pastebin.com or leakforums.org.

Database files contain records about user accounts, attacks and payments. We aggregated attack records and user records from multiple databases in order to build a comprehensive view which is not specific to any given DDoSaaS service. Unfortunately, only one payment database file was available for quantitative analysis. An aggregated summary of the databases is provided in Table 4.1.

The source code archives contain a total of 23,443 files with 13,983 unique MD5 hashes. Aggregated statistics of source codes are listed in Table 4.2. Each archive has an associated name of a stresser whose files it supposedly contains. However, in most cases we were unable to verify that the archive is indeed related to the announced stresser, except for trivial checks such as verifying the logo image.

In order to analyze the properties of attack traffic, we created a dedicated high performance cloud-based server at the Amazon Elastic Cloud [Ama]. We bought subscriptions
4. DDoS-as-a-Service

Table 4.1: Database files summary.

<table>
<thead>
<tr>
<th>Database</th>
<th>Booters</th>
<th>Records</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attack logs</td>
<td>17</td>
<td>153,578</td>
</tr>
<tr>
<td>User logs</td>
<td>31</td>
<td>90,962</td>
</tr>
<tr>
<td>Payment logs</td>
<td>Quantum</td>
<td>16,990</td>
</tr>
</tbody>
</table>

Table 4.2: Source code files summary.

<table>
<thead>
<tr>
<th>File extension</th>
<th>Files</th>
<th>MD5s</th>
</tr>
</thead>
<tbody>
<tr>
<td>png/gif/jpg</td>
<td>14,676</td>
<td>8,770</td>
</tr>
<tr>
<td>php</td>
<td>4,094</td>
<td>2,285</td>
</tr>
<tr>
<td>js</td>
<td>1,832</td>
<td>1,227</td>
</tr>
<tr>
<td>html/htm</td>
<td>431</td>
<td>316</td>
</tr>
<tr>
<td>other/no ext</td>
<td>2,410</td>
<td>1,385</td>
</tr>
<tr>
<td>Total</td>
<td>23,443</td>
<td>13,983</td>
</tr>
</tbody>
</table>

at 16 DDoSaasSs and ordered numerous attacks against this server between December 2014 and April 2015. Attacks were directed either to the server IP address or to a hosted dummy webpage that mimicked a webpage of a video gaming clan. Traffic records were collected for 300 seconds starting just prior to an attack launch, while the attack itself was executed for 30 s. A total of 272 attacks were recorded. Every attack method was tested twice in order to increase the chance of successful attack traffic recording (see Section 4.3.1). The recorded attacks are listed by attack classes and source DDoSaasSs in Table 4.4. Further information about the collected attack traffic traces is available with our DDoS-Vault dataset in Chapter 5.4.

Detailed listing of tested services, global rank of their web pages [SW], estimated monthly visits, numbers of created accounts and number of performed attacks can be found in Table 4.3. We focused on highly popular, extensively used DDoSaasSs with many users. Statistics show that DDoSaasSs are used by tens of thousands of users and are responsible for a staggering number of DDoS attacks.

4.3 Traffic analysis

4.3.1 Attack success rate

An attack was considered successful if its power exceeded predefined bitrate and packet rate limits (see below) and if the real attack type corresponded to the attack type requested by the
Table 4.3: Booter statistics for February 2015. Global ranks and monthly visit estimates were collected from [SW]. Total user account and executed attack statistics were collected directly from dashboards at stresser web pages where available.

<table>
<thead>
<tr>
<th>Booter</th>
<th>Global rank</th>
<th>Est. visits</th>
<th>Accounts</th>
<th>Attacks</th>
</tr>
</thead>
<tbody>
<tr>
<td>anonymous-stresser.com</td>
<td>571,894</td>
<td>25,000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>booter.in</td>
<td>258,375</td>
<td>55,000</td>
<td>6,324</td>
<td>22,635</td>
</tr>
<tr>
<td>booter.io</td>
<td>464,756</td>
<td>35,000</td>
<td>9,336</td>
<td>45,073</td>
</tr>
<tr>
<td>connectionstresser.com</td>
<td>319,831</td>
<td>40,000</td>
<td>17,444</td>
<td>180,751</td>
</tr>
<tr>
<td>destressbooter.com</td>
<td>659,154</td>
<td>25,000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>hornystress.me</td>
<td>297,124</td>
<td>70,000</td>
<td></td>
<td>16,310</td>
</tr>
<tr>
<td>ipstresser.com</td>
<td>45,082</td>
<td>420,000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>legion.cm</td>
<td>1,254,496</td>
<td>10,000</td>
<td>10,393</td>
<td></td>
</tr>
<tr>
<td>networkstresser.com</td>
<td>215,904</td>
<td>100,000</td>
<td></td>
<td>28,523</td>
</tr>
<tr>
<td>networkstresser.net</td>
<td></td>
<td></td>
<td></td>
<td>20,417</td>
</tr>
<tr>
<td>powerstresser.com</td>
<td>169,402</td>
<td>130,000</td>
<td>10,197</td>
<td>44,273</td>
</tr>
<tr>
<td>quantumbooter.net</td>
<td>323,716</td>
<td>55,000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ragebooter.com</td>
<td>314,984</td>
<td>50,000</td>
<td>14,022</td>
<td>12,148</td>
</tr>
<tr>
<td>restricted-stresser.info</td>
<td>1,821,164</td>
<td>5,000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>titaniumstresser.net</td>
<td>79,299</td>
<td>310,000</td>
<td></td>
<td>305,494</td>
</tr>
<tr>
<td>vdos-s.com</td>
<td>689,739</td>
<td>20,000</td>
<td>17,452</td>
<td></td>
</tr>
</tbody>
</table>

Customer. Power is a key metric of a flooding denial-of-service attack. It is expressed by a bitrate or a packet rate. Bitrate determines the capability to flood network links towards the victim network with an undesired traffic. A high packet rate can cause failures at network devices between attack sources and the victim (e.g., firewall, proxy, office router).

Since DDoSaaSs are primarily used against home connections and small servers, the limits were set in accordance to average internet connection speeds as listed in Q4 2014 report from Akamai [Aka15]. An attack power was deemed sufficient if the average bandwidth exceeded 25 Mbit/s or if the average packet rate exceeded 20,000 packets per second during the 30 s attack period.

Columns Bit and Packet in Table 4.4 show the percentage of recorded attacks of a chosen booter that surpass respective attack power limits. Approximately 51% of all attacks, regardless of source booter, failed to exceed either power limit. Approximately 43% did not even reach 1 Mbit/s. Such attacks can be considered ineffective against any target. There are two possible explanations. The cloud security architecture might have
detected and blocked the attack or DDoSaaS provider failed to generate the attack traffic. Results of Santanna et al. from an academic network without a DoS protection show, that at least a portion of attacks is underpowered even before reaching the victim network [SvRDS+15]. Measuring the precise effectiveness of cloud-based DoS mitigation systems will require further research.

We also observe significant differences in attack power success rate of different booters. There are numerous potential reasons why a DDoSaaS attack strength is low: over-provision of resources, scams, malfunction of backend stressing infrastructure, DDoS-prevention measures at ISP network and/or cloud infrastructure.

We could not identify any time relations between attacks that fail to reach the desired power. We compared attack bitrates and packet rates between each two attacks with the same attack method on the same booter. Out of 135 pairs of such attacks, both attacks failed to exceed the bitrate/packet rate limits at 60 pairs and one attack failed to exceed the limits at 18 pairs. For example, two DNS amplification attacks were executed at hornynstress.me in the span of three hours. The first attack failed to generate any harmful traffic while the second attack reached up to 380 Mbit/s of incoming traffic.

Differences between attack launch times in these 18 pairs oscillate between one hour and three days. Other attacks were also successfully executed in the meantime at the same booter. Therefore, we believe that a combination of factors is behind power drops. Issues at the side of DDoSaaS are not solely responsible. As a consequence, the success rate of attacks against home connections or cloud providers without a DDoS protection may be significantly higher. Further research will be needed to evaluate the conditions that affect the attack power.

A service customer specifies the requested attack type. However, collected traffic records did not always correspond to the requested type. Potential reasons include: maintaining public image (booters claim to have capabilities that they actually lack), malfunction of backend stressing infrastructure or unwillingness to use non-spoofed attacks. The column Type in Table 4.4 shows the percentage of attacks whose dominant portion of traffic corresponds to the customer request.

An aspect of DDoSaaS quality is the speed with which an attack is launched after it is requested by the customer. We measured the time between an attack order and the timestamp of first incoming attack traffic packet. Average time to start an attack was 7 seconds and 80% of the attacks started in 10 seconds or less. Such a rapid response is especially important for gamers, who represent a large portion of DDoSaaS customers.
Table 4.4: Recorded attacks.

