ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Institut für
Technische Informatik und
Kommunikationsnetze

Cyrill Bannwart

Predicting the Impact of Denial of Service Attacks

Master Thesis MA-2012-03
February 2012 to August 2012

Main advisor: Dr. Vincent Lenders
Advisor ETH: David Gugelmann
Supervisor: Prof. Dr. B. Plattner

Abstract

Denial-of-service (DoS) attacks have become a major threat to current systems and networks in the Internet. Yet the existing infrastructure is rarely tested for potential damage caused by (D)DoS attacks because no method exist to audit a productive system without flooding and thus risking system outages and resulting losses. Having a method that does (D)DoS attack auditing without requiring the observation of the system under massive load is however crucial when testing critical infrastructure in a productive environment.

We present a novel auditing method to externally assess and predict the impact of (D)DoS attacks on web servers based on low strength (D)DoS attack measurements. The developed method reduces the required probe traffic and relies on pre-established (D)DoS attack models to infer the impact of similar attacks at a stronger attack strength. To model the impact of (D)DoS attacks a multitude of server-internal as well as server-external metrics has been analyzed to identify those metrics, that characterize the state of the server and can be measured externally without requiring privileged access to the system and its network.

The presented auditing method is evaluated for multiple current and common (D)DoS attacks using extensive measurements, analyzing the influence caused by variations in the intermediate network as well as in software and hardware on the web server. Calculating the error rate between prediction and actual measurements the accuracy of the method is verified resulting in an expected error rate of 10% at a limited attack strength of 30% of the strength causing a DoS.

Additionally as a prototype an audit framework was developed, which implements the presented auditing method and allows anyone to asses the impact of a (D)DoS attack on a web server in the Internet.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Since the early days of the commercially used Internet, its systems and network infrastructures have always been the target of malicious parties [32]. While the number of connected hosts is ever increasing so do the number of observed attacks, their strength and also their level of sophistication [2, 86]. Internet criminality has become a well-organized and profitable business. Access to large Botnets can be rented by anyone [7], making it simple to send SPAM or launch distributed denial-of-service (DDoS) attacks against their competitors [15, 87].

In recent years the occurrence of malicious activities has received plenty of attention in the media. Being it the appearance of sophisticated computer malware such as Stuxnet, Duqu or Flame [60, 38, 37, 13] and the assumed involvement of governments and government-sponsored groups in espionage and sabotage [1] or also the increased occurrence of massive, coordinated DDoS attacks. Using tools that require little knowledge about computer- and network security allow any Internet user to take part in those attacks, to promote political ideas [76] or as revenge [14, 41].

The intentional participation in attacks carried out by arbitrary groups, which can often quickly organize many participants using social platforms, and the fact that little knowledge is required to perform a simple (D)DoS attack using readily available software, poses further threats to anyone providing services in the Internet. On these grounds, the interest in attack detection and defensive measures has been increasing [2]. However while attack and anomaly detection as well as the mitigation of attacks have become popular fields of research during the last decade, the impact analysis of attacks has received attention to a much lesser extent.

Yet, due to the cost of defensive measures, which according to a report of a security provider are at least $100’000 for the typically used hardware equipment and also require employees to maintain the systems around the clock [42], and the lack of ability to quantify the consequences of a successful attack, the adoption of mitigation solutions still remains low [2]. A further survey in the year 2011 showed that although 78% of the 225 questioned IT executives in the United States were concerned about (D)DoS attacks still 71% did not plan to implement a (D)DoS mitigation system over the course of the next year.

1.2 Goals

The goal of this master thesis is to develop and evaluate a novel approach to audit arbitrary Internet web server systems and to analyze and predict the impact of (D)DoS attacks. While current impact analysis techniques require the target system to be put under massive load to be observed, the method developed in this thesis will reduce the required load during the testing.

Having a method that does (D)DoS attack testing without requiring massive flooding is essential for system and computer administrators. Reducing the attack strength is key to probe critical and productive systems, which otherwise cannot be examined due to the performance degradation and unpredictable side effects on the whole environment. Unexpected issues may otherwise result in potential system and network outages and as such can cause significant yet avoidable losses.

System outages caused by (D)DoS attack auditing should by all means be avoided on productive systems, however any system should be capable to tolerate a short (D)DoS attack, run at a low intensity. As such this thesis aims to show a new approach for an impact analysis method that uses measurement data captured during low intensity attacks, to predict the progress and impact of (D)DoS attacks on the system.

Not only should the master thesis provide new insight on the feasibility and the potential of the suggested (D)DoS attack auditing method, but also result in an audit framework for (D)DoS attacks, similar to the existing Nessus vulnerability scanning program for system vulnerabilities [64].

1.3 Tasks

The master thesis consists of the following tasks:

- Reviewing available literature on (distributed) denial-of-service attacks and identifying related work that aims at predicting the impact of attacks on Internet web servers.
- Becoming acquainted with existing network as well as end-to-end measurement tools that will be used during the thesis.
- Developing a prediction model to externally asses the collateral damage or impact of denial-of-service attacks using probe and attack traffic sent from multiple client nodes.
- Implementing or evaluating existing tools to support the model and allow its practical application.
- Testing as well as evaluating the model and its required tools in a lab environment with varying network and system setups.

1.4 Outline

After this short introduction the remaining master thesis report is structured as follows:

In chapter 2, a formal problem statement will be presented outlining the expected requirements and the potential of the new analysis method.

Chapter 3 presents the related work that has been studied during the master thesis, where the attention during the literature research has mostly been turned on the existence of available (D)DoS impact metrics. For each publication that was analyzed and that contained a (D)DoS impact metric, a short summary of the parts involving the metric is presented.

The methodology of the master thesis is shown in chapter 4. In the beginning of the chapter the lab network setup, including its hardware and software components as well as the network layout, will be presented before in the next section the impact metrics used during the thesis as well as the measurement tools and their configurations, are given.

The next chapter 5 will then dive into the topic of (D)DoS attacks. The chapter involves the taxonomy of attacks and further the specific attacks that have been selected and analyzed during the thesis. Subsequently the implementation of the attacks with the used tools, their functionality and also their configurations will be explained. In the last section of the chapter the (D)DoS attack strength measurements that have been performed during the thesis to determine the required attack strength to cause a DoS for a legitimate client are presented.

Chapter 6 introduces the mathematical methods, before chapter 7 presents the modeling where the methods have been used. The chapter 7 is split into two parts, the first part deals with the metric selection progress and the second part with the model fitting.

The evaluation of the models is summarized in chapter 8, explaining the setup variations that have been studied and giving the corresponding results. The results obtained during the model prediction and the associated error calculations are also presented in this chapter.

The audit framework that has been developed during the thesis is presented in chapter 9. This chapter outlines the design and requirements of the framework and also explains how the framework can easily be extended with additional (D)DoS attacks and further options.

Finally the last chapter 10 summarizes and concludes the thesis and takes a look at future work that the author deems to be suitable to further improve the presented analysis method.

Lastly the appendix will then include additional explanations of used tools and configuration settings that did not fit in the previous context and also list the audit framework and MATLAB code developed during the thesis.

Chapter 2

Problem Statement

(D)DoS attacks still represent a major threat in the Internet, especially since the potential damage on critical infrastructure caused by the attacks is usually not a priori known. As such the measurement-based auditing method that is developed in this master thesis should asses the impact of (D)DoS attacks on Internet web servers.

For the auditing method the following systems are assumed to exist:

- s the Internet web server under consideration
- c the client system assessing the attack impact using external metrics
- a_i the hosts used to generate the attack (probe) traffic, where $i = 1, \dots, N$ with N being the number of hosts and $N > 1$ in the case of a DDoS attack.

Since the view of an outsider is taken during the analysis, the auditing method requires no privileged administrator access to the Internet web server under consideration or to the involved networks.

The auditing process should be such that the client system c continuously monitors the web server under consideration s and thus can abort the process when the server becomes overstressed. Ideally however, the server does not get overloaded during the measurements due to the trade off of the measurement load with the accuracy of the prediction.

During the auditing process, hosts a_i send attack traffic to the web server s starting with no traffic and stepwise increasing the strength. The (D)DoS attack strength required to cause a DoS for legitimate clients depends on the system under consideration and sample values for the lab system are given in a later stage of this report.

Combining the measurements performed by the client system with the predictive models of the (D)DoS attacks developed during the thesis, the impact of the (D)DoS attack can be estimated. To make the application of the auditing method suitable in an operational environment, the auditing method will trade off accuracy for probe traffic.

The goal of the presented auditing method is to achieve an average error rate between the theoretically calculated attack impact and the actually measured impact, which is lower than

10%. The required attack strength should not exceed 25-50% of the attack strength causing a DoS condition. This seems to be a reasonable tolerance relating to the current usage of over-provisioning the bandwidth by 55 to 75% over the traffic peak [27].

Chapter 3

Related Work

In this chapter the related work that has been studied in the course of this master thesis is presented and summarized.

3.1 Impact Metrics

To be able to examine the impacts of DoS attacks one requires accurate metrics. In an ideal world one would have access to a holistic network-wide view that spans across all involved networks and allows to fully assess the impact of an attack. In addition any collateral damage to the transporting networks and users that is caused by the DoS attack or the applied defense strategy could be captured.

In a real world scenario it is however very unlikely to possess such a view and we are restricted to monitor a couple of parameters in a few parts of the impacted networks. Due to the broad range of current DoS and DDoS attacks [2] there exist a variety of impact metrics often targeted to a specific group of attacks.

In this section an overview over various metrics used in previous work is given. The presented metrics range from purely economic and theoretical to measurable and application specific ones. Since the goal of this thesis will be to predict the impact without complete access to a network, further attention has been paid to the possibility of measuring the attack impact without any sophisticated knowledge about the attacked network or end-host system. The results of the literature research are also summarized at the end of the chapter in table 3.1.

In [25] a purely economic damage model for large-scale Internet attacks was presented. Various types of financial damage, divided into the four categories «**Downtime Loss**», «**Disaster Recovery**», «**Liability**» and «**Customer Loss**», which are expected after any significant degradation of Internet performance have been included in the model. Since there is a vast numbers of factors involved that not only depend on the target and its business area but also on the actual DoS attack, the duration and possible countermeasures, the presented financial loss calculations can only serve as a rough approximation or worst case scenario. Due to the required financial insight and situation of risk, the presented model is mostly suited as a basis to estimate how much investment into improved infrastructure robustness and faster disaster recovery is justified.