<table>
<thead>
<tr>
<th>DDoSaaS</th>
<th>UDP</th>
<th>TCP</th>
<th>HTTP</th>
<th>Bit</th>
<th>Packet</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>anonymous-stresser.com</td>
<td>8</td>
<td>2</td>
<td>16</td>
<td>15</td>
<td>19</td>
<td>100</td>
</tr>
<tr>
<td>booter.in</td>
<td>10</td>
<td>6</td>
<td>0</td>
<td>50</td>
<td>56</td>
<td>63</td>
</tr>
<tr>
<td>booter.io</td>
<td>8</td>
<td>4</td>
<td>0</td>
<td>50</td>
<td>67</td>
<td>83</td>
</tr>
<tr>
<td>connectionstresser.com</td>
<td>10</td>
<td>2</td>
<td>0</td>
<td>67</td>
<td>50</td>
<td>100</td>
</tr>
<tr>
<td>destressbooster.com</td>
<td>20</td>
<td>0</td>
<td>2</td>
<td>5</td>
<td>0</td>
<td>73</td>
</tr>
<tr>
<td>hornystress.me</td>
<td>18</td>
<td>14</td>
<td>2</td>
<td>32</td>
<td>44</td>
<td>76</td>
</tr>
<tr>
<td>ipstresser.co</td>
<td>14</td>
<td>2</td>
<td>6</td>
<td>59</td>
<td>82</td>
<td>100</td>
</tr>
<tr>
<td>legion.cm</td>
<td>2</td>
<td>0</td>
<td>8</td>
<td>0</td>
<td>0</td>
<td>80</td>
</tr>
<tr>
<td>networkstresser.com</td>
<td>12</td>
<td>2</td>
<td>0</td>
<td>86</td>
<td>86</td>
<td>21</td>
</tr>
<tr>
<td>networkstresser.net</td>
<td>10</td>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>100</td>
</tr>
<tr>
<td>powerstresser.com</td>
<td>4</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>100</td>
</tr>
<tr>
<td>quantumbooter.net</td>
<td>5</td>
<td>4</td>
<td>0</td>
<td>56</td>
<td>78</td>
<td>100</td>
</tr>
<tr>
<td>ragebooster.com</td>
<td>10</td>
<td>6</td>
<td>0</td>
<td>31</td>
<td>6</td>
<td>75</td>
</tr>
<tr>
<td>restricted-stresser.info</td>
<td>8</td>
<td>4</td>
<td>18</td>
<td>21</td>
<td>29</td>
<td>100</td>
</tr>
<tr>
<td>titaniumstresser.net</td>
<td>4</td>
<td>2</td>
<td>4</td>
<td>100</td>
<td>50</td>
<td>100</td>
</tr>
<tr>
<td>vdos-s.com</td>
<td>4</td>
<td>12</td>
<td>0</td>
<td>63</td>
<td>88</td>
<td>88</td>
</tr>
<tr>
<td>Total attacks</td>
<td>147</td>
<td>69</td>
<td>56</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Class success (%)</th>
<th>Bit</th>
<th>Packet</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>57</td>
<td>17</td>
<td>7</td>
</tr>
<tr>
<td></td>
<td>42</td>
<td>55</td>
<td>18</td>
</tr>
<tr>
<td></td>
<td>72</td>
<td>100</td>
<td>100</td>
</tr>
</tbody>
</table>

4.3.2 Attack power

A histogram of measured bitrates of attacks in our dataset is shown in Table 4.5. Bitrate only rarely exceeds 1 Gbit/s. The attack types most likely to reach a high bitrate are CHARGEN and DNS. TCP-based and HTTP-based attacks showed a poor bitrate performance. Successful NTP and SSDP attacks have the clearest power boundaries around 400 Mbit/s and 300 Mbit/s respectively.

An attack packet rate histogram is given in Table 4.6. Attack types associated with the high packet rate are SYN, TCP, SSDP and NTP. TCP SYN attacks exhibit below-average values both for bitrate and packet rate, because this attack is based on the exhaustion of victim connection state table buffer.

Overall, even successful DDoSaaS attacks were not powerful enough to cause a denial-of-service effect against a cloud-based server with high resources. However, the attack
power of successful attacks may be sufficient to saturate uplinks of low- to mid-range servers or at least cause a degradation of service if the traffic reaches the server itself. Conversely, more than 40% of all attacks would fail to overwhelm even the most basic home internet connections with 1 Mbit/s or less download speed.

### 4.3.3 Attack traffic properties

Tables 4.7 and 4.8 show values of most common attack traffic packet lengths and source ports. The booters column specifies how many booters contained the listed feature values in their attack traffic. The traffic column indicates the percentage of all traffic in an appropriate attack class that has the respective property. Source ports clearly show that UDP-based attack employ amplifiers, hence the traffic is incoming from well-known
Table 4.7: Most frequent packet lengths.

<table>
<thead>
<tr>
<th>Attack type</th>
<th>Length (B)</th>
<th>Booters</th>
<th>% traffic</th>
</tr>
</thead>
<tbody>
<tr>
<td>CHARGEN</td>
<td>57</td>
<td>7/9</td>
<td>5%</td>
</tr>
<tr>
<td>DNS</td>
<td>4044</td>
<td>4/10</td>
<td>9%</td>
</tr>
<tr>
<td>NTP</td>
<td>468</td>
<td>9/11</td>
<td>99%</td>
</tr>
<tr>
<td>SSDP</td>
<td>296</td>
<td>10/11</td>
<td>6%</td>
</tr>
<tr>
<td>SYN</td>
<td>40</td>
<td>14/14</td>
<td>93%</td>
</tr>
<tr>
<td>TCP</td>
<td>40</td>
<td>3/4</td>
<td>99%</td>
</tr>
</tbody>
</table>

Ports. Conversely, TCP-based attacks rely on simple IP spoofing and their traffic source ports are evenly distributed. Due to a low number of useable HTTP attack traffic samples, this attack type has been excluded from further research.

Table 4.9 lists some of the manually chosen key unique identifiers that distinguish the attack traffic from the benign traffic. Particularly interesting are similar domain names in DNS amplification attacks. Since a domain name is not inherent to an attack type, we assume that DDoSaaS operators either rent their back-end infrastructure from other providers or buy attack scripts on an open market. Both of these approaches have been known to be used for other service types [SE13].

Table 4.8: Most frequent source ports.

<table>
<thead>
<tr>
<th>Attack type</th>
<th>Port</th>
<th>Booters</th>
<th>% traffic</th>
</tr>
</thead>
<tbody>
<tr>
<td>CHARGEN</td>
<td>19</td>
<td>7/9</td>
<td>92%</td>
</tr>
<tr>
<td>DNS</td>
<td>53</td>
<td>10/10</td>
<td>54%</td>
</tr>
<tr>
<td>NTP</td>
<td>123</td>
<td>9/11</td>
<td>99%</td>
</tr>
<tr>
<td>SSDP</td>
<td>1900</td>
<td>10/11</td>
<td>90%</td>
</tr>
<tr>
<td>SYN</td>
<td>80</td>
<td>11/14</td>
<td>22%</td>
</tr>
<tr>
<td>TCP</td>
<td>80</td>
<td>3/4</td>
<td>&lt;1%</td>
</tr>
</tbody>
</table>

NTP traffic in our dataset shows to be extremely homogenous. All attack packets come from port 123/UDP, carry NTP payload with IP length 468 bytes and the request code equal to 42 (MON_GETLIST) and the . These signs are consistent with a well-documented NTP vulnerability CVE-2013-5211. NTP amplification attacks that were based on the MON_GETLIST command were extensively analysed by Czyz et al. in [CKG+14].

The predominant SSDP attack variant is fairly new, having been first observed in the wild in July 2014 [PLX14]. The attack is amplified by an unpatched small home routers and smart appliances.
4. DDoS-as-a-Service

Table 4.9: Application-layer artifacts.

<table>
<thead>
<tr>
<th>Class</th>
<th>Type</th>
<th>Artifact type</th>
<th>Values</th>
<th>Booters</th>
</tr>
</thead>
<tbody>
<tr>
<td>UDP</td>
<td>DNS</td>
<td>Query domain name</td>
<td>fkfkfkfz.guru</td>
<td>2/10</td>
</tr>
<tr>
<td>UDP</td>
<td>DNS</td>
<td>Query domain name</td>
<td>doleta.gov</td>
<td>2/10</td>
</tr>
<tr>
<td>UDP</td>
<td>SSDP</td>
<td>HTTP Location</td>
<td>IGD.xml</td>
<td>8/11</td>
</tr>
<tr>
<td>UDP</td>
<td>SSDP</td>
<td>HTTP Location</td>
<td>rootDesc.xml</td>
<td>8/11</td>
</tr>
<tr>
<td>UDP</td>
<td>NTP</td>
<td>Request code</td>
<td>MON_GETLIST</td>
<td>9/11</td>
</tr>
</tbody>
</table>

SYN attacks have a standard well-understood form. The incoming attack traffic has spoofed source IP addresses, packet length 40 B and the SYN flag set. Anomalous are TCP window sizes where in 86% of cases values were set to 0.

We can see that attack traffic even from different DDoSaaS shows remarkable similarities, such as packet lengths or application-layer artifacts. The traffic is simple, constructed for maximum attack effectiveness rather than for stealthiness. By using attack reflectors, the DDoSaaS operators sacrifice the capability to randomize attack traffic properties (e.g., packet lengths, source ports, header field values) and circumvent advanced victim DDoS protection solutions. DDoSaaS operators have no control over reflectors, therefore the final attack traffic exhibits a high degree of uniformity. It is fairly easy to configure rules for packet filters to drop or throttle most of the attack traffic. Since the primary DDoSaaS targets are home connections or low-end servers without trained security teams who would react to evolving attacks, we do not expect any sudden increase in the use of detection avoidance techniques in the future.

4.4 Database analysis

We have collected records from leaked databases of 31 services. The statistics presented are based on aggregated records of databases as specified in Table 4.1.

Table 4.10 shows more than 75% of attacks performed by stressers are at most 10 minutes long. Our aggregated records therefore support the results of Karami and McCoy [KM13]. Unsurprisingly, most common lengths of actual boots are equivalent to subscription maximum booter lengths (Table 4.11). Therefore, we can assume that DDoSaaS customers execute attacks for the maximum boot length available to them.

Table 4.12 shows that UDP-based flooding attacks have a significantly higher popularity than TCP-based or HTTP-based attacks. This is likely to be caused, at least partially, by DDoSaaS operators who set UDP attacks as the default option. UDP is also preferable due to its amplification factor. Protocols such as CHARGEN, NTP, SSDP or DNS are
frequently exploited by DDoSaaS operators to increase the impact on victims without having to increase the attacker’s available bandwidth. TCP-based attacks are almost exclusively variations of SYN flooding. Somewhat surprising is popularity of RUDY and Slowloris attacks compared to generic HTTP GET/POST/HEAD flooding.