A further DDoS impact scale that takes the economic impact of an attack into account was presented in [74]. The metric has been designed from the perspective of network operators where particular interest is paid to «cost of SLA violations» and «the cost of losing customers». In a further step the transition from the proposed impact scale which requires sophisticated data and knowledge to calculate is made to one that can be easily estimated in practice. Using only measurable or readily available network data the authors approximate their *Measure of Impact of DDoS AttackS* (MIDAS) to ease the computation.

The **MIDAS2007 scale factor (SF)** [74] which is constructed to be globally applicable is the approximated cost of a DDoS attack normalized using the network operator's revenue of the last year.

$$MIDAS2007\_SF = \frac{C2007_{DDoS}}{NetworkTotalRevenue(12months)} \tag{3.1}$$

The approximated DDoS cost consists of the SLA violations and the possible revenue loss due to the risk of leaving customers. It is defined as $C2007_{DDoS} = C2007_{SLA} + \sum_c [Risk2007(c) * Rev2007_{future}(c)]$.

The **MIDAS2007NET factor** [74] is the previously mentioned variant that can be computed based on collected network data. It has the same properties as the original factor however they are not directly comparable. The idea behind the approximation is that the provisioned bandwidth is estimated to be proportional to the actual traffic volumes on the network and thus proportional to the revenues.

$$MIDAS2007NET = \frac{C2007\_net_{DDoS}}{totalcapacity} \tag{3.2}$$

As such the costs of a DDoS attack are defined as $C2007\_net_{DDoS} = totalcapacity + \sum_c [Risk2007(c_i) * customercapacity(c_i)]$ where the total revenues of the provider have been replaced by the sum of the link capacities at the perimeter of its network and the revenue from each customer has been replaced by the total link capacities to which the customer connects to. The model is suited for network operators as in most cases the required network data can be easily collected.

In the following, we are turning away from the mostly theoretical economic impact models to the measurement based ones.

The authors of the paper [28] took a look at the impact caused by a particular DDoS attack, called «Land Attack», to Windows XP, Windows Vista and Mac OS X 10.5. To measure the impact they monitored and compared the **memory and processor exhaustion** of the systems under attack. Since such measurements require full access to the targeted system they cannot be performed without the system operator's cooperation.

In [11] the differences between multiple emulation based testbeds versus simulation of TCP targeted denial-of-service attacks have been explored. For the experimental design the following metrics were computed.

- **Average goodput** in KBps, computed by dividing the transfer size by the transfer completion time
- **Average congestion window size** in packets, computed by dividing the weighted congestion window average by the average maximum segment size (MSS)
- **CPU** percentage utilization
- **Packets per second** received and sent on the test network interfaces

The conclusions drawn in the paper were that there exist key differences between emulation and simulation based experiments with seemingly identical configurations. Some of the differences were attributed to the assumptions taken by the simulator; as such the authors recommend the usage of testbed scenarios which are more accurate to real world scenarios than purely simulated experiments. The chosen metrics again require the cooperation of the network and system operator.

In another paper [39] a queue-based analysis approach has been taken to qualitatively and quantitatively analyze the impact on three affected system parameters. As an important fact the authors stress that the chosen parameters do not have to be observed for a long time to understand the system degradation that may happen under certain attacks. The analyzed parameters are the **average queue-growth rate** which is often used in congestion control [26], the **arrival rate** which can be used to detect flooding attacks [33] as well as the **response time** which may point to the presence of a DoS attack.

In the paper the authors formally analyzed the performance of the chosen metrics during flooding as well as complexity DoS attacks. A flooding attack was depicted to correspond to an attack that sends too many requests to a system resource, thus having a high arrival rate. A complexity attack was assumed to consist of lengthy requests to a resource without increasing the arrival rate significantly.

While the average queue-growth rate and the arrival rate are measured server side the response time can be measured on the client side.

The results of the formal analysis showed that during strong attacks the average queue-growth rate can detect flooding as well as complexity attacks. Under the assumption that weak attacks do not pose a significant threat the authors decided not to further comment on them. The analysis showed that either of both attacks can have more impact on size of the resource queue depending on the attack parameters. While the queue-growth rate increased linearly during a flooding attack with increased arrival rate it remained constant for complexity attacks after a certain level. Likewise the response time of the requests depended on the chosen attack parameters such that any of the attacks may have a greater impact.

The authors of the papers [47] and [48] decided to take a look at various existing metrics before presenting their own metrics. The following listing gives an overview over the various metrics, their definition and usage and also tries to point out in which cases they fail to provide a useful result.

- The **packet loss** is the number of bytes or packets that were lost due to the interaction of the attack traffic with the legitimate traffic or due to collateral damage caused by the defense strategy. Its use is primarily to measure the presence and extent of network congestion that is caused by flooding attacks. As such it cannot be used reliably against attacks without continuous network congestion as is the case with low rate, pulsing or attacks that target application vulnerabilities. Another weak point of this metric is the fact that in general it does not distinguish between the different types of lost packets although the type provides essential information since they have different impact (e.g. lost TCP SYN versus lost data packets).
- The **throughput** metric is defined as the number of transferred bytes per time unit. **Goodput** is similar however it does not count the retransmitted bytes. Indirectly this metric captures not only the network congestion but also the prolonged duration of legitimate transactions since both parameters are connected to TCP-based traffic which responds to network congestion by lowering its sending rate. However any applications that are sensitive to jitter or loss of specific (e.g. control) packets may still experience high throughput levels although the quality of service requirements expected by the user are not satisfied anymore. Also the metric cannot be applied effectively to traffic consisting of short connections with few packets since it already has a low throughput.
- Denial of service of an interactive application (e.g. telnet) can be measured using the **request** / **response delay**. It is defined as the interval between the issue of a request and the complete reception of its corresponding response. The metric is inapplicable to noninteractive applications which already have large thresholds for the delay. Likewise it is not applicable to one-way traffic where no response is expected.
- Looking at a complete set of messages exchanged between a source and its destination one can define the **transaction duration** which captures the time needed for such an exchange. The metric depends heavily on the exchanged data volume and the type of application (e.g. interactive or congestion sensitive). While accurately measuring the service denial for interactive applications (e.g. web browsing) it does not capture applications such as one-way traffic (e.g. media streaming). Also for many noninteractive applications extended transaction duration does not imply a denial of service and is accepted by humans.
- The **allocation of resources** is another often considered metric. It is the fraction of a critical shared resource allocated to legitimate traffic versus attack traffic. The shared resource may be bandwidth, buffer space or any other application specific resource. The metric fails to capture the user-perceived service quality and assumes that a lack of resource is the cause of service denial. Further it fails to capture the collateral damage since a defense strategy that drops 90% of legitimate traffic and 100% of attack traffic would appear to be perfect.

The authors of the paper conclude that all those metrics even when combined have the disadvantage of not being quantitative since they do not specify any appropriate ranges that correspond to service denial. In fact they bring up the fact that such values cannot be specified in general because they depend highly on the type of applications (e.g. loss of VoIP versus DNS traffic). Also the metrics have never been proven to correspond to human user's perception of denial of service. Instead the authors propose a new Dos impact metric where they paid attention to application specific QoS requirements. In addition to their research they extended their findings with those made by the 3GPP consortium in [49] as well as several findings from further contemporary QoS research [6], [4], [5], [61].

The list of applications can more or less be grouped into five categories each with specific QoS requirements that have to be fulfilled for a successful transaction. A transaction usually involves a single request- reply exchange between a client and a server or a very close sequence of multiple request reply exchanges.

The first category which are the ***interactive applications*** such as web, file transfer or e-mail (between user and server) always involve a human user that sends a request and waits for its response which has to be served within an user-acceptable delay. According to the paper a higher total delay is tolerated if the data arrives incrementally. As such the category has two delay constraints: *partial delay* between the receipt of any two packets and the *whole delay* measured from the end of the request until the entire response has arrived. Further applications may even have additional delay specific requirements (e.g. echo delay requirements for telnet < 250ms).

Media applications (e.g. audio or video streaming) have strict requirements for low loss, jitter and one-way delay. Since media applications usually use multiple channels (control and data) both of them are required to provide satisfactory results to the users. The control traffic is considered to be part of the interactive applications category.

Since ***online games*** have strict requirements as well for low one-way delay and loss, they are part of an individual group. The online games are divided into two subgroups of first person shooters and real-time strategy games as according to research they have different requirements.

Chat applications may have media channels where the requirements of the media category apply. Further a threshold value of 4 seconds is defined that applies to each acknowledge message sent by the server in reply to a user initiated message.

The last category contains any ***noninteractive services*** (e.g. e-mail transfer between servers) where users accept a longer delay as long as the transaction will be successful within a given interval.

The complete list of applications and their defined QoS requirements can be found in table 1 in [48].

According to those QoS requirements the authors came up with the following metrics which aggregate the transaction success and failure measures.

- The **percentage of failed transactions (pft)** is a metric that is collected per application type. It quantifies the QoS experienced by the users that captures the impact of a (D)DoS attack. Instead of directly calculating the percentage as the failed transactions divided by all transactions, which may give a biased result for certain applications, the authors chose to calculate the *pft* as the difference between 1 and the ratio of successful transactions divided by *all transactions that would have been initialized if the attack were not present*.

- The **DoS-hist** metric is used to understand the application's resilience to an attack by showing the histogram of *pft* measures across applications.
- **DoS-level** is the weighted average of *pft* measures. Since it depends highly on the chosen weight the DoS-level can be biased but is still interesting when a single number is required to describe the DoS impact.
- Finally to quantify the severity of service denial the **QoS-degrade** metric has been introduced which is computed for failed transactions. It is the ratio by which the applications transaction's measurements exceed its QoS threshold. e.g. A *QoS-degrade* value of N means that the service of failed transactions was N times worse than a user could tolerate.

The collection of the required data to calculate the metrics in a testbed or real-world scenario is complex. Either the client applications have to be modified to compute the required measurements directly or a trace based approach can be used where the transactions and request / responses have to be identified later on during the trace analysis which may be challenging [46].

In [59] the authors employ multiple metrics to evaluate the impact of DDoS attacks on web services during flash crowd events (legitimate web traffic at high rate). The analysis uses the following metrics which also according to [46] properly signal denial of service for HTTP, FTP and DNS traffic.

- The **throughput** which can be divided into goodput and badput, where the **goodput** is defined as the number of bits per second of legitimate traffic over the bottleneck link whereas the **badput** is the attack traffic per second.
- The elapsed time between the end of a request and the beginning of a response is defined as **response time**.
- The percentage of legitimate request that are being dropped due to congestion on the bottleneck bandwidth is the **percentage of request packets lost**.
- The **legitimate packet survival ratio** measures the delivered legitimate packets during an attack.
- Transactions that fail with a RST packet being sent by the targeted server are accounted under the **percentage of failed transactions** metric.
- Finally the **bottleneck bandwidth utilization** is defined as the percentage of bandwidth that is used for goodput.

During a similar simulation based approach to evaluate the performance of web services under a DDoS UDP flooding attack the same authors [58] additionally used the **average serve** / **request rate**, which is defined as the number of receives generated by the server to the number of requests generated by the clients.

Flooding attacks with UDP, TCP and ICMP bandwidth flood with FLAT, PULSE and also RAMP distributions have been as DDoS attack scenarios. The results showed that various attacks had different impact on the measured goodput, the average response time increased almost 3 to 5 times during the attacks and the average percentage of failed transactions increased 5 to 10 times. As expected the legitimate packet survival ratio started to decrease with increased rates of attack traffic. Finally the average bottleneck bandwidth utilization showed that during an attack the value drops more than 50%.

While the average serve / request rate is close to 1 without attack it decreases as the strength of the attack increases.

Table 3.1: DoS impact metrics

Metric	internal / external	data	OSI layer
financial damage model	internal	expected financial damages	*none*
MIDAS2007 scale factor	internal	risk, SLA costs & (expected) revenue	*none*
MIDAS2007NET factor	internal	risk & provisioned bandwidth	2
memory exhaustion	internal	local memory usage	*none*
processor exhaustion	internal	local system cpu usage	*none*
average goodput	mostly internal	transfer size & transfer duration	7
average congestion window	depends	congestion window size & MSS	4
packets per second received and sent	internal	ethernet frames / ip packets per second	2,3
average queue-growth rate	internal	arrival rate, queue capacity, service time	2
arrival rate	internal	IP packet arrival rate	3
response time	internal	full reception of request & start of reply	7
packet loss	depends	lost packets or bytes (due attack)	3,4
throughput (goodput / badput)	mostly internal	transfered bytes / time unit	2,3 (good / bad 7)
request / response delay	external	duration between request and complete reception	7
transaction duration	external	duration between a complete transaction	7
allocation of resources	internal	measurement or resource usage	*various*
percentage of failed transactions *(pft)*	external	failed transactions per application types	7
DoS-hist	external	histogram of *pft*	7
DoS-level	external	weighted average of *pft*	7
QoS-degrade	external	required QoS values and measured values	7
percentage of request packets lost	internal	sent request packets and dropped ones	7
legitimate packet survival ratio	internal	legitimate delivered and sent packets	7
percentage of failed transaction	external	failed transaction with RST sent	4
Bottleneck bandwidth utilization	internal	measured bandwidth and actual bandwidth	2
average serve / request rate	internal	number of server receives & client requests	7

Table 3.1: DoS impact metrics (continued)

Metric	financial	evaluation & tools	references
financial damage model	✓	*theoretical calculation*	[25]
MIDAS2007 scale factor	✓	*theoretical calculation*	[74]
MIDAS2007NET factor	✗	*calculation*	[74]
memory exhaustion	✗	procfs, free	[28]
processor exhaustion	✗	procfs, mpstat	[28]
average goodput	✗	*trace analysis*	[11]
average congestion window	✗	[c], [e]	[11]
packets per second received and sent	✗	procfs,[b] iptables (counter)	[11]
average queue-growth rate	✗	*theoretical calculation*	[39], [26]
arrival rate	✗	[b] iptables (counter)	[39], [33]
response time	✗	[a] (internal)	[39]
packet loss	✗	ping (ICMP), [g] Iperf (UDP), [f] sting (TCP)	[48]
throughput (goodput / badput)	✗	Iperf (TCP)	[48]
request / response delay	✗	[d] (external)	[48]
transaction duration	✗	*trace analysis*	[48]
allocation of resources	✗	*various*	[48]
percentage of failed transactions *(pft)*	✗	*modified clients / trace analysis*	[48]
DoS-hist	✗	*modified clients / trace analysis*	[48]
DoS-level	✗	*modified clients / trace analysis*	[48]
QoS-degrade	✗	*modified clients / trace analysis*	[48]
percentage of request packets lost	✗	*trace analysis*	[59]
legitimate packet survival ratio	✗	*trace analysis*	[59]
percentage of failed transaction	✗	*trace analysis*	[59]
Bottleneck bandwidth utilization	✗		[59]
average serve / request rate	✗	*trace analysis*	[58]

[a] http://silentorbit.com/notes/2010/01/measuring-delay-in-webserver-response/
[b] http://stackoverflow.com/questions/349576/linux-retrieve-per-interface-sent-received-packet-counters-ethernet-ipv4-ipv
[c] http://linuxgazette.net/136/pfeiffer.html
[d] http://www.vanheusden.com/httping/
[e] http://linuxgazette.net/137/pfeiffer.html
[f] http://cseweb.ucsd.edu/ savage/papers/Usits99.pdf
[g] http://sourceforge.net/projects/iperf/
[h] http://tstat.tlc.polito.it/index.shtml
[i] http://www.hping.org/
[j] http://netsniff-ng.org/

Chapter 4

Methodology

This chapter presents the methodology applied in this master thesis. In a first part, the used lab network setup is outlined and in a subsequent section the analyzed internal and external metrics are presented including the tools used to measure them. The last part of the chapter describes the post-processing of the captured data.

4.1 Lab Network Setup

Most of the experiments during the thesis have been performed in a lab environment that had no access to the institute's productive network or the Internet to prevent accidental damage to the remaining network infrastructure and its systems. The set up lab network was chosen to have a dumbbell topology as shown in figure 4.1, such that multiple TCP sessions share a single bandwidth-limited bottleneck and thus have to compete for its bandwidth. The dumbbell topology is often used when studying network traffic properties, as all intermediate links except for the bottleneck link can usually be ignored [31, 10].

For our experiments the bottleneck link was chosen to represent the access link to the server infrastructure. On the server side the attacked web server as well as an additional system, which was used when simulating cross traffic that further utilized the bottleneck link, were stationed. The other side of the bottleneck link represented the various Internet hosts, that included multiple attacking hosts and clients. On one of the clients various measurement parameters, that will be explained in section 4.2 in detail, have been collected while additional hosts have been used to generate simulated TCP traffic.

To reduce the effects that additional bottleneck links would cause, all networking devices and cables were gigabit capable, except for certain measurements where a few of the cables have been replaced with different ones that had their wire pairs 1 and 4 disconnected such that they would only allow a speed of 100 Mbit/s. To connect the network links, two unmanaged desktop gigabit Ethernet switches have been used [54]. The use of a modified network cable was necessary between the switches when requiring 100 Mbit/s links, whereas on the end systems the network device configuration could also be changed on the software level e.g. using the «ethtool» utility, as is shown in section 4.3.

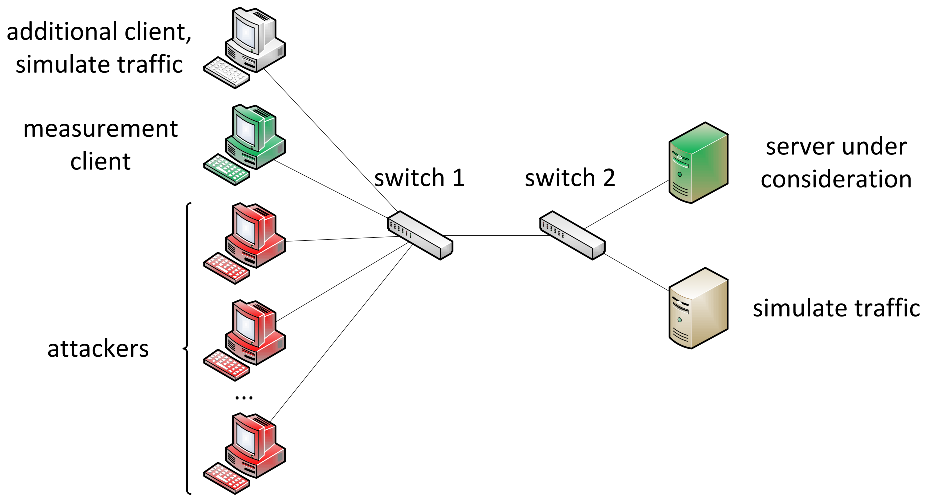


Figure 4.1: Lab network setup

The client and attacker hardware all consisted of similar desktop computer systems that included an Intel Core i3-540 processor [35] and 4 GB of RAM. The used operating system was the current LTS (Long Term Support) version of Ubuntu (an operating system based on the Debian Linux distribution), which at the time of the thesis beginning was still version 10.04.4. The used Kernel version was still at 2.6 with the exact version being 2.6.32-24.

A first web server consisted of the same hardware while the second server had a more recent Intel Core i7-2600K processor [36], 16 GB of RAM and a solid-state drive (SSD) installed. Both server systems also used the Ubuntu flavor in its LTS version as their operating systems but in its 64-bit server edition with the Kernel version 2.6.32-41.

As web server software Apache 2 [65] as well as lighttpd [40] have been used together with PHP [69] and MySQL [51] as a database backend when dynamic content was served during the measurements.

During the last steps of the thesis a few measurements could be performed in an actual productive environment. The measurements employed the network infrastructure that was shared with multiple users and the attacks targeted three systems that were in productive use as well.

While one targeted system was running in a virtual machine, the remaining systems were two virtual machine hosts running VMware's ESXi virtualization hypervisor [75]. On the virtual machine webn application was located, running on a Jetty web server [66]. The ESXi servers, one running version 4 and the other version 5 of the virtualization product, both provided access to a limited web interface with simple product information and links to the product website. In the case of ESXi version 4, the web server also provided access to a downloadable management software.

The network layout was such that the attackers and the client were connected to a 100 Mbit/s switch that directed the network traffic through a firewall to a second switch where the targeted servers were connected.

4.2 Metrics

An initial task of the master thesis consisted of selecting suitable metrics that could be used to capture the current web server system state and thus also the impact caused by the (D)DoS attacks.

As explained in chapter 3, in an ideal world one would have access to a holistic network-wide view that allows to fully assess the impact of an attack. This is however not the case in a real world scenario, where the monitoring is restricted to a few points and parameters in the system. Also as a further restriction that was posed by the thesis' task description, was to take the view of an outsider without access or sophisticated knowledge about the attacked network or system. As such although multiple internal metrics were included in the measurements, especially during the first few ones where they have been used to determine the required attack strength to cause a denial-of-service, they were avoided in the modeling when possible.

The chosen metrics have been selected based on the literature presented in chapter 3, that unfortunately, mostly contains internal metrics, such that additional internal and external metrics have been included during the measurements.

To measure the defined metrics appropriate measurement tools and configurations have been chosen. The summary of the selected internal as well as external metrics, along with their definition and selected measurement tools, is given in tables 4.1 and 4.2. The measurement tools and their configurations are described in detail in section 4.2.1.