Moreover, our collected traffic samples show an actual error in RUDY implementations. By design, this attack aims to exhaust the victim’s connection table with a huge number of simultaneous connections. However, the real attack was executed in a compliance with DDoSaaS modus operandi of mass flooding. Instead of opening many simultaneous connections, a huge number of SYN packets with the same source IP address was sent to the victim.

In 84% of attacks, the target port of the attack was 80 (HTTP), followed by ports 3074 (Xbox LIVE), 6005 (BMC Software), 25565 (Minecraft/MySQL), 53 (DNS) and 27015 (GoldSrc game engine – e.g., Counter-Strike). We noticed that port 80 is used by DDoSaaS as a default value, probably because it is rarely filtered by firewalls. Conversely, several representatives of gaming services in the list of most popular target ports confirm the prominent role of gamers among DDoSaaS customers.

Geographic location of victim IP addresses suggests that DDoSaaSs are used primarily against North American and European targets (Table 4.13). Almost 39% of attacks are aimed at the US IP space, FR accounts for 11% and UK for 7%.

We analyzed the payment database records of the Quantum booter from September 2012 to March 2014. The database contains records related to 10,269 paying customers out of 20,695 registered. The mean payment was approximately 21 USD, with median and mode both 8 USD. Total income during the period exceeds 220,000 USD, while monthly income averages at 12,000 USD. That is a significantly higher income than

<table>
<thead>
<tr>
<th>Interval (s)</th>
<th>Attacks</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 – 100</td>
<td>40,836</td>
</tr>
<tr>
<td>101 – 200</td>
<td>27,971</td>
</tr>
<tr>
<td>201 – 400</td>
<td>31,940</td>
</tr>
<tr>
<td>401 – 600</td>
<td>2,649</td>
</tr>
<tr>
<td>601 – 800</td>
<td>2,753</td>
</tr>
<tr>
<td>801 – 1000</td>
<td>6,650</td>
</tr>
<tr>
<td>1001 – 1200</td>
<td>4,076</td>
</tr>
<tr>
<td>&gt;1201</td>
<td>19,671</td>
</tr>
<tr>
<td>Total</td>
<td>136,546</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Length (s)</th>
<th>Attacks</th>
<th>Booters</th>
</tr>
</thead>
<tbody>
<tr>
<td>300</td>
<td>20,866</td>
<td>16</td>
</tr>
<tr>
<td>120</td>
<td>18,414</td>
<td>16</td>
</tr>
<tr>
<td>60</td>
<td>12,570</td>
<td>17</td>
</tr>
<tr>
<td>600</td>
<td>12,557</td>
<td>16</td>
</tr>
<tr>
<td>250</td>
<td>6,843</td>
<td>11</td>
</tr>
<tr>
<td>100</td>
<td>5,749</td>
<td>16</td>
</tr>
<tr>
<td>1800</td>
<td>5,280</td>
<td>10</td>
</tr>
<tr>
<td>90</td>
<td>5,205</td>
<td>11</td>
</tr>
<tr>
<td>500</td>
<td>5,093</td>
<td>14</td>
</tr>
</tbody>
</table>

Table 4.10: Boot lengths histogram.

Table 4.11: Boot lengths popularity.
4. DDoS-as-a-Service

<table>
<thead>
<tr>
<th>Type</th>
<th>Category</th>
<th>Attacks</th>
<th>Booters</th>
</tr>
</thead>
<tbody>
<tr>
<td>UDP</td>
<td>UDP</td>
<td>69,635</td>
<td>11</td>
</tr>
<tr>
<td>ESSYN</td>
<td>TCP</td>
<td>19,744</td>
<td>3</td>
</tr>
<tr>
<td>NTP</td>
<td>UDP</td>
<td>19,416</td>
<td>2</td>
</tr>
<tr>
<td>SSYN</td>
<td>TCP</td>
<td>13,714</td>
<td>10</td>
</tr>
<tr>
<td>RUDY</td>
<td>HTTP</td>
<td>6,310</td>
<td>8</td>
</tr>
<tr>
<td>TCP</td>
<td>TCP</td>
<td>4,648</td>
<td>5</td>
</tr>
<tr>
<td>Slowloris</td>
<td>HTTP</td>
<td>2,958</td>
<td>8</td>
</tr>
<tr>
<td>UDPLAG</td>
<td>UDP</td>
<td>2,929</td>
<td>7</td>
</tr>
<tr>
<td>DRDOS</td>
<td>unknown</td>
<td>2,816</td>
<td>4</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Country</th>
<th>Attacks</th>
</tr>
</thead>
<tbody>
<tr>
<td>US</td>
<td>53,509</td>
</tr>
<tr>
<td>FR</td>
<td>15,811</td>
</tr>
<tr>
<td>UK</td>
<td>9,239</td>
</tr>
<tr>
<td>CA</td>
<td>6,901</td>
</tr>
<tr>
<td>DE</td>
<td>6,317</td>
</tr>
<tr>
<td>NL</td>
<td>4,962</td>
</tr>
<tr>
<td>AU</td>
<td>4,465</td>
</tr>
<tr>
<td>SE</td>
<td>2,622</td>
</tr>
<tr>
<td>Other</td>
<td>34,861</td>
</tr>
</tbody>
</table>

the income reported from the twBooter analysis [KM13] (see 4.1). We expect that the prospect of such future income, coupled with few barriers to entry (see Section 4.5) will lead to an increasing number of DDoSaaS sites in the future.

4.5 Website source code analysis

DDoSaaS webpages are built with PHP and common frameworks, such as Bootstrap, jQuery, jQuery UI, jQuery Sparklines, Modernizr, prettyPhoto or Raphael according to our screening of 65 unique live websites.

We also collected code from 53 DDoSaaS websites and analyzed them for similarities. All sites used PHP scripts, usually supported by the MySQL database. Each site consisted of 105 PHP source code files on average. We calculated the MD5 hashes of all PHP files and found 94 PHP files that were shared/reused (each) by at least 3 sites. We manually analyzed all the 94 shared files to understand their role and divided them into 7 categories. Table 4.14 summarizes our findings. Most shared source codes are files handling user management and CAPTCHA.

Finding similarities in source code files is not always trivial. The cryptographic hash algorithm MD5 can only find perfectly identical files. Even a slight change, such as rewriting an email address in a support ticket submission form, makes MD5 matching impossible. Therefore, we decided to also use the spamsum algorithm implemented in the ssdeep program [Kor06]. The spamsum algorithm calculates context triggered piecewise hashes based on the FNV (Fowler/Noll/Vo) hash algorithm. The algorithm was used to find similarities in the source code of the 53 previously mentioned websites. We calculated how many source files across various sites have their ssdeep hash simi-
4. DDoS-as-a-Service

Table 4.14: Shared source codes categories.

<table>
<thead>
<tr>
<th>Category</th>
<th>Files</th>
<th>Category description</th>
</tr>
</thead>
<tbody>
<tr>
<td>user management</td>
<td>39</td>
<td>Managing user accounts and passwords, user logins</td>
</tr>
<tr>
<td>CAPTCHA</td>
<td>11</td>
<td>CAPTCHA</td>
</tr>
<tr>
<td>index pages</td>
<td>10</td>
<td>Index/home page + news/messages</td>
</tr>
<tr>
<td>lib</td>
<td>10</td>
<td>CCS, JavaScript, ...</td>
</tr>
<tr>
<td>attack management</td>
<td>7</td>
<td>Attack management + statistics...</td>
</tr>
<tr>
<td>PayPal</td>
<td>6</td>
<td>PayPal payments</td>
</tr>
<tr>
<td>misc</td>
<td>11</td>
<td>IP geolocation, IP logging, database access etc.</td>
</tr>
</tbody>
</table>

Larity score higher than 95. On average, each site shares at least one ssdeep hash with 7.45 other sites and has 46.5 ssdeep (>95) similarity relationships (i.e., shared similar files).

We have identified 9 similarity clusters. All websites in a cluster share 10 or more files with ssdeep similarity higher than 95. These 9 clusters were formed by 25 websites. Another 20 sites shared some of their files with others, but without distinctive partners. The remaining 9 sites did not share any similarities.

Such similarity might indicate that the same, or similar, teams are behind multiple services. Another reason might be simple code reuse. As the functionality required by most of the websites for a DDoS services is very similar, and as the source code of many web sites has leaked to the public, the coders of the web sites will be tempted to reuse the existing code. Availability of source code will lead to an easier establishment of new DDoSaaSs.

4.6 Economics

4.6.1 Payment methods

Desired properties of payment methods supported by DDoSaaS are user friendliness for technically unskilled users, anonymity for both seller and buyer and low fees, because exchanged payments are usually fairly small. Service providers also prefer payment methods that do not support payment revocation.

In December 2014, we analyzed payment methods supported by 82 DDoS-as-a-Service providers and found 19 different payment systems. The most popular system was PayPal, which was supported by 63 DDoS services, followed by Bitcoin (42) and Google Wallet (21). Contrary to findings in [SE13], WebMoney was not among sup-
4. DDoS-as-a-Service

We have noticed a distinct move towards the support of cryptocurrencies during our research. Cryptocurrencies are anonymous, decentralized, gaining popularity among the general population, subjected to only limited regulation and payments cannot be revoked as soon as they are included in the blockchain. Bitcoin is now a widely accepted payment method among DDoSaaSs, but we also encountered support for Omnicoin, Litecoin and Dogecoin, mostly thanks to aggregating payment gateways such as Go-Coin or CoinPayments.

Direct use of credit cards is very rare, supported by only 3 services. However, online payment services that allow the user to transfer money from his credit card or bank account to the service account are still common. PayPal, Skrill, Starpass, 4Virtuals, Okpay and Dwolla all fit into this category. With the exception of PayPal, at least one of these services was supported by 11 DDoSaaSs.