4.2.1 Measurement Tools

The following section presents the used metric measurement tools in detail, including the configuration options employed during the measurements.

mpstat [19, 30]

The mpstat utility, which is part of the «sysstat» package in Ubuntu, is used to report the activity of the available processors or processor cores. The first output of the utility was ignored, as has been recommended in [79], since the first report interval is defined since the system startup and only afterwards the reporting interval is measured between two subsequent outputs.

By default the tool reports the CPU utilization that is divided into several categories, including the time spent at the user level (applications), at the system level (kernel), waiting for I/O, servicing hardware or software interrupts or being idle.

Table 4.1: Internal metrics

metric	definition	measurement tool
CPU utilization	Reports the CPU utilization divided into multiple categories as explained in the tool description.	mpstat
memory usage	Measures the current memory usage either overall or divided into several categories.	free, /proc/meminfo
IO utilization	Reports the device utilization of the drives.	iostat
IO avg wait	Measures the average time for requests sent to the device including the time spent in queues and servicing.	iostat
received pkts / s	Captures the number of received packets per second by the network interface.	netstat
throughput	The number of bits per second of traffic sent over the bottleneck link.	netstat
goodput	Defined as the number of bits per second of legitimate traffic sent over the bottleneck, not counting the retransmitted bits.	tcpdump, curl
No. of TCP conn.	Counts the number of TCP connections that are currently in the ESTABLISHED state.	netstat
served req. / s	Assesses the number of requests per second that are served by the web server.	web server log

Table 4.2: External metrics

metric	definition	tools
ICMP RTT	Measures the round-trip time of packets using the ICMP echo request and reply packets .	ping
ICMP packet loss	The packet loss on a link is determined using the rate of lost ICMP echo reply or request packets.	ping
ACK RTT	Captured by the metric is the round-trip time of TCP acknowledgment packets.	tcpdump, tshark
retr. segments	The retransmitted TCP segments count assesses the number of retransmissions that occur, incl. packet loss or corrupt packets.	tcpdump, tshark
lost segments	Quantified by the lost TCP segments metric is the number of times the previously expected TCP packet was not seen.	tcpdump, tshark
avg. TCP window	Shows the calculated average TCP window size.	tcpdump, tshark
throughput	Measures the average rate of successful packet deliveries between server and client.	curl

The command is executed as `mpstat -P ALL 5 24`, where the `-P ALL` options reports the stats for all processors individually and the arguments `5` and `24` define the interval in seconds between two reports and the number of outputs before exiting.

free [17]

To determine the current memory consumption two different methods have been used. The first method used the `free` utility which displays the total amount of free as well as used memory in the system and it is also capable of showing the buffers in the kernel.

The command is executed using `free -m -s 5`, activating the output in megabytes using the `-m` flag and continuously refreshing the output statistics every 5 seconds using the `-s` switch.

/proc/meminfo [22]

As the second method, which can collect memory information in more detail the proc file system was used. The proc file system is an interface to the kernel data structures that can be accessed over a pseudo-file system that is commonly mounted at /proc.

The `/proc/meminfo` file reports the statistic about the current memory usage on the system. The reported usage is divided into more than 30 different categories, including the amount of free memory, cache memory or memory used by the Kernel but also the amount of memory that would be necessary to never obtain an out of memory error with 99.99% probability.

The content of the pseudo file was copied every second during the measurements.

iostat [18, 30]

The iostat tool is similar to the mpstat command. While the mpstat is restricted to the CPU statistics the iostat tool can also report system input and output device load. Using the «-x» flag, extended statistics will be output that include average request and queue sizes, number of blocks/s or MB/s written but also the device utilization as a percentage.

During the measurements the tool is executed using the command `iostat -d -x -t 5 24` that displays extended statistics of the device utilization report that is enabled with the `-d` flag. The `-t` switch enables the printing of the current timestamp, while the arguments `5` and `24` define the interval in seconds between two reports and the number of outputs before exiting similar to the mpstat utility.

netstat [20]

The netstat utility provides access to various networking information. The information ranges from networking connections and routing tables to interface statistics.

The netstat utility was used for two different purposes during the thesis. Firstly the tool was used to determine the number of current connections and their current states (e.g. ESTABLISHED or LAST_ACK). This information was only collected on the server and was mainly used to observe how many parallel connections the server could handle. The required netstat command is `netstat -W -n -e -e -t -u -w` where the `-W` flag activates the wide output, the `-e` flags active extended information and the `-t` , `-u` and `-w` flags enable the output of TCP, UDP and raw socket information.

Secondly the statistics interface of the netstat tool provides multiple metrics that relate to TCP connections and their performance. To enable the reporting of the statistics the `netstat -s` command is used.

Similar to the /proc/meminfo interface, the output was collected once per second.

tcpdump [70]

tcpdump is a powerful command-line tool that captures network packets using the libpcap library and also implements a packet analyzer that is commonly used to analyze network behavior and performance as well as any application that generates or receives network traffic. As tcpdump has been around since 1987 it has been ported to nearly all Unix like operating systems and has become the most commonly used packet analyzer. A description of the received and sent packet contents can be printed out directly to the console or the partial / whole packets can be stored for later analysis e.g. using the tools that are explained below.

To be able to handle the captured data, the data was recorded into multiple files using the `-G 60` option that rotates the file every 60 seconds. The filename is defined by the `-w %s.pcap` option and contains the current timestamp. To further reduce the captured data a filter option was specified as `host 192.168.74.16` such that only traffic destined to the host was included, also the `-s 96` option shortened the recorded packet data such that only the headers of the packets remained.

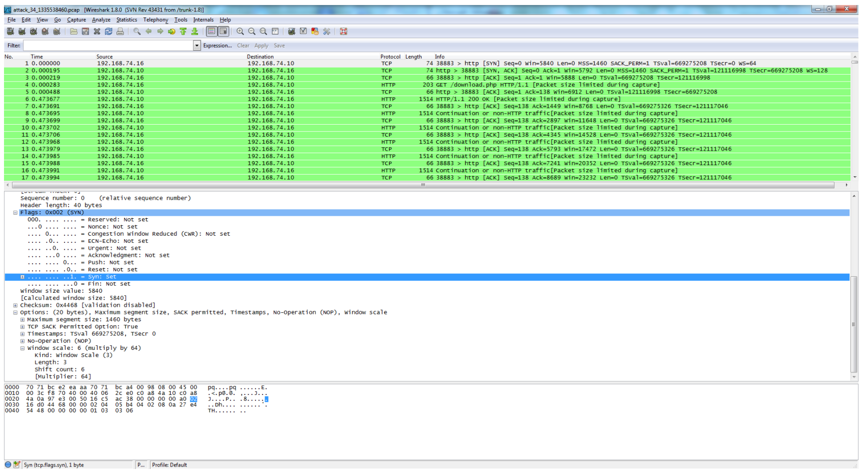
The complete command used is `tcpdump -i eth0 -n -w %s.pcap -G 60 -W 2 -s 96 host 192.168.74.16` where the only argument that has not yet been explained is the `-i eth0` option, that defines the network interface to listen on.

Wireshark [84]

Wireshark is a packet analyzer similar to tcpdump but has a graphical front-end as well as various post-processing, sorting and filtering options that ease the analysis of the captured packets. The post-processing features allow to take apart the captured data and decode the various packets and its layers, thus the data can be scrutinized almost arbitrarily.

tshark [83]

tshark is a command line network protocol / packet analyzer and works similar to tcpdump. It is part of the Wireshark tool suite and has the same advanced options as the graphical


attack_34_1335538460.pcap [Wireshark 1.8.0 (SVN Rev 43431 from /trunk-1.8)]
File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help
Filter:
Expression... Clear Apply Save

```
No.  Time      Source          Destination     Protocol Length Info
   1 0.000000  192.168.74.16   192.168.74.10   TCP        74 38883 > http [SYN] Seq=0 Win=5840 Len=0 MSS=1460 SACK_PERM=1 TSval=669275208 TSecr=0 WS=64
   2 0.000195  192.168.74.10   192.168.74.16   TCP        74 http > 38883 [SYN, ACK] Seq=0 Ack=1 Win=5792 Len=0 MSS=1460 SACK_PERM=1 TSval=121116998 TSecr=669275208 WS=128
   3 0.000219  192.168.74.16   192.168.74.10   TCP        66 38883 > http [ACK] Seq=1 Ack=1 Win=5888 Len=0 TSval=669275208 TSecr=121116998
   4 0.000283  192.168.74.16   192.168.74.10   HTTP      203 GET /download.php HTTP/1.1 [Packet size limited during capture]
   5 0.000488  192.168.74.10   192.168.74.16   TCP        66 http > 38883 [ACK] Seq=1 Ack=138 Win=6912 Len=0 TSval=121116998 TSecr=669275208
   6 0.473677  192.168.74.10   192.168.74.16   HTTP     1514 HTTP/1.1 200 OK [Packet size limited during capture]
   7 0.473691  192.168.74.16   192.168.74.10   TCP        66 38883 > http [ACK] Seq=138 Ack=1449 Win=8768 Len=0 TSval=669275326 TSecr=121117046
   8 0.473695  192.168.74.10   192.168.74.16   HTTP     1514 Continuation or non-HTTP traffic[Packet size limited during capture]
   9 0.473699  192.168.74.16   192.168.74.10   TCP        66 38883 > http [ACK] Seq=138 Ack=2897 Win=11648 Len=0 TSval=669275326 TSecr=121117046
  10 0.473702  192.168.74.10   192.168.74.16   HTTP     1514 Continuation or non-HTTP traffic[Packet size limited during capture]
  11 0.473706  192.168.74.16   192.168.74.10   TCP        66 38883 > http [ACK] Seq=138 Ack=4345 Win=14528 Len=0 TSval=669275326 TSecr=121117046
  12 0.473968  192.168.74.10   192.168.74.16   HTTP     1514 Continuation or non-HTTP traffic[Packet size limited during capture]
  13 0.473979  192.168.74.16   192.168.74.10   TCP        66 38883 > http [ACK] Seq=138 Ack=5793 Win=17472 Len=0 TSval=669275326 TSecr=121117046
  14 0.473985  192.168.74.10   192.168.74.16   HTTP     1514 Continuation or non-HTTP traffic[Packet size limited during capture]
  15 0.473988  192.168.74.16   192.168.74.10   TCP        66 38883 > http [ACK] Seq=138 Ack=7241 Win=20352 Len=0 TSval=669275326 TSecr=121117046
  16 0.473991  192.168.74.10   192.168.74.16   HTTP     1514 Continuation or non-HTTP traffic[Packet size limited during capture]
  17 0.473994  192.168.74.16   192.168.74.10   TCP        66 38883 > http [ACK] Seq=138 Ack=8689 Win=23232 Len=0 TSval=669275326 TSecr=121117046
```

```
    Sequence number: 0    (relative sequence number)
    Header length: 40 bytes
  ⊟ Flags: 0x002 (SYN)
      000. .... .... = Reserved: Not set
      ...0 .... .... = Nonce: Not set
      .... 0... .... = Congestion Window Reduced (CWR): Not set
      .... .0.. .... = ECN-Echo: Not set
      .... ..0. .... = Urgent: Not set
      .... ...0 .... = Acknowledgment: Not set
      .... .... 0... = Push: Not set
      .... .... .0.. = Reset: Not set
    ⊞ .... .... ..1. = Syn: Set
      .... .... ...0 = Fin: Not set
    Window size value: 5840
    [Calculated window size: 5840]
  ⊞ Checksum: 0x4468 [validation disabled]
  ⊟ Options: (20 bytes), Maximum segment size, SACK permitted, Timestamps, No-Operation (NOP), Window scale
    ⊞ Maximum segment size: 1460 bytes
    ⊞ TCP SACK Permitted Option: True
    ⊞ Timestamps: TSval 669275208, TSecr 0
    ⊞ No-Operation (NOP)
    ⊟ Window scale: 6 (multiply by 64)
        Kind: Window Scale (3)
        Length: 3
        Shift count: 6
        [Multiplier: 64]
```

```
0000  70 71 bc e2 ea aa 70 71  bc a4 00 98 08 00 45 00   pq....pq ......E.
0010  00 3c f8 70 40 00 40 06  2c e0 c0 a8 4a 10 c0 a8   .<.p@.@. ,...J...
0020  4a 0a 97 e3 00 50 16 c5  ac 38 00 00 00 00 a0 02   J....P.. .8......
0030  16 d0 44 68 00 00 02 04  05 b4 04 02 08 0a 27 e4   ..Dh.... ......'.
0040  54 48 00 00 00 00 01 03  03 06                     TH...... ..
```