4.6.2 Subscriptions

DDoSaaS services provide a variety of subscriptions for different prices. Subscriptions are characterized by price, currency, subscription length, maximal boot time, attack con-
currency and available attack bandwidth. Surprisingly, available attack bandwidth is rarely advertised. Some DDoSaaS services employ client-based botnets as their attack infrastructure. Limited knowledge about bandwidth and availability of particular hosts makes it difficult for service providers to estimate real available bandwidth at any given moment. Attack concurrency is similarly obscured by most services, although generally only one attack is permitted at a time if not otherwise stated.

Subscriptions are time-bound. During the subscription, a customer may initiate an arbitrary number of attacks. Monthly subscriptions are most popular, with more than 95% services offering these, followed by lifetime subscriptions offered by 66%. Price for monthly subscription varies between 1.99 and 35 USD for the cheapest and from 7.5 to 289 USD for the most expensive subscription.

Figure 4.1 shows samples of monthly subscriptions in USD. We can see that the boot length/price ratio does not converge to a common value. Monthly subscription with the same boot length can be ordered for considerably different prices at different services. Oppositely, increasing attack concurrency clearly increases the price of subscription. A combination of low subscription prices with unlimited attacks during the duration of subscription makes the per-attack price potentially extremely low.

In the case of payments via cryptocurrencies, subscriptions are activated automatically. When purchasing a subscription, the customer is offered several payment methods. Once the payment is successfully finished, the customer’s requested subscription is activated without any further intervention from an operator. In the case of cryptocurrencies, automated subscriptions decrease the initial time for the customer to be able to launch attacks to a couple of hours at most. Elimination of a direct contact channel between the customer and the DDoSaaS operator also results in increased privacy for both parties.

4.7 DDoSaaS attack kill chain

We analyze the lifecycle of a DDoS attack launched through a DDoS-for-hire service. We employ the kill chain methodology which was originally developed for tracking the advanced persistent threats. This subchapter is based on our papers [BLM14, BRN+ 15].

Advanced persistent threat (APT) is a term coined for an advanced long term stealthy intrusion into a computer system, with the aim to steal intellectual property from the owner [Man10]. APTs are quickly becoming a nightmare for security officers. APT actors are usually well-funded and possess extensive knowledge. They employ effective intrusion methods such as zero-day attacks or stealth techniques and often have vast infrastructure of compromised servers for support. Their attacks come in campaigns and
are often aimed at only a single target globally, being tailored specifically to the target with reliance on prior reconnaissance. Traditional security measures such as antivirus software, signature based IDSs and systems hardening are largely ineffective against APTs.

To combat the rapidly growing threat of APTs, security experts from Lockheed Martin recommended adopting the concept of kill chain [HCA11]. The idea behind the kill chain is to create a knowledge base of indicators from all observed phases of an APT in order to continuously improve the defense. The struggle between APT actors and defenders leads to a game where APT actors are adapting their techniques to penetrate encountered defense measures and defenders are developing new signatures and indicators to have the upper hand in campaigns in the future. The kill chain concept is quickly becoming the weapon of choice against APTs, being fostered by renowned companies such as RSA [RSA12], Dell SecureWorks [Sec13], Hewlett-Packard [HP13] and NSS Labs [FA12]. Relevant academic research focuses primarily on efficient data aggregation and analysis [BY13, ILCP13].

The kill chain methodic splits every attack into several consecutive phases. Failure to overcome the defense measures at any phase results in the interruption of the entire process. Detection of an intrusion during a certain phase implies that all previous phases were completed successfully. The phases of kill chain are as follows: reconnaissance, waponization, delivery, exploitation, installation, command and control, actions on objectives [HCA11].

The cyber kill chain concept is not limited to APTs. Harris, Konikoff and Petersen investigated the application of kill chain on distributed denial of service (DDoS) attacks. While performing DDoS attacks can always be considered an intended Action on Objective, authors also look for DDoS-related events at other phases of the kill chain [HKP13]. Company Imperva compiled a timeline of a DDoS campaign that was launched by the hacktivist group Anonymous in [Imp12]. The campaign consisted of a recruiting and communications phase, a reconnaissance and application attack phase and a DDoS attack phase.

We apply the kill chain methodic to evaluate various phases of attacks generated by DDoSaas. Compared to other intrusions, a DDoSaas kill chain is specific in its division of activities between the customer and the service provider.

4.7.1 Reconnaissance

The potential attacker attempts to identify IP addresses and domain names of the chosen target. In the trivial case, the attacker simply uses a well-known domain name or gains the needed information through basic system tools, such as netstat.
However, gaining a target IP address of an opponent is not a trivial task for video game players. Video games usually employ a client-server architecture which disallows a direct communication between participating players. Therefore, customers of DDoSaaS frequently rely on so-called resolvers. Resolvers allow mapping between nicknames of a chosen service (e.g., Skype) and an IP address. Even though the accuracy of resolvers is debatable, they are very popular. DDoSaaS webpages frequently include resolvers themselves or provide a link to associated services.

Another aspect of reconnaissance is the identification of weaknesses in DoS defense of the victim. Since DDoSaaS allows to execute a seemingly unlimited number of attacks, DDoSaaS users may test several different attack types at different IP addresses/domain names during the Weaponization & Delivery phase. The attack impact information that is collected during the test attacks is subsequently used to identify the most devastating attack.

4.7.2 Weaponization & Delivery

The attacker selects the parameters of the future attack. Attack execution form usually requires not only the target IP address or domain name, but also the desired attack length, the attack type and the target port. However, none of the services that we analyzed allowed to specify advanced attack parameters such as TCP/IP header fields randomization, payload content or flow behavior. Many DDoSaaSs also do not give the attacker any control over the final attack power. The possibility of DDoSaaS-generated attack customization is significantly lower than the customization options usually found at standalone DoS attack tools.

A DDoSaaS usually offers between 5 and 10 different attack types. The choice of attack type also determines the delivery of attack traffic towards the victim. UDP amplification flooding attacks with spoofed source IP addresses are the most popular (see Table 4.12).

4.7.3 Exploitation

Even though the scope of possible DDoS attacks is very wide, DDoSaaSs prefer rudimentary flooding attacks to take down the victims. We interpret this fact as a necessity. DDoSaaSs are intended for technically unknowledgeable users. It is not reasonable to expect these users to be able to select a proper advanced attack type (e.g., a suitable application layer attack type). This also explains why slow-rate attacks such as RUDY or Slowloris are erroneously implemented as flooding types.

In advertisements, the attack power is usually presented in the range of 5 Gbps to
40 Gbps. However, this power is only a theoretical limit that is rarely reached and even so, it is shared among all concurrent attacks. The resulting average attack power is less than 1 Gbps as seen in Table 4.5.

4.7.4 Installation

We did not observe any attempts to spread malware to the victim server. Therefore, the installation phase of the kill chain has no associated action and is skipped.

4.7.5 Command and Control (C2)

Most attacks performed by DDoSaaS have a length of a couple of minutes at most (see Table 4.11). If an attacker wants to maintain a long-term attack against a target, he or she can execute attacks against the target repeatedly. Pauses between subsequent attacks could be used to alter the properties of the attack, for example as a reaction to countermeasures gradually introduced by the victim. However, the attacker is rather limited to changing the destination port and the attack type.

Even though the attacker is always notified about a successful attack initiation, most DDoSaaS do not support a constant availability monitoring of the victim (e.g., via the ping command or with SYN packets). Especially technically unsavvy DDoSaaS customers may therefore mistaken the attack initiation for an attack success.

4.7.6 Actions on objectives

As soon as the attacker submits the request to execute an attack, the request is transferred to the back-end flooding infrastructure. Wait times are very short. An attack is executed immediately after it was requested or with only a couple of seconds delay.

4.8 Summary

Over the years, DDoS-for-hire services have matured into user friendly services with a wide customer base that extends beyond technically savvy users. Main advancements are automated subscription activation, automated attack execution and support for anonymous payment methods such as Bitcoin.

The key findings of our research are as follows:

- Attacks generated by DDoSaaS are not overly powerful with bitrates only sporadically exceeding 1 Gbit/s.
4. DDoS-as-a-Service

- Attack traffic has a low complexity, does not employ randomization and shares similarities even between various DDoSaaSs.
- More than a third of attacks were not fully blocked by a cloud provider.
- DDoSaaSs are widely accessible and can be used easily and efficiently even by users with zero technical knowledge.

We believe that the threat of DDoSaaS will increase in time, mainly due to a low price, open advertisement, achievable anonymity and a service model that makes these services quickly and widely accessible to many potential customers. In the same time, the number of DDoSaaS services will grow, due to freely available source code and low initial entry costs when compared to potential earnings.
5 DoS attacks dataset

Designing experiments that evaluate the efficiency of a DDoS defense is a difficult process. Problems encountered are similar to those in experiments with botnet detection systems. Aviv et al. name basic problem sources: multiple administrative domains, heterogeneity of networks, privacy concerns and lack of ground truth [AH11]. An experiment must be designed with focus on realism, representativeness, generality, false positives/negatives, repeatability and comparability.

Suggestions for good DDoS defense experiments are presented by Hussain et al. in [HST+06]. An emphasis is put on a careful choice of attack tools and background traffic generators, on building a representative network topology and on a proper specification of conditions under which the defense is tested.

In order to support repeatability and comparability of our experiments, we created a labelled dataset for evaluation of DoS attack detection and mitigation systems called DDoS-Vault. The dataset is composed of packet capture (PCAP) files that store all the traffic which was observed during the data collection. The dataset includes PCAP files that were generated by standalone DoS tools for research work in Chapter 3 as well as attack traffic records from testing of DDoSaaS in Chapter 4. Moreover, the dataset is supplemented with scripts, standalone DoS tools binaries and queries that were used for analysis. Each file has an associated metadata that describe its content.

Our collected 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 recognizing the ground truth and precisely determining the false positives rate and false negative rate of the evaluated detection system [AH11].

In addition to the source network traffic, we share also processing scripts and search queries that were used during our analyses. Both the packet capture files and analysis files are freely available for use. All files are available for download at the DDoS-Vault dataset webpage [Buk15].

5.1 Existing DoS datasets

The sparsity of contemporary DoS attack datasets is one of the biggest limitations to comparability and repeatability of DoS experiments. Researchers tend to use old, well-
known datasets that no longer reflect a current state of denial of service attacks. If contemporary proprietary attack traces are used, they are rarely provided back to the research community. Moreover, existing publicly available datasets provide little variability in recorded attacks. Full packet capture is most frequently used form of sharing datasets.

CAIDA 2007 dataset contains approximately one hour long anonymized sample of a single flooding DDoS attack from August 4, 2007 (20:50:08 UTC to 21:56:16 UTC) [CAI07]. Only the attack traffic and victim responses are included in the sample. Legitimate traffic and background noise were removed in post-processing. The sample is provided as a set of PCAP files, each of which contains 5 minutes of attack. The payload has been removed from all packets. The total size of collected PCAP files is approximately 21 GB.

DARPA 1999 is a labeled dataset that was created for evaluation of intrusion detection systems [DAR99]. Among other audit sources, the dataset contains packet captures of both incoming and outgoing network traffic. The datasets spans over five weeks, with first, second and third weeks intended as training data and fourth and fifth weeks serving as testing data. The network traces alone have 20.6 GB. There were 63 DoS attacks of the following types in the test data: Apache2, arppoison, Back, CrashIsis, dosnuke, Land, Mailbomb, SYN Flood (Neptune), Ping of Death, Process Table, selfping, Smurf, sshprocessstable, Syslogd, tcpreset, teardrop, Udpstorm.

KDD Cup 1999 dataset was created for The Third International Knowledge Discovery and Data Mining Tools Competition [Lic13]. The aim of the competition was to build an intrusion detection system that can successfully distinguish good and bad network connections. The raw training data was 4 gigabytes of dumped network traffic over the span of seven weeks. The raw training data was processed into approximately five million connection records, stored as text records about 100 bytes each. Additional two million records were added as the testing data. The total size of sample is 743 MB. DoS attacks included in the dataset are: Back, Land, Neptune, Pod, Smurf, Teardrop.

Santanna et al. made available the dataset that was used in analysis of DDoSaaSs in [SvRDS +15, SvRDH +15]. The dataset includes traces of 9 DoS attacks generated by DDoS-for-hire services. Of those, 7 attacks are DNS-based and 2 attacks are CHARGEN-based. The total size of collected traces is more than 250 GB.

The ANT lab lists several DoS attacks dataset, which were provided by USC/ISI ANT project [AL]. A notable example is a SYN flood attack, which was a part of 2009 DARPA Scalable Network Monitoring (SNM) Program Traffic [DAR13]. Another example is a dataset of a DNS amplification attack, which was staged by security researchers between USC/ISI, Marina del Rey, California and CSU, Fort Collins, Colorado [DNS14]. Traces are provided as packet captures and usually last several minutes.
5. DoS attacks dataset

5.2 Standalone DoS tools traces

At the time of writing of this dissertation, DDoS-Vault contains 215 PCAP files that contain DoS attack traffic generated by standalone DoS tools. Of those, 116 files were analyzed in Chapter 3. Another 86 files were generated with desktop standalone DoS tools. The attack traffic in remaining 13 files was generated by standalone DoS tools for Android mobile platform. The collection of mobile applications attack traffic was performed on a standard laptop via WiFi network interface. The traces that were not analyzed in Chapter 3 are provided “as is” without any details about the network traffic.

The trace files were created by dumpcap tool from the Wireshark suite [Wt]. The traffic was collected directly at the attacking machine, consistently with our focus on the source-end DoS attack detection. The data collections were performed in a virtual environment 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.

The topology consists of two virtual machines: the attacker virtual machine (VM) and the victim VM. We created a simple point-to-point virtual network between the 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 3.3.1, 3.3.2 and 3.3.3) 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 3.3.6, 3.3.7 and 3.3.9). 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. The saved webpage consists of 109 files and the total size is 3.3 MB including images.

Metadata associated with each trace contains an ID, timestamps of the first and the last packet in the trace, the name of a standalone DoS tool that generated the traffic, configuration of this tool and any other possibly interesting information. An example is provided below:

<table>
<thead>
<tr>
<th>ID</th>
<th>BAD-G-2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tool</td>
<td>BanglaDos</td>
</tr>
<tr>
<td>First</td>
<td>2013-11-01 12:09:57.401</td>
</tr>
<tr>
<td>Last</td>
<td>2013-11-01 12:10:56.222</td>
</tr>
<tr>
<td>Attack type</td>
<td>Slow headers</td>
</tr>
<tr>
<td>URL</td>
<td><a href="http://192.168.1.20/index.htm">http://192.168.1.20/index.htm</a></td>
</tr>
<tr>
<td>Proxy</td>
<td>&lt;empty&gt;</td>
</tr>
<tr>
<td>Connections</td>
<td>400</td>
</tr>
<tr>
<td>Timeout</td>
<td>60 s</td>
</tr>
<tr>
<td>Random</td>
<td>True</td>
</tr>
<tr>
<td>User agent</td>
<td>OWASP DDoS</td>
</tr>
<tr>
<td>Diagnostics</td>
<td>False</td>
</tr>
<tr>
<td>Use POST</td>
<td>True</td>
</tr>
</tbody>
</table>

The ID comprises of a tool ID, an attack identifier an identifier of a sample with the chosen configuration. Attack type identifiers are listed in Table 5.1. Tool IDs are available in Tables 3.1 and 5.2.

### 5.3 Standalone DoS tools

The full list of standalone DoS tools, that were used to generate DoS traffic traces for Chapter 3 is provided in Table 3.1. Other desktop tools for which we collected attack traffic traces are listed in Table 5.2.

Each tool has an associated metadata record. We list the tool name, version (if known), ID, MD5 hash of the main executable/file, an information whether the tool is a script or a standalone executable, whether administrative rights are necessary for launching the
### 5. DoS attacks dataset

Table 5.1: Attack types and identifiers.

<table>
<thead>
<tr>
<th>Class</th>
<th>ID</th>
<th>Attack type</th>
<th>Variants</th>
</tr>
</thead>
<tbody>
<tr>
<td>HTTP</td>
<td>ARME</td>
<td>Apache vulnerability DoS</td>
<td></td>
</tr>
<tr>
<td>UDP</td>
<td>CHARGEN</td>
<td>CHARGEN reflection attack</td>
<td></td>
</tr>
<tr>
<td>UDP</td>
<td>DNS</td>
<td>DNS flooding attack</td>
<td></td>
</tr>
<tr>
<td>HTTP</td>
<td>GET</td>
<td>HTTP GET flood</td>
<td>GHPGET, SRCGET</td>
</tr>
<tr>
<td>HTTP</td>
<td>HEAD</td>
<td>HTTP HEAD flood</td>
<td>ARME, GHPHEAD, SRCHEAD</td>
</tr>
<tr>
<td>UDP</td>
<td>NTP</td>
<td>NTP reflection attack</td>
<td></td>
</tr>
<tr>
<td>UDP</td>
<td>NUDP</td>
<td>NUDP reflection attack</td>
<td></td>
</tr>
<tr>
<td>HTTP</td>
<td>POST</td>
<td>HTTP POST flood</td>
<td>GHPPOST, SRCPOST</td>
</tr>
<tr>
<td>HTTP</td>
<td>REC</td>
<td>ReCoil attack</td>
<td></td>
</tr>
<tr>
<td>HTTP</td>
<td>RUDY</td>
<td>HTTP slow RUDY attack</td>
<td></td>
</tr>
<tr>
<td>HTTP</td>
<td>SLOW</td>
<td>HTTP slow Slowloris attack</td>
<td></td>
</tr>
<tr>
<td>UDP</td>
<td>SNMP</td>
<td>SNMP reflection attack</td>
<td></td>
</tr>
<tr>
<td>UDP</td>
<td>SSDP</td>
<td>SSDP reflection attack</td>
<td></td>
</tr>
<tr>
<td>UDP</td>
<td>SUDP</td>
<td>SUDP reflection attack</td>
<td></td>
</tr>
<tr>
<td>TCP</td>
<td>SYN</td>
<td>TCP SYN flooding</td>
<td>ESSYN, SSYN, SYNACK, XSSYN, XSYN, XYN</td>
</tr>
<tr>
<td>TCP</td>
<td>TCP</td>
<td>TCP-based attack</td>
<td>TCPACK, TCPAMP, TCPFIN, TCPNO, TCPPSH, TCPPTK, TCPRES2, TCPRND, TCPRST, TCPSEQ, TCPURG</td>
</tr>
<tr>
<td>UDP</td>
<td>UDP</td>
<td>UDP flooding attack</td>
<td></td>
</tr>
<tr>
<td>UDP</td>
<td>UDPLAG</td>
<td>UDP lag attack</td>
<td></td>
</tr>
<tr>
<td>HTTP</td>
<td>XML</td>
<td>Vulnerability DoS</td>
<td></td>
</tr>
</tbody>
</table>

Table 5.2: Other standalone DoS tools with collected attack traffic samples.

<table>
<thead>
<tr>
<th>Desktop</th>
<th>Android</th>
</tr>
</thead>
<tbody>
<tr>
<td>Assault v1.0</td>
<td>AnDOSid</td>
</tr>
<tr>
<td>DecFlooder 1.0</td>
<td>LOIC</td>
</tr>
<tr>
<td>Razor’s DoS Tool</td>
<td>NetSwiss</td>
</tr>
<tr>
<td>UDP Test Tool</td>
<td>PingFree</td>
</tr>
</tbody>
</table>
attack and any other interesting information. Below is an example of a metadata record that is associated with a DoS tool:

Name = LOIC
Version = 1.1.2.0b
ID = LO4
Author = NewEraCracker (based on Praetox code)
URL = https://github.com/NewEraCracker/LOIC/
MD5 = 976104ade0e9e67a275ae4a5ea58ece9
Type = binary executable
Supported attacks = TCP, UDP, HTTP, ReCoil, slowLOIC
Admin rights required = False

Standalone DoS attack tools rarely support command-line interface. Therefore, in order to generate traffic samples a constant presence of a human operator is necessary. Unfortunately, such manipulation is not acceptable for situations when attack traffic and background traffic are created simultaneously, when multiple attacks from multiple sources are tested or when a large number of attacks with various properties and lengths are repeatedly executed. Moreover, such experiment design does not scale for DoS experiments performed on common testbeds.