Figure 4.2: Wireshark

interface provides. While tshark is not as common as tcpdump it is at least as powerful and as such allows statistical analysis of the captured data.

curl [63]

During the measurements when the throughput between web server and client was of interest, the cURL program was used to transfer a file with a large file size between the two hosts since the usually used iperf [68] throughput measurement tool cannot be used without privileged access to the web server.

As the output of the cURL program can be formated arbitrarily using the `--write-out` option, it was arranged such that it could be parsed easily. To prevent any discrepancies caused by the receiving host and its hard-drive, no data was stored on the end host. The cURL measurements have been run once per minute during the measurements using the following command:

```
curl --output /dev/null --silent --show-error --write-out
  "total=%{time_total} connect=%{time_connect}
  pre=%{time_pretransfer} start=%{time_starttransfer}
  size=%{size_download}  speed=%{speed_download}
  con=%{num_connects}\n" --max-time 59 192.168.74.10
```

ping

The ping utility sends ICMP echo request datagrams that elicit an echo response from the host they are sent to. The packet stores a timestamp that enables the stateless calculation of the packet round-trip time.

The used command was `ping -n -c 5 -l 5 -W 2` that includes the option `-n` to show numeric output only, thus never resolve any addresses, the sending of 5 requests without waiting between them using options `-c 5` and `-l 5` and lastly the inclusion of a timeout of 2 seconds with the `-W 2` option. To end more than 3 concurrent requests super user privileges are required.

4.3 Additional Tools

ethtool [16]

ethtool is a utility to display or change the settings of an Ethernet device. As previously noted in section 4.1 some measurements have been performed using 100 Mbit/s links, as an alternative to the modified network cables the ethtool also allows to change the speed mode of a device. Using the ethtool the following command was used `ethtool -s eth0 speed 100 duplex full autoneg off`.

The used switches were also capable of sending PAUSE frames to reduce the sending rate of the attached hosts, when a link was congested, to make sure the attackers would not respect this information and reduce their transmission speed their Ethernet device has been configured to ignore the received PAUSE frames. The required command is `ethtool --pause eth2 autoneg off rx off`.

ntpdate [21]

When a coordination of the measurements was required a common NTP server was setup and used by all systems. Using the ntpdate utility the date and time was then periodically updated on each host (`ntpdate 192.168.74.1`).

4.4 Post-processing

After collecting the various output information that was produced by the different measurement tools, the required metric values had to be extracted and further processed. Since the post-processing also involved aggregation, computation of statistics as well as plotting of the results, the processing was performed using MATLAB [67]. The MATLAB code developed during the thesis can be found on the accompanied CD and an overview over the involved MATLAB code with a short description of each file is also attached in appendix A.3.1.

The different steps of post-processing in MATLAB involved:

- Extraction of the metrics from the collected output.
- Aggregation of each individual metric and the calculation of its average for each measurement interval.
- Plotting of the metrics including their 95% confidence intervals.

Chapter 5

(D)DoS Attacks

All (D)DoS attacks have the same purpose, to deny service to legitimate clients [9]. However there is a huge range of attack types, each targeting different properties of applications, systems or the network infrastructure. As broad as the range of attacks, so are also the classifications of (D)DoS attacks [44]. A simple but common classification is to distinguish between bandwidth depletion in the intermediate network links and resource depletion on the network components or attacked end hosts.

5.1 Attack Taxonomy

The range of attacks is nearly infinite and figure 5.1, that is loosely based on [57], will not be able to distinguish all existing attacks. However the figure should give an overview over the selected attacks and allow the reader to put the attacks into context. The figure shows the most common DoS attack taxonomy where one divides between bandwidth and resource depletion attacks.

In a further subdivision of the bandwidth depletion attacks are the flooding attacks, where the attacker or systems controlled by the attacker (e.g. clients in a botnet) send a large amount of packets directly to the targeted system. Using spoofed source IP addresses in the sent packets further allow the attacker to hide its network location. As an alternative to directly attacking a system an amplification attack uses additional systems to increase the generated traffic and also allow the attackers to stay in second row where they remain hidden. In a DNS reflection attack, short queries with the target IP address as spoofed sender addresses are sent to DNS servers that will respond with larger reply packets and thus amplify the attack traffic.

Attacks that have become less prominent include the Smurf or Fraggle attack that use spoofed source IP addresses and send ICMP ping requests respectively UDP traffic to broadcast addresses to amplify the response packet count. Due to a change in the standard how broadcast packets are handled over network segments, the attacks have become less efficient.

On the resource depletion side of the diagram, the subdivision includes protocol exploit attacks that abuse certain properties of Internet protocols. The most common attack there being the TCP SYN attack, that abuses the threeway handshake in the TCP protocol to keep the connection in its waiting state and thus prevent further connections.

Application attacks target specific applications e.g. the used web server or database software. The targeted resources differ for the different attacks and include the number of used connections or memory and CPU consumption.

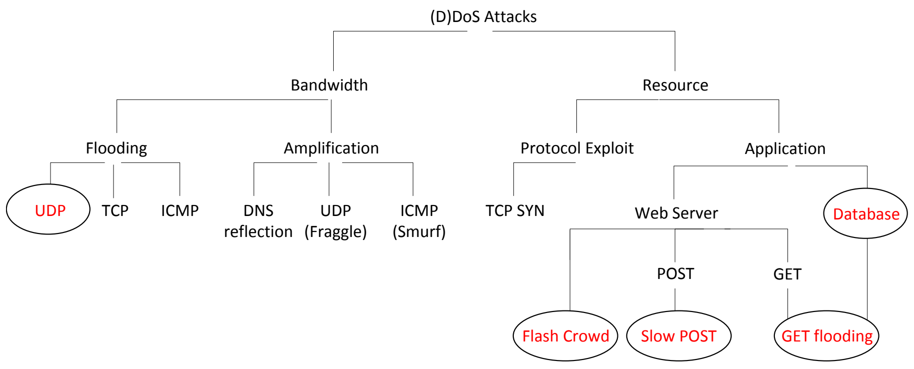


Figure 5.1: (D)DoS attack taxonomy based on [57]

5.2 Selected (D)DoS Attacks

Since different attacks target a wide range of parameters related to either end-host or networks, the (D)DoS attacks have been limited to a few specific cases that were selected in accordance with the current situation of observed attacks and their supposed impact taken from current reports and literature [2]. Where readily available attacks tools existed, some of them have been evaluated, while in other cases tools were further prepared or modified for the measurements, as will be shown later in this chapter.

5.2.1 UDP flooding

The typically observed UDP flooding attack is designed to consume large amounts of the available bandwidth of the intermediate network and its bottleneck links, resulting in high packet delays and packet loss rates on the impacted links. Additionally the attacked host will respond with ICMP «Port Unreachable» packets. To perform the attack, the attacking hosts send a constant stream of large UDP packets (e.g. 1500 bytes) where the source IP and ports are usually spoofed to harden the detection of the attacking hosts.

If the purpose of the attack is not bandwidth depletion but rather CPU depletion on the attacked system, a UDP flooding attack can also be used, however a very high rate of small packets would be the preferred and more effective configuration in that case [45].

The attack strength of UDP attacks has been observed to be steadily increasing in the Internet. Attacks of multiple gigabit per second have become common and in recent years traffic of up to 100 gigabit per second has been observed during large attacks [2].

To generate such high packet rates an attacker usually requires very large bot-nets of multiple thousands of hosts as most of those hosts will only have a very limited upload rate. A modern host is however easily capable to generate traffic of multiple gigabits per second. A software that allows the generation of such large amounts of traffic is the Linux packet generator pktgen [50] that allows to generate the packets directly in the kernel mode instead of having to use the user mode as most other UDP attacking tools do.

5.2.2 Slow POST

The Slow POST attack targets the end host system or more specifically the web server running on it. The attacking hosts initiate multiple HTTP POST connections to the web server and keep the connections alive by sending arbitrary POST data to the server over and over at a very low rate. The web server spawns child processes to handle the simultaneous requests until it reaches the maximum allowed number of child processes such that additional requests will be placed into a queue and eventually time-out. Initial measurements using the Apache 2 web server showed that a single attacking host would be capable of establishing enough connections to deny further connections by other hosts however during the measurements multiple attacking hosts have been used to distribute the load. To perform the attack on a web server a readily available tool with the name «Tor's Hammer» exists [52].

A similar variant of the attack is used by the Slowloris tool which became popular during the 2009 Iranian presidential elections [81, 56]. The Slowloris tool establishes TCP connections and keeps them open by creating HTTP requests that are kept incomplete by continuously sending additional header fields.

There exist a few methods to mitigate slow attacks, either by using a different web server software as not all are affected by the attack, by using additional software to limit the influence (e.g. Apaches «mod_security» or «mod_reqtimeout» modules) or also on the network level by using reverse proxies, load-balancers or by using firewalls that filter on the application layer.

5.2.3 Flash-crowd attack

A flash-crowd attack is similar to the happening of a flash mob where a group of unfamiliar people suddenly assemble in a place, often performing an unusual act, before they disperse again [78]. The term flash-crowd has been taken from the 1973 story «Flash Crowd» by Larry Niven where due to the invention of cheap teleportation an argument in a book shop, that is covered by a news team, quickly becomes a riot as more and more additional people join the scene.

A similar effect can be observed when a popular website publishes a link to a smaller website that subsequently cannot handle the thereby generated amount of requests and traffic. The effect is often called Slashdot-, which is a popular news site, or Twitter-, a microblogging service, effect.

The flash-crowd attack emulates the legitimate behavior of individual clients such that the malicious users cannot easily be distinguished from legitimate ones. The chosen attack scenario, where the attackers try to continuously download large files, can also be seen as an asymmetric resource utilization attack which results in a starvation for the legitimate clients. As most broadband clients have a much faster download than upload connection the attack will often require less hosts to generate a similar effect than are required with other flooding attacks.

The flash-crowd attack scenario is also interesting to look at from the point of information collection where a certain client tries to quickly collect a large amount of data from a web server without drastically impacting the performance of other users, that otherwise may result in undesired actions (e.g. blocking) being performed against that user.

5.2.4 GET flooding

GET flooding attacks target the web server by sending large amount of GET requests to the server which effectively will get overloaded by the requests. Due to the amount of requests and connections that the server has to handle it will not be able to further communicate successfully with any legitimate clients.

The effects of GET flooding attacks heavily depend on the additional software that is used on the server and is involved in the processing of the requests. As such the attack has been divided into two cases. Once the data accessed by the server system was readily stored on the hard-disk, while the second test case used an additional database back-end where the accessed data was stored. Furthermore on each request the server would store logging information containing the IP of the user and the current date in the database.

According to the World Wide Infrastructure Report, the HTTP GET flooding was the most common attack observed in the Internet during the 12 month survey period [2].

5.3 Attack Implementation

5.3.1 UDP flooding

To perform the UDP flooding attack the Linux packet generator «pktgen», that is provided as a Linux kernel module, has been used. The packet generator can be used to generate arbitrary sized UDP packets that during the attack were sent directly to the attacked web server. The attack strength ranged from 0 to 2000 Mbit/s in total, which was split between 5 hosts to distribute the load. The script that was used to configure the packet generator for the attacks can be found in appendix A.2.2.

```
./pktgen_udp.sh <delay between pkt [ns]> <target IP address>
  <next hop MAC> <pkt size [byte]>
```

Since the script takes the delay between two packets as its argument the conversion between the attack traffic to generate and the delay between two packets has to be approximated according to formula 5.1.

$$\text{delay [s]} = \frac{1 - \text{pkt\_rate [1/s]} * \text{pkt\_duration [s]}}{\text{pkt\_rate [1/s]}} \tag{5.1}$$

To compute the number of packets to be sent per second (pkt_rate [1/s]) and the duration to send a single packet over the link (pkt_duration [s]) the formulas in 5.2 are used.

$$\text{pkt\_rate [1/s]} = \frac{\text{attack\_traffic [bit/s]}}{\text{pkt\_size [bit]}}, \quad \text{pkt\_duration [s]} = \frac{\text{pkt\_size [bit]}}{\text{link\_speed [bit/s]}} \tag{5.2}$$

During the initial attack a packet size of 1500 bytes was used which was supposed to saturate the bottleneck link of the network with UDP packets. However the attack was only partially successful as the co-existing TCP traffic during an attack was merely reduced by a third of the usual throughput measured without an attack.

A second attack with a packet size of 60 bytes was more effective and created a DoS due to the huge amount of received packets at the web server system.

5.3.2 Slow POST

The slow POST attack has been performed using the Tor's Hammer [52] attack tool which is written in Python. The name Tor's Hammer is derived from the fact that the tool has an option to be used through the Tor anonymizing network [71]. For the measurements performed during this thesis the Tor option was however not used.

The strength of the attack is regulated by the number of parallel threads that the program uses and is specified using the `-r` option. The number of parallel connections were increased from 0 to 400 during the attacks. The audit framework presented in chapter 9 was responsible to evenly distribute the load between 5 attacking hosts.

```
./torshammer.py -t <target IP address> -r <parallel threads>
```

5.3.3 Flash-crowd attack

To simulate a flash-crowd attack with up to 150 concurrent connections, multiple parallel downloads have been executed using the curl tool, described in 4.2.1. Due to the limited number of attacking hosts, the parallel downloads were distributed equally between 5 attacking hosts. The various files that would be downloaded contained random data and had sizes between 256kB and 150MB.

After each download an attacker would wait a random time period between 1 and 5 seconds before starting the next download which is a behavior commonly found in web crawlers. In [12] a delay of 10 seconds is used while in [24] the time between two requests is only one second. To prevent slow downloads from stalling, a timeout depending on the size of the download was used, after which a non completed download would be aborted.

5.3.4 GET flooding

For the GET flooding attacks a set of different tools including netcat and ncat, that are described in A.1.3 and A.1.4, have been used to repeatedly transmit a previously recorded GET request. A script repeatedly called the program and piped the previously recorded request from a file as its input. Creating a new instance of the program made sure that a new connection was established for each GET request. Using 5 attacking hosts a maximum rate of up to 7000 GET requests in total, sent per second was achieved when no database access was involved and when a database back-end was used 2 attacking hosts with 40 GET requests per second were sufficient to deny further access due to the resulting memory exhaustion at the web server.

On the Linux systems used during the attacks the number of connections that could be established concurrently was initially limited to around 1000 due to the fact that a user is restricted to having 1024 open files at a time (in Linux a socket is managed through a file descriptor). To temporarily increase the soft and hard limit of open file descriptors to 65000, the `ulimit -SHn 65000` command has been used.

In addition the Twisted network engine as described in A.1.2 was used in a Python script when dynamically generating GET requests with further arguments instead of using the static but prerecorded GET request.

During the GET attack without MySQL back-end, the attackers and clients accessed a dynamic website that presented various PHP and server related information (collected using the `phpinfo()` command) to the user. The GET attack with database back-end always involved access to the MySQL database such that each request and the passed arguments would be logged into the database and the current number of requests with the same arguments have been presented to the user in addition to the information presented without database access.

5.4 (D)DoS Attack Strength

(D)DoS attacks and the used tools often provide multiple options that will influence the attack and thus also their strength and impact e.g. by choosing the maximum number of parallel connections that are established during the attack. However for most attacks and tools there exist no easy method to determine the ideal options that would cause a DoS condition on the web server or in the involved network. As such the options and parameter that had to be given to the attack tools first had to be determined by extensive measurements involving trial and error.

An attack strength measurement usually followed the following process:

1. Prepare and start several measurement tools on the web server and on a legitimate client host.

2. Start the attack tool with a new set of parameters and options.

3. Monitor the status on the the web server and the performance of the server measured by the legitimate client.
4. If no influence, caused by the attack, is determined directly on the web server or in terms of performance for the legitimate client, the current attack is stopped and the process is started again.

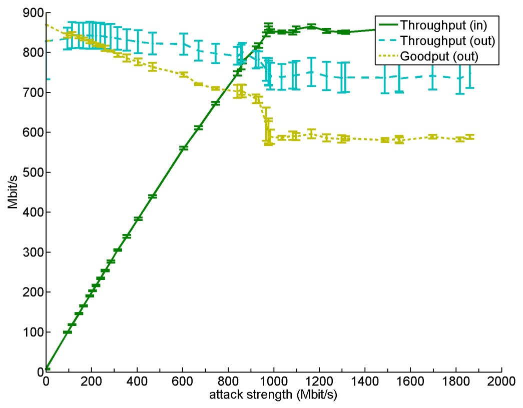
The whole process was repeated until a set of parameter was found that would cause a complete DoS condition for the measurement client or a limited DoS condition where the goodput dropped to 2% or less of the usual goodput without an attacker. If no DoS or similar condition could be obtained using a specific attack tool, the tool has been dropped or the attack was further modified to cause additional impact.

The figures in 5.2 present the goodput and throughput plots for the executed attacks. As has been shown in [59] and [46] the displayed metrics properly signal a DoS for Internet web servers.

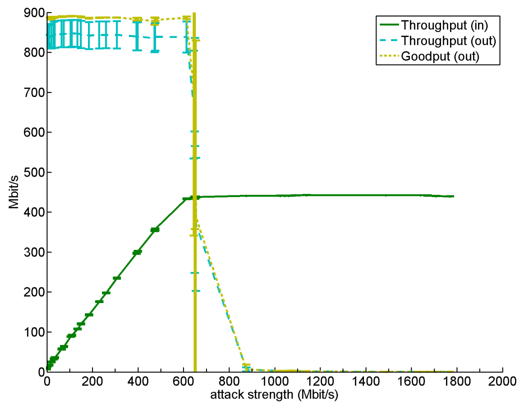
- The **throughput** metric is defined as the number of bits per second over the bottleneck link and was divided into the traffic being sent (out) and received (in) on the server side. Having both out and in metrics allows to see the attack traffic towards the targeted server and the resulting impact on the sent traffic as shown in figure 5.2a. The metrics were measured directly on the web server system since all traffic that leaves the network interface is directed over the bottleneck link. To measure the metric the `inoctet` and `outoctet` statistics, that are the number of received and sent bytes of the network interface, have been used.
- The **goodput** of the web server is similarly defined and captures the number of bits per second of legitimate traffic being sent over the bottleneck link. However in contrast to the throughput metric, the goodput does not count the retransmitted bytes. As previously described in chapter 3, the goodput metric indirectly captures not only the network congestion but also the prolonged duration of legitimate transactions since both parameters are connected to TCP-based traffic which responds to network congestion by lowering its sending rate. The metric is computed by dividing the transfer size by the transfer completion time.

The plots do not only show the average value of the metrics but also their 95% confidence intervals. Since the total measurement duration was 80 minutes with 2 minutes per attack strength, this resulted in 120 samples for the throughput metric that was collected once per second and 6 samples for the goodput metric collected 3 times per minute.