Macros simulating a clicking user may be used to address this challenge. AutoHotkey is an utility and a scripting language which allows to automate repetitive tasks in Windows environment [MTC]. We provide files with saved macros that can be used to automate the work with standalone DoS tools.

Below is an example script for running a DoS attack against www.example.edu domain for 60 seconds with HTTPFlooder tool:

```
// Attack configuration

// Attack execution

Run, %httpflooder_path%
```
5. DoS attacks dataset

5.4 DDoSaaS traces

We have included all 272 DDoSaaS attack traffic traces that were analysed in Chapter 4. Other traces from similar or new DDoS-for-hire services are added continuously.

DDoS attack traces were recorded at a virtual victim server which was hosted on the Amazon Elastic Compute Cloud (EC2) [Ama]. The virtual machine was configured with 4 virtual CPUs running on Intel Xeon core, 15 GB RAM and SSD storage. The server was connected with at least 1 Gbit/s line. The operating system was Ubuntu Trusty 14.04. The server was hosting a dummy webpage that imitated a webpage of a gaming clan. Packet captures in PCAP format were collected with tcpdump command-line packet analyzer [TG]. First 1500 bytes were stored for each packet.

Only one attack was recorded at a time. The traffic record was started just prior to launching an attack on DDoSaaS webpage. Each attack was requested to be 30 seconds long. The recording was stopped 300 seconds after it was started. All network traffic except Secure Shell (SSH) to port 22 was recorded. However, no background traffic was knowingly generated, except basic communication native to the chosen operating system and environment.

Each trace has an associated metadata record. An example is below:

ID = ANON-NTP-02
DDoSaaS = anonymous-stresser.com
First = 2014-12-23 23:01:21.286
Last = 2014-12-23 23:06:18.949
Requested attack = NTP
Length = 30 s

Unique trace ID consists of a DDoSaaS identifier, attack type identifier and an identifier of a sample with the chosen configuration. Attack type identifiers are listed in Table 5.1. DDoS-for-hire service identifiers are available in Table 5.3.
Table 5.3: DDoS-for-hire service identifiers.

<table>
<thead>
<tr>
<th>ID</th>
<th>Booter domain name</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANON</td>
<td>anonymous-stresser.net</td>
</tr>
<tr>
<td>BOOTIO</td>
<td>booter.io</td>
</tr>
<tr>
<td>CONS</td>
<td>connectionstresser.com</td>
</tr>
<tr>
<td>CSTR</td>
<td>cstress.net</td>
</tr>
<tr>
<td>DEST</td>
<td>destressnetworks.com</td>
</tr>
<tr>
<td>HORNY</td>
<td>hornystress.me</td>
</tr>
<tr>
<td>IPSTR</td>
<td>ipstresser.com</td>
</tr>
<tr>
<td>ISIT</td>
<td>isitdownyet.com</td>
</tr>
<tr>
<td>KRYPT</td>
<td>kryptonic.pw</td>
</tr>
<tr>
<td>LEGION</td>
<td>legion.cm</td>
</tr>
<tr>
<td>NETCOM</td>
<td>networkstresser.com</td>
</tr>
<tr>
<td>NETNET</td>
<td>networkstresser.net</td>
</tr>
<tr>
<td>POWER</td>
<td>powerstresser.com</td>
</tr>
<tr>
<td>QUANT</td>
<td>quantumbooter.net</td>
</tr>
<tr>
<td>RAGE</td>
<td>ragebooter.com</td>
</tr>
<tr>
<td>REST</td>
<td>restricted-stresser.info</td>
</tr>
<tr>
<td>TITAN</td>
<td>titaniumstresser.net</td>
</tr>
<tr>
<td>VBOOT</td>
<td>vbooter.org</td>
</tr>
<tr>
<td>VDOS</td>
<td>vdos-s.com</td>
</tr>
</tbody>
</table>

5.5 Summary

We have created a dataset of denial of service attack traffic traces called DDoS-Vault. The collected traces come from our work with standalone DoS tools and DDoS-for-hire services. Metadata associated with trace contains information such as unique identifiers, timestamps or various information about attack traffic source.

The key advantages of our dataset compared to other available datasets:

- Our attack traces contain a wide variety of attacks that are still used in the wild.
- The traces do not contain any mentionable background traffic, which makes them an ideal data source for widely used data overlay methodology.
- Each sample is labelled and has an associated metadata about the source.
The dataset is freely available for use by network security community [Buk15], in order to facilitate the usage of contemporary input data in DoS attacks research. To support repeatability and comparability of our experiments, we also released scripts that were used for analyses in our own experiments.
6 Framework for evolution of DoS attacks

We have proposed and designed a prototype of a novel generic framework for quick automated evolution of denial-of-service (DoS) attacks in a virtual environment. The framework applies genetic algorithms to existing DoS attacks in order to discover advanced, more potent attack variants and also to identify DoS vulnerabilities in applications that are serving as targets. The framework is sufficiently universal to be used for evaluation of an arbitrary denial-of-service attack. Our framework architecture and initial experiment results were published in [BOS+15]. The framework is still under development with Radim Ostadal as the main maintainer.

The framework has been initially created to examine possible enhancements to the HTTP GET flooding attack by modifying HTTP request headers. The project aim was to search for such HTTP GET headers where their processing by the victim server would be significantly more resource demanding than the processing of HTTP GET headers from common web browsers (see Chapter 6.4.1). Later, we analyzed the resistency of implementations of SSL protocol in web browsers against a new SSL slow handshake DoS attack. To our knowledge, this attack was not published yet (see Chapter 6.4.2). The research into vulnerability of web servers against SSL slow handshake DoS attack is still in progress.

The modular architecture enables seamless changes. Employed genetic algorithm, virtual machine operating system and target application can all be changed with minimal impact on the other parts of the framework. Once the task and its properties are fixed, the framework can be left to produce relevant results automatically.

Prototype implementation of the framework is developed in the Python programming language [PSF]. Our initial experiments were conducted with common web servers. VirtualBox was chosen as virtual environment [OC] and Ubuntu as both guest and host operating system.

During the testing, we have encountered numerous unexpected issues with virtual environments. These issues made us question the reliability of experiment results from virtual environments and their comparability to experiment results from physical environment. The issues are briefly discussed in Chapter 6.5.

6.1 Genetic algorithms

Genetic algorithms (GAs) are search heuristics that are inspired by the process of natural selection. They belong among the Evolutionary Algorithms. Genetic algorithms are mostly used to solve optimization problems and search problems.
Mitchell characterizes GA’s work as “discovering, emphasizing, and recombining good “building blocks” of solutions in a highly parallel fashion. The idea here is that good solutions tend to be made up of good building blocks – combinations of bit values that confer higher fitness on the strings in which they are present.” [Mit98].

Every genetic algorithm has at least the following elements: populations, selection according to fitness, crossover to produce a new offspring and random mutations of offsprings.

6.2 Architecture

The framework architecture consists of a central management host and multiple physical computation hosts for conducting experiments themselves. Each computation host has a hypervisor installed and hosts two virtual machines – attacker and victim. Each management unit can assign tasks to multiple computation hosts, therefore each generation can be evaluated on dozens of physical hosts simultaneously. Virtual machines (VMs) on different physical hosts are clones of attacker and victim initial source VM images. Thanks to snapshot restoration, each evaluation is performed in exactly the same virtual machine state. The architecture is outlined in Figure 6.1.

6.3 Workflow

Each assigned task is processed in several rounds. During each round a set of candidate solutions is evaluated on available virtual machines. The quality of solutions is derived from the measurements collected on a victim virtual machine. Precise order of steps is as follows:

- **A** – Management module initiates evaluation of a new population.
- **B1, B2** – Genetic algorithms module creates a list of candidate solutions to be evaluated in the current round and provides the list to the management module. Each candidate solution provides a representation of network traffic that needs to be evaluated.
- **C1** – Management module maintains a list of active computational hosts. Candidate solutions are distributed equally to all available computation hosts in order to minimize the time required to evaluate the entire population.
6. Framework for evolution of DoS attacks

Figure 6.1: Framework architecture.

- **C2** – Management module restores snapshots of all virtual machines on computation hosts. Restoring snapshots is quick and establishes a common initial state for evaluation of every candidate solution.

- **D** – Attacker virtual machine contains a network traffic generator that can translate the received specification (i.e., candidate solution) into an arbitrary network traffic. The actual generated stream of packets is sent towards the victim virtual machine.

- **E** – Monitoring tools on the victim VM measure the impact of received traffic on the host (e.g., consumption of RAM, CPU load or values of application-specific performance counters). Measured values are converted into a common format and sent as fitness function values (how well candidate solution satisfies the goal – high load in our case) to the genetic algorithms module.

- **F** – Genetic algorithms module evaluates all received fitness function values, then it chooses the best solution(s) and provides results to the management module for a manual review. Once enough results are received, management module starts a new round.
6. Framework for evolution of DoS attacks

6.4 Scenarios

6.4.1 HTTP requests

As mentioned before, our original goal was to search for such HTTP GET headers that could generate a burden on the victim server with significantly more resource demanding load than would be that of a processing of HTTP GET headers from common web browsers.

A candidate solution is an ordered list of pairs (HTTP header field, value). Candidate solutions differ in the chosen header fields, appropriate values and the order of pairs in the header. Each candidate solution is incorporated into a HTTP GET request with a constant URL before being sent. The URL targets a copy of a well-known news webpage which is running on the victim VM. Each request is sent 10,000 times. Monitoring tools on the victim VM collect CPU time of all Apache processes.