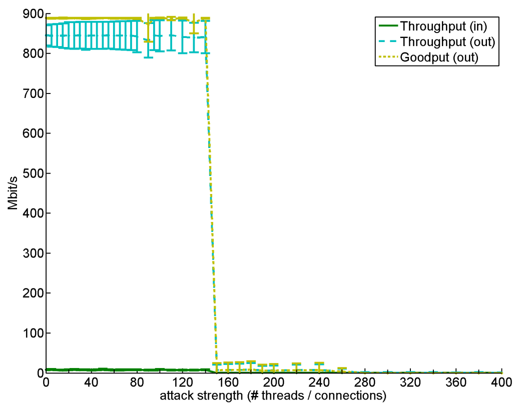
The UDP flooding attack with large packets was subject to saturate the bottleneck link of the involved network and thus prevent further access from legitimate clients, however the measurements showed that this was not the case. The web server was still reachable and the network congestion caused only minor issues. In figure 5.2a the attack strength of the UDP flooding attack (traffic sent to server) is displayed against the network performance (e.g. goodput). In the particular figure the difference between the measured goodput and the throughput being sent is due to retransmissions as well as the resulting ICMP errors (e.g. Port Unreachable) that have been sent by the web server as response to the attack.



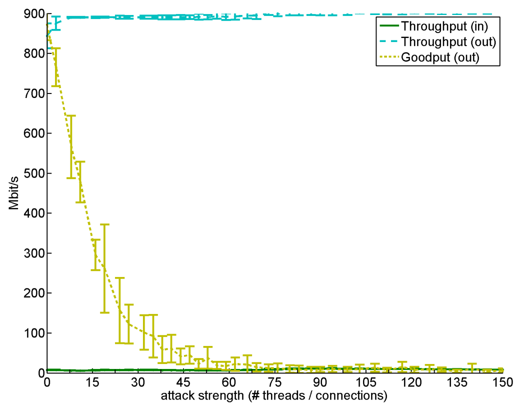
(a) UDP flooding (large packets)



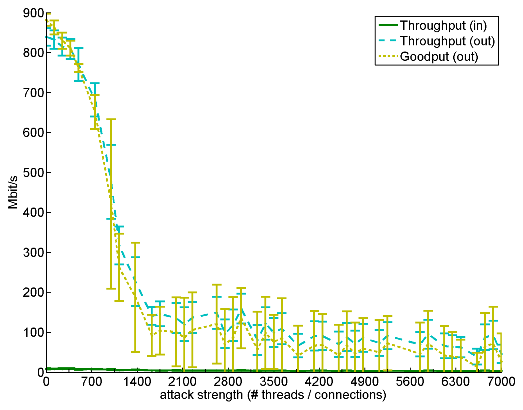
(b) UDP flooding (small packets)



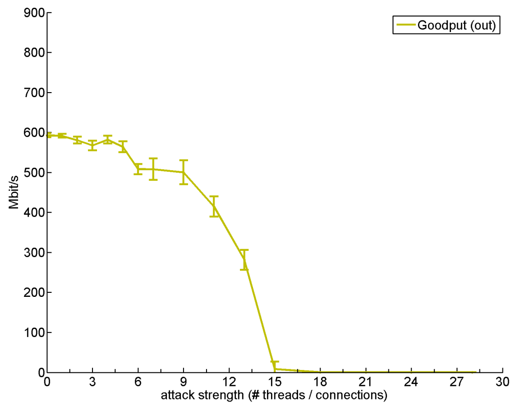
(c) slow POST



(d) Flash-crowd attack



(e) GET flooding (wo database)



(f) GET flooding (wi database)

Figure 5.2: Throughput plots for the performed attacks

In contrast the UDP attack which used small packets and that is shown in figure 5.2b was successful in creating a DoS condition as the server stopped the transmitting of further packets. The DoS condition was observed well before reaching the theoretically existing bottleneck link limit.

Figure 5.2c shows the slow POST attack, where the sharp bend is due to the web servers configuration, that limits the maximum number of parallel connections that can be established concurrently to 150.

In figure 5.2d, which shows the flash-crowd attack, the goodput decreases gradually with the attack strength while the total throughput remains nearly constant over the duration of the attack. The previously defined limit of 2% goodput, that is interpreted as a partial DoS condition, is reached after 69 concurrent connections.

Figure 5.2e, presents the throughput of a client, achieved during the GET flooding attack, where no database back-end was being used, shows that the attack had quite an influence at the beginning of the attack. However, the server remained reachable for the client with a throughput of around 10% of the maximum, such that the previously established definition of a DoS was not achieved.

In contrast, figure 5.2f shows a successful GET flooding attack where the database back-end was in use. Clearly visible only a few connections are necessary before the DoS condition occurs. Also captured in the figure is the fact, that the maximum goodput is well limited through the database back-end. The throughput metrics are not included in this plot as due to the enourmous CPU load (up to 100%) the metric was not reliably captured.

The final results, including all measurements, have been summarized in table 5.1 that shows the attacks, the used tools and if a DoS condition has been achieved during the measurement. The achievement of a DoS is defined as the access being denied for a legitimate client or alternatively the average goodput at the client dropping to less than 2% of the goodput without an attack being present. The condition column of the table shows the attack strength of the DoS attack that was required to successfully cause a DoS.

Table 5.1: DoS attack strength

Attack	Attack tool	DoS	condition[a]
UDP flooding (large packets)	pktgen	✗	-
UDP flooding (small packets)	pktgen	✓	875 Mbit/s
Slow POST	Tor's Hammer	✓	150 threads
Flash-crowd attack	curl	✓	69 connections
GET flooding (wo database)	ncat, nc, python-twisted	✗	-
GET flooding (wi database)	python-twisted	✓	15 connections

[a] required attack strength to cause DoS as defined

For the UDP flooding attack with large packets it is unlikely to achieve a DoS in the used environment since the network performance stabilizes after reaching the bottleneck limit as can be seen in figure 5.2a. It is assumed that the observed behavior is due to the used networking hardware.

In contrast the GET flooding attack in 5.2e could still result in a DoS as the performance is still decreasing. To achieve a DoS the attack strength should be increased further. As such additional attacking hosts may have to be utilized as the evaluated tools could not generate a stronger GET flooding attack.

During the measurements in the productive environment, which was introduced in chapter 4, a few additional observations were made that have not been experienced in the lab network and thus are worth mentioning.

Firstly the implemented slow POST attack was found to be limited to the HTTP protocol since the used attack tool does not support the HTTPS protocol. As such the web interface of the virtual machine hosts could not be tested as they are only accessible over HTTPS. The remaining Jetty [66] web server installation however proved to be vulnerable to the slow POST attack with 260 threads using all available connections such that no further client could connect.

A further observation was made during the GET flooding attack, where the virtual machine hosts would not be impacted severely with the available attack strength. However the virtual machine running the Jetty web server could be brought to a standstill and did not recover until the system was rebooted. For the attack to be successful a constant rate of 10 requests per second was sufficient.

The last attack that has been executed in the productive environment was the flash crowd attack where the previously mentioned management software was repeatedly downloaded from the ESXi version 4 virtual machine host. The attack showed to be very efficient such that 22 parallel downloads with an aggregated bandwidth of only 28 Mbit/s resulted in a complete DoS of the virtual machine host including the virtual machines running on the host. The server did not recover after the attack completed and thus finally had to be manually restarted. Unfortunately the attack could not be tested against the recent ESXi version 5 host, since its web interface does not provide a local download of the management software anymore.

Chapter 6

Mathematical Methods

In this chapter the mathematical methods used during the modeling process are elaborated. The chapter starts with Spearman's rank correlation that has been used to determine the correlation between the measured metrics and the DoS attack strength and is followed by the non-linear least squares method used during the model fitting.

6.1 Spearman's Rank Correlation

To measure the association between two variables, Spearman's rank correlation coefficient, which is also called Spearman's rho, is computed. If two variables are associated with each other, the knowledge of one value provides information about the likely value of the associated variable. Since the Spearman's rank correlation is a distribution-free measure it does not depend on the assumption of an underlying distribution such as the bivariate Normal distribution, as is the case with Pearson's correlation [43]. Also in contrast to Pearson's correlation is the fact that no linear dependency is required between the two variables but only a monotonic one.

The ranking algorithm used when calculating Spearman's rank correlation produces variables that fulfill the condition of correlation as their relationship becomes linear, such that Pearson's correlation can be computed between the ranked variables.

In practice to compute the ranks of a variable, declining positions are assigned to every value of the variable, using the average of the positions when ties exist (e.g. a value exists multiple times) [85]. To compute Spearman's rank correlation Pearson's correlation is then computed between the ranked variables as shown in equation 6.1.

$$\rho = \frac{\sum_i (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_i (x_i - \bar{x})^2 \sum_i (y_i - \bar{y})^2}} = \frac{Cov(x,y)}{s_x s_y},$$

where x_i, y_i are the ranks, $\bar{x}, \bar{y}$ the average thereof and s_x, s_y the standard deviations. (6.1)

If the Spearman correlation coefficient is positive this indicates that the y variable increases when x increases, while a negative coefficient indicates that y decreases. A perfect Spearman correlation, that is a test score of either -1 or $+1$, occurs when one variable is a perfect monotonic function of the other variable. As a rule of thumb Spearman's correlation coefficient can be interpreted according to table 6.1.

Table 6.1: Interpretation of the correlation

Size of correlation	Interpretation
1	Perfect correlation
0.90 - 0.99	Very high correlation
0.70 - 0.90	High correlation
0.50 - 0.70	Moderate correlation
0.30 - 0.50	Low correlation
0.10 - 0.30	Very low correlation
0.01 - 0.10	Markedly low and negligible correlation
0	No correlation

Source: Bansal et al. [3]

6.2 Significance

The calculation of the significance was used to determine if the result is indeed significant that means it has to be unlikely that the value could have occurred by chance. As significance level during the calculations a value of 5% ($\alpha = 0.05$), which is the most commonly used significance level in statistics, was used.

Using MATLAB's statistics toolbox to perform the calculations also provided the option to directly calculate the p-values that denote the probability of obtaining a test statistic that is at least as extreme as the observed one, under the assumption that the null hypothesis holds [82]. With the null hypothesis for the Spearman's rank correlation being that there exists no association between the variables in the underlying population.

6.3 Non-linear Least Squares Method

During the modeling a regression analysis was performed and since the collected data was mostly non-linear the non-linear least squares method is used to fit the non-linear model to the data. Although most of the calculations are transparent to the user when MATLAB's curve fitting toolbox is used, a good understanding of the method is still recommended as there exists no method to precisely estimate the coefficients of the model equation and thus often needs manual improvement of various fitting options to obtain the best results. This is mostly caused by the iterative approach that is required to estimate the coefficients as the non-linear model is approximated by a linear one [80].

To obtain the coefficient estimates, the least-squares method minimizes the summed square of residuals, where the residuals are the difference between the observed value (y_i) and the fitted value ($\hat{y}_i$) of the data points. The residual of a data point is also called the error associated with the data and its formula is shown in 6.2.

$$r_i = y_i - \hat{y}_i \tag{6.2}$$

The complete equation that has to be minimized is then the summed squares of all residuals as shown in 6.3 where n is the number of data points included in the fit.

$$S = \sum_{i=1}^{n} r_i^2 = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 \tag{6.3}$$

The minimum value of S is obtained when the gradient equations are zero, as shown in equation 6.4. Having m parameters in the non-linear model results in m gradient equations.

$$\frac{\partial S}{\partial B_j} = 2 \sum_i r_i \frac{\partial r_i}{\partial B_j} = 0 \qquad (j = 1, \ldots, m) \tag{6.4}$$

To solve those equations an iterative approach is used that follows the following steps:

1. Start with an initial estimate for each coefficient.
2. Linearize the model equation using a first-order Taylor series expansion as shown in equation 6.5.
3. Adjust the coefficients and determine whether the fit improves.
4. Iterate the process by returning to step 2 until the fit reaches a previously specified convergence criteria.

$$f(x_i, \beta) \approx f(x_i, \beta^k) + \sum_j J_{ij} \Delta \beta_j \tag{6.5}$$

where $f(x_i, \beta)$ is the non-linear model in matrix form with β being a vector of the coefficients in the equation. J is the Jacobian which is defined as a matrix of partial derivatives taken with respect to the coefficients, $\Delta\beta_j$ is the coefficient shift vector and k is the iteration step index.

6.4 Goodness of Fit

To determine the quality of the calculated curves, after curve fitting, the coefficient of determination (R^2) has been computed. R^2 is a common measure used with statistical models that have the purpose to predict future outcome, where R^2 provides a measure of how well the future outcome is likely to be predicted by the model [77].

The R^2 coefficient is calculated as shown in equation 6.6.

$$R^2 = \frac{SS_{reg}}{SS_{tot}} = 1 - \frac{SS_{err}}{SS_{tot}} \tag{6.6}$$

where $SS_{tot} = \sum_i (y_i \bar{y})^2$ is the total sum of squares, $SS_{reg} = \sum_i (\hat{y}_i \bar{y})^2$ the regression sum of squares and $SS_{err} = \sum_i (y_i \hat{y}_i)^2$ the sum of squares of residuals. $\bar{y}$ is the mean of the observed values and is calculated as $\frac{1}{n} \sum_i^n y_i$ with n being the number of observations.

6.4.1 Interpretation

The R^2 coefficient of determination is a statistical measure of how well the regression curve approximates the actual data points. Thus the coefficient gives some information about the goodness of fit of the chosen model. If the R^2 value is 1 this indicates that the regression perfectly fits the actual data while a value of 0 shows that there is no correlation between the regression and the data.

Chapter 7

Modeling

After the introduction to the mathematical methods in the previous chapter 6, this chapter discusses the application of those methods during the modeling, that was heavily based on the collected measurement data.

In the first section of the chapter the process of metric selection, including the requirements of the metrics and the results obtained during the analysis are presented, while the second part deals with the selection and fitting of the models.

7.1 Metric Selection

To determine the metrics that would be suitable for the further modeling process, the first step involved the determination of a set of requirements that the metrics had to fulfill. The criteria that have been established during that process are presented in the following list and are further elaborated in due course.

1. The metric had to be externally measurable without privileged access to the server or network.
2. The metric needed to be stable such that a measurement could be repeated multiple times with the same results.
3. The metric had to exhibit a measurable influence that was caused by the performed (D)DoS attack.
4. And the influence of the (D)DoS attack had to be related to the externally measured metric.

The first criteria was given by the task description and implied that only the metrics that in the previous sections have been classified as being external, should be used during the further modeling process. This allows the usage of the audit framework, developed during the thesis, on any Internet web server without requiring privileged access.

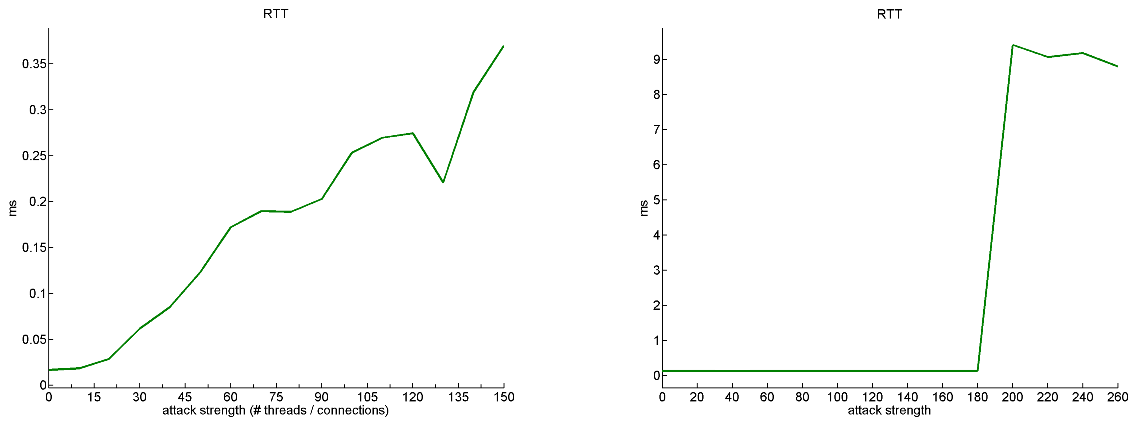
For the remaining analysis Spearman's rank correlation as explained in section 6.1 of chapter 6 was calculated with a significance level of 5% ($\alpha = 0.05$). The correlation coefficient was computed between the server-internal goodput metric and the externally measured metric that was to be evaluated. For both metrics their average values for each used attack strength have been employed during the correlation calculation. The goodput metric was used as according to [59] and [46] it is a suitable metric to measure the impact of (D)DoS attacks for Internet web servers and properly signals a DoS.

The Spearman rank correlation coefficients that are significant for all measurements are included in table 7.1 and are given in ranges since the attacks have been repeated at least three times, resulting in a correlation value for each metric and measurement. The correlation results are finally interpreted according to table 6.1 that is taken from the work by Bansal et al. [3].

The different behaviors between attack strength and measured metrics is also illustrated with the help of figure 7.1, where the round-trip time of the acknowledgment packets is shown versus the attack strength. As can be seen in the left figure, that shows the flash-crowd attack, the round-trip time increases almost linearly with the attack strength, whereas in the right figure the round-trip time remains constant up until the slow POST attack causes a DoS since the maximum number of parallel connections, that the server allows to be established, have been utilized.

This behavior was distinctly visible with the slow POST attack due to the fact, that the attack uses almost no resources on both client and server sides before the attack eventually reaches the maximum allowed number of parallel connections.

Because the maximum number of connections cannot be determined without actually reaching it, the slow POST attack cannot be used in a predictive model and thus was excluded from metric selection and the subsequent modeling.



(a) flash-crowd attack (b) slow POST

Figure 7.1: Attack impact on round-trip time

The UDP flooding with small packets as well as the GET flooding with database back-end showed no correlation and the correlation for the ICMP loss rate in the UDP flooding attack

Table 7.1: Metric correlation (Spearman's rho)

	UDP (large)	UDP (small)	Slow POST	Flash-crowd attack	GET wo. database	GET w. database
ICMP loss rate	0.366-0.678	-	*	-	-	-
ICMP RTT	-	-	*	0.636-0.980	0.636-0.877	-
ACK RTT	-	-	*	0.982-0.993	-	-
retr. segments	-	-	*	-	-	-
lost segments	-	-	*	-	-	-
TCP window	-	-	*	-	-	-

* excluded from modeling, as explained in the text

with large packets was only low to moderate according to the interpretation based on table 6.1. For the flash-crowd attack and the GET flooding without database the correlation was higher with values in the upper moderate to very high correlation range.

7.2 Model Fitting

After the metric selection process the metrics that have been determined as suitable for modeling have been used to establish predictive models.

7.2.1 Approach

As has been suspected and also has been confirmed in the previous metric selection section, the selected (D)DoS attacks had different attack points and thus also cause different impacts. Therefore it was concluded that the best solutions would be obtained when for each attack a separate model is established.

To find suitable models regression analysis was performed using the non-linear squares method as described in chapter 6. Using MATLAB's curve fitting tool allowed the rapid comparison of multiple models that were deemed to be appropriate and reasonable. In the following sections the individual attacks are looked at, and either the models that were found to be most suitable are presented or an explanation is given why no appropriate model could be established.

7.2.2 Flash-crowd attack

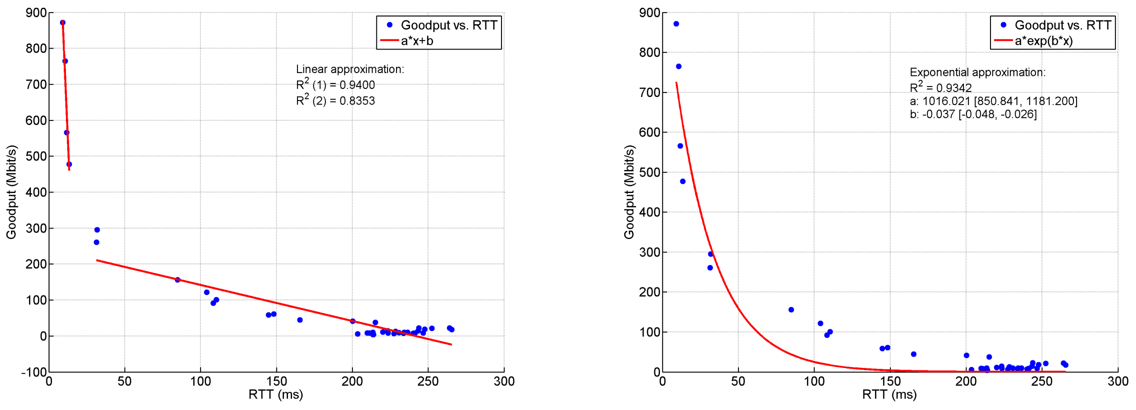
For the flash-crowd attack, the metric selection process has shown that the ICMP round-trip time metric as well as the metric of the acknowledgment packets' round-trip time can be used in a predictive model.

In figure 7.2 two models that have been fitted to the measurements are presented. On the left side of the figure a dual linear model is shown where the measured sample points have been divided into two regions and a linear approximation has been applied to both datasets. Since the assumed reason for the duality in the model, that was expected to be due to a configuration parameter, could