Measured values were afterwards compared to CPU time of HTTP requests that were constructed to mimic requests from common web browsers (i.e., Google Chrome 35, Internet Explorer 11 and Mozilla Firefox 31).

Figure 6.2 illustrates some of our findings. Random HTTP headers represent distribution of CPU consumption of 200 randomly constructed HTTP request headers. Cluster between 0 and 10 represents malformed requests that are responded with 400 error code. Cluster between 45 and 60 represents standard common requests. Best1, Best2 and Best3 show consistency of measurements of 200 iterations of three most demanding requests that we were able to construct. Measurement precision is sufficient even for a fine-grained evolution. IE baseline represents consistency of measurements of 200 iterations from common Internet Explorer 11 request.

Contrary to our hypothesis, we were unable to find a sequence of HTTP header fields and respective values whose CPU requirements would be significantly higher than computational requirements of standard browser requests. Apparently, the impact of HTTP header fields processing on a standard Apache webserver is negligible, with the exception of Accept-Encoding field. Accept-Encoding field value can significantly increase CPU consumption when zip compression is required. However, such behavior is a default for common HTTP requests.

6.4.2 Slow attacks

Although HTTP requests research project was ultimately unsuccessful, the framework proved to be both simple and effective.

As a next step, we are adapting the framework for searching for slow DoS attack
opportunities in common protocols. An inherent property of most network protocols is to proceed with the next phase of the protocol only when previous phase was completed. Meanwhile, each side has to allocate its computational resources for any (half-)open connection. Under normal circumstances, network connections are closed only when they are no longer used, either explicitly with a close message or when an inactivity timeout expires.

For example, a webserver can only send response when a full HTTP request has been received. The Slowloris attack exploits this behavior by sending a never-ending HTTP header [Han09]. Therefore, the request is never finished and connection socket is effectively and indefinitely blocked for other users. If the attacker is able to maintain a sufficient number of simultaneously opened connections, legitimate users cannot reach the webserver.

We discovered a new resource exhaustion slow attack against Secure Sockets Layer (SSL) protocol. The attacker opens a Layer 4 TCP socket with the victim server. Then the attacker separates SSL handshake messages into multiple parts and introduces artificial delays between each two messages. The attacker’s goal is to keep as many simultane-
ously open sockets as possible, which results in exhaustion of the victim server buffer. If the SSL implementation of a target application does not enforce global SSL handshake timeout, the attacker can keep a socket open for an indefinite amount of time, just by sending short “keep-alive” messages in regular intervals.

We employed genetic algorithms to identify maximal possible time gaps between messages of SSL handshake. Each evaluation of a candidate starts with sending Client Hello message towards the server. Server responds accordingly with Server Hello message, followed by Server Certificate message. At this point, we introduce an artificial delay between the receiving Server Hello message and sending the next message in handshake. Similarly, we introduce delays between each subsequent pairs of exchanged messages. Genetic algorithms are used to identify the maximum duration of SSL handshake, that can be created with various combinations of delays. Genetic algorithms module may also split existing messages into several parts, creating many more new opportunities for introducing delays. Candidate solutions with long total SSL handshake durations are promoted, while candidate solutions with limited total SSL handshake durations are discarded.

Using genetic algorithms allows us to automatically discover and overcome general time limitations (e.g., maximal interval between two subsequent messages or between two subsequent parts of a same message), specific time limitations (e.g., maximal interval for checking a certificate validity) and any their combinations and overlaps (e.g., a Client Certificate message must be sent within a specified time frame after Client Hello message, while no two messages may be more than a specified time interval apart).

The attack was effective against a subset of tested applications. Some tested webservers enforce global SSL handshake timeouts, usually less than 2 minutes long. Oppositely, vulnerable servers enforce usually only a timeout between two subsequent messages, which is reset after each new segments. Such webservers are vulnerable by our SSL slow handshake DoS attack. Table 6.1 provides examples of vulnerable webservers. Results were collected with key contributions by Radim Ostadal and Tatevik Baghdasaryan.

6.5 Virtualization issues

It has been known for a long time that network simulations do not reflect behavior in real networks correctly, especially when considering high-volume traffic. Research on denial-of-service attacks has been badly affected, because simulations assume ideal environment without limitations of the physical world (e.g., sizes of buffers on routers) [CFS06]. Emulation testbeds [MBF+10] and hybrid physical-virtual testbeds [SST+10] have been built to support experimentation with large-scale attacks as a response.
6. Framework for evolution of DoS attacks

Table 6.1: Webservers vulnerable to SSL slow handshake DoS attack.

<table>
<thead>
<tr>
<th>Webserver</th>
<th>Version</th>
<th>Vulnerable?</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Apache</td>
<td>2.2.22</td>
<td>False</td>
<td>Global SSL handshake timeout</td>
</tr>
<tr>
<td>Apache</td>
<td>2.2.29</td>
<td>True</td>
<td>Consecutive TCP segments timeout only</td>
</tr>
<tr>
<td>IIS</td>
<td>7.0</td>
<td>False</td>
<td>Global SSL handshake timeout</td>
</tr>
<tr>
<td>lighttpd</td>
<td>1.4.33</td>
<td>True</td>
<td>Consecutive TCP segments timeout only</td>
</tr>
<tr>
<td>nginx</td>
<td>1.4.6</td>
<td>False</td>
<td>Global SSL handshake timeout</td>
</tr>
<tr>
<td>openlitespeed</td>
<td>1.3.8</td>
<td>True</td>
<td>Consecutive TCP segments timeout only</td>
</tr>
</tbody>
</table>

Virtualization has a significant impact on how network security experiments are performed. It allows for a high flexibility in both experiment design and scope setting, and it also supports experiment repeatability with quick restoration of predefined state. Virtualization enables a great utilization of available resources with a high scalability. Yet are the environments that were built over standard hypervisors (e.g., Xen, VMware, Hyper-V) truly representative? Are experiment results obtained in virtual environment applicable also in physical environment? During our research, we have encountered issues that could have a huge impact on the results collected in virtual or hybrid environments. We would like to initiate a discussion about issues that could change our perception of virtual environment as a helpful servant.

We have encountered a number of problems to cope with:

- **Measurement precision deviation.** While the deviation of multiple measurements of the same phenomena was less than 3% with physical hosts, the deviation increased up to 20% when similar measurements were conducted in the virtual environment.

- **No relation of results in the virtual environment and physical machines.** We were unable to replicate some results from the virtual environment on real hardware. The example is Figure 6.2 with results from the virtual environment. In the virtual environment, Best 1, 2 and 3 requests require higher CPU load than the common Internet Explorer request. When we sent the same HTTP headers on two separate physical machines, the difference between them and IE was negligible.

- **Different interpretations on physical machine and in virtualization.** We observed that the same version of Wireshark on the same version of operating system interprets the same network traffic differently, when running on real HW and when running in a virtualized environment. This behavior could influence any automated analysis of PCAP files that is based on the libpcap library.
6. Framework for evolution of DoS attacks

- **Incomparable performance from hosts with different hardware configuration.** Virtual machine performance is heavily influenced by underlying physical hardware. Two virtual machines running on a different hardware will provide different measurement, even though the environment seems to be exactly the same when observed from inside. All candidate solutions must be evaluated on computational hosts with similar HW configurations. This presents a significant challenge for comparability of any cloud-based computations.

- **Cable vs. Wi-Fi connection.** We also applied a variant where attacker VM and victim VM resided on separate physical hosts. Operating system performance counters values were distinctively different when attack traffic was sent via Wi-Fi and via cable connection.

- **Lower precision bound.** When using virtualized environment, there is always background noise (e.g., fluctuations of CPU load, OS native network traffic, RAM consumption varying in time). This noise sets a lower bound for useable precision of measurements. With less than 1,000 HTTP requests during each run, the noise was too dominant for measurements to have any real informational value. We therefore used at least 10,000 HTTP requests. Noise in physical environment is arguably lower.

- **Results interpretation.** Sometimes it was difficult to identify what parameters were key influencers of final results (e.g., VM configuration, physical host properties, network configuration, and internal application configuration). We had to employ try-error approach to interpret some of the observed anomalies. Also, it was helpful to collect fitness values for minimal size HTTP requests and then use these fitness values as guidance for mutual comparison of more complex headers.

6.6 Summary

We have proposed and designed a prototype of a novel framework for automated denial of service attack testing in virtual environments, based on genetic algorithms. We provide a description of the framework architecture as well as a step-by-step workflow.

The framework can be used for:

- Automatic mutation of existing DoS attacks in order to discover new, more potent variants.
6. Framework for evolution of DoS attacks

- Assessing the vulnerability of existing network applications to arbitrary types of flooding DoS attacks.

The framework was employed in two scenarios. In the first scenario, we attempted to find a new HTTP GET flooding attack variant with higher impact due to a specially crafted HTTP header. In the second scenario, we proposed a new slow DoS attack that is based on introducing delays into the SSL handshake. Then we evaluated the vulnerability of common web servers to this new attack.

The framework has shown potentially serious discrepancies between virtual and physical environments. We initiated a discussion on hidden caveats of experimenting in virtual environment testbeds.
7 Conclusions and future work

Denial of service attacks are constantly evolving, which calls for a continuous development of DoS attack mitigation systems. In this thesis, we focused on the traffic properties of contemporary DoS attacks with low to medium attack power, as well as the ecosystem they fit in, because we believe that is an area which have received less attention than it deserves.

In order to design an effective DoS attack mitigation system that can be used in practice, the knowledge of attack traffic features is imperative. Security researchers need to access real attack samples and they need to have an overview of the features that can be used as an input in their detection systems. To address this challenge, we collected and analyzed numerous DoS attack traffic samples from widely used standalone DoS tools and from commercial DDoS-for-hire services. All samples were subsequently made freely available for evaluation of existing DoS detection systems as well as the development of new systems. In order to set contemporary DoS attacks into a context, we also analyzed the ecosystem of DDoS-for-hire services, their leaked databases and leaked source codes. We have presented the anatomy of a typical attack that is generated by a DDoS-for-hire service on the basis of a well-known kill chain methodology. Finally, we have designed a framework that can be used both to automatically enhance existing DoS attacks in order to find their limits and to discover DoS-related vulnerabilities in existing network applications.

In Chapter 2 we explored the state-of-the-art research on network monitoring host-based intrusion detection systems. We surveyed standalone HIDSs, collaborative HIDSs and cloud-based HIDSs. We perceive network monitoring HIDSs as a complementary part of an overall intrusion detection architecture. Their greatest benefits are the capability to provide rich semantic information about attack vectors and useability in otherwise untrustworthy environments. Current research on HIDSs follows two main directions, the utilization of virtualization technologies and the shift towards a real-time detection. The real-time detection is a necessary prerequisite to creating a dependable host-based intrusion protection system. Virtualization has a great impact on HIDS tamper resistance, simplification of HIDS management and detection capabilities. An ongoing challenge is how to mask the virtualization layer from malware.

In Chapter 3 we collected and analyzed attack traffic samples generated by popular contemporary standalone DoS tools. We selected key attack traffic features and provided a classification of the traffic by these features. We discussed the importance of any observed traffic anomalies. Our analysis shows the existence of repeating patterns in all analyzed samples. We proposed a new research area for DoS detection that is
7. Conclusions and future work

Based on the recognition of these repeating patterns. We also analyzed used detection evasion techniques. We note that randomization of attack traffic is rare, possibly due to its impact on attack effectiveness.

In Chapter 4 we focused on the ecosystem of DDoS-for-hire services. We analyzed their webpages, their attack traffic, content of leaked databases and leaked source codes. We established a server in a cloud, bought subscriptions at 16 DDoSaaS services and launched attacks against this server. The collected traces showed that attacks generated by DDoSaaSs only rarely exceed 1 Gbit/s bandwidth, have low complexity, rarely use randomization and frequently use amplification. We also identified similarities between attacks from different services. The analysis of databases showed that DDoSaaSs are used mainly for short UDP-based attacks, mostly against webservers. Moreover, DDoSaaSs are useable for a wide audience, mostly due to low subscription prices, accessibility through mainstream search engines and user-friendliness.

In Chapter 5 we presented our DoS attacks dataset. The dataset comprises of traces collected both from standalone DoS tools and DDoS-for-hire services. Contrary to existing DoS attack datasets, our dataset contains many traces with a wide variety of contemporary attack types. Moreover, the traces are labelled and do not include any background noise traffic. Each trace has an associated metadata record with basic information about the trace, such as the type of attack, source or timestamp when the trace was acquired. The dataset is available for free use by the network security community.

In Chapter 6 we presented a novel framework for automated denial of service attack testing. The framework can be used either for discovering new potent DoS attack variants or for vulnerability assessment of existing network applications against DoS attacks. We presented two scenarios where the framework was used, the enhancement of an existing HTTP GET flooding attack and a vulnerability assessment of common web servers against a novel type of slow SSL handshake DoS attack. We also briefly discussed the challenges of comparing experiment results between a virtual environment and a physical environment.

7.1 Future work

We have identified several areas worth a future investigation:

- **Source-end view on amplification attacks.** Amplification attacks are unique by their involvement of poorly configured amplifiers. The popularity of amplification attacks is steadily growing due to a high potential attack bandwidth and difficult traceback. Our studies about standalone DoS tools can be complemented
by including amplification attacks, both from the perspective of attacking hosts and amplifiers.

- **Impact of attack traffic randomization on attack effectiveness.** Randomization of attack traffic properties is frequently listed as one of major limitations to successful use of signature-based DoS attack detection systems. However, we rarely see any kind of randomization in real attacks. One of possible explanations is that a randomization of key attack features decreases the attack effectiveness (e.g., using request code other than MON_GETLIST in NTP amplification attack decreases the amplification factor). Identification of these features would be a valuable contribution.

- **Intuitively understandable language for DoS attack description.** Sharing information about DoS attack properties is limited by the lack of formalized, yet intuitively understandable format. We plan to create a simple description language that would express basic properties of DoS attacks, such as flow counts, flow characteristics and repeated operations characteristics.

- **Analysis of DDoSaaS backend infrastructure.** We have identified similarities in attacks from multiple DDoSaaSs, as well as similarities in their source code. This indicates that there may be a middleware service (booster API) that provides denial of service attack capability to multiple DDoSaaSs. Identifying and analysing booster APIs will give another view on the existing DDoSaaS ecosystem.

- **More scenarios for DoS attack evolution framework.** There are many scenarios where our DoS attack evolution framework could be used. To name a few: (1) slow HTTP attacks optimization, (2) finding optimal balance between packet lengths, flow counts and packets per flow count in order to find maximum effectiveness of UDP flooding attacks with regards to different targeted resources (e.g., RAM, CPU time, router buffers), (3) web form submissions fuzzing in order to maximize the response time of backend database.
Bibliography


BIBLIOGRAPHY


## A Standalone DoS tools traffic features

Table A.1: Summary of standalone DoS tools traffic features – TCP-based features.

<table>
<thead>
<tr>
<th>Traffic burstiness</th>
<th>HDT, HU, LO4, SL, SF</th>
<th>BD, HO, LO, LC, UD</th>
<th>AD, BAD, DS, GB, HDT, TH</th>
<th>ADR, FF, HF, JA, JL, LO, LC, SD, UD, XO</th>
</tr>
</thead>
<tbody>
<tr>
<td>Full burstiness</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Regular peaks</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>One-time extreme</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>None</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Stability</td>
<td>AD, ADR, BAD, BD, FF, HF, HO, HDT, JA, JL, LC, LO, SD, XO</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Flow count change</td>
<td>HDT, HU, JL, LO, SF, SL, UD</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pulsing</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Decreasing count</td>
<td>DS, GB, HDT</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Flow parallelity</td>
<td>AD, BAD, LC, SF, SL</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>All simultaneous</td>
<td>DS, GB, HDT, HU, LO4, TH, UD</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mostly simultaneous</td>
<td>LO</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Long-term consecutive</td>
<td>ADR, BD, FF, HF, HO, JA, JL, UD, XO</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mostly consecutive</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Average packet size with limit – TCP configurations</td>
<td><strong>Average packet size =&lt; 64 B</strong></td>
<td>BD, DS, JL, LC, SD, SF</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Average packet size &gt; 64 B</strong></td>
<td>JL, LO1, LO2, LO3, LO4, XO</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Average packet size with limit – HTTP configurations</td>
<td><strong>Average packet size =&lt; 64 B</strong></td>
<td>LO4, SL, TH</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Average packet size &gt; 64 B</strong></td>
<td>ADR, AD, BAD, FF, GB3, GB5, HDT, HO, HF, HU, JA, JL, LO1, LO2, LO3, LO4, LC, UD</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Flow packet count distribution</td>
<td>All flows the same packet count</td>
<td>ADR, BD, DS, FF, GB, HDT, JA, JL, LC, LO1, SD, SF, SL, UD, XO</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Minimal differences</td>
<td>LO, SD, UD</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Significant differences</td>
<td>HDT, HO, HU, JL, LO, SD, TH, UD</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Table A.2: Summary of standalone DoS tools traffic features – HTTP-based features.

<table>
<thead>
<tr>
<th>HTTP requests per flow</th>
<th>One per flow</th>
<th>Infinite per flow</th>
</tr>
</thead>
<tbody>
<tr>
<td>AD, FF, GB, HDT, HF, HO, JA, JL, LO, SL, TH, UD</td>
<td>AD, BAD, LC</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>HTTP request URIs</th>
<th>URI string set</th>
<th>Parameter change</th>
<th>Random URI</th>
</tr>
</thead>
<tbody>
<tr>
<td>AD, FF, GB, HDT, HF, HO, JA, JL, LO, LC, SL, TH, UD</td>
<td>AD, BAD, HU</td>
<td>JL, UD</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>HTTP requests User-Agents</th>
<th>User-agent field missing</th>
<th>User-agent field static</th>
<th>User-agent field dynamic</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADR, GB3, GB5, HO, HF, JL, LO1, LO2, LC, UD</td>
<td>AD, BAD, FF, HDT, JA, LO3, LO4, SL</td>
<td>HU, TH, UD</td>
<td></td>
</tr>
</tbody>
</table>
B List of author’s publications

B.1 Book chapter


B.2 International conferences and workshops

- Vít BUKAČ, Vlasta ŠŤAVOVÁ, Lukáš NĚMEC, Zdeněk ŘÍHA and Václav MATYÁŠ. Service in denial – clouds going with the winds. Accepted to 9th International Conference on Network and System Security, New York City, USA. 14 pages. 2015. [Main author, research team leader. Author contribution: 45%].

- Vít BUKAČ and Václav MATYÁŠ. Analyzing traffic features of common standalone DoS attack tools. Accepted to 5th International Conference on Security, Privacy, and Applied Cryptography Engineering, Jaipur, Rajasthan, India. Proceedings to be published by Springer in the LNCS series. 20 pages. 2015. [Main author. Author contribution: 90%]


- Vít BUKAČ, Pavel TUČEK and Martin DEUTSCH. Advances and Challenges in Standalone Host-Based Intrusion Detection Systems. In Lecture Notes in Computer Science 7449 : Proceedings of the 9th International Conference on Trust, Privacy and
B. List of author’s publications

B.3 National conferences and journals


- Vít BUKAČ. *Sledování komunikačních kanálů malwaru*. Data Security Management, TATE International, s.r.o. 5 pages. 2013. [Main author. Author contribution: 100%]

- Vít BUKAČ. *Záplavové aplikační útoky odepření služby*. Data Security Management, TATE International, s.r.o. 4 pages. 2012. [Main author. Author contribution: 100%]

B.4 Technical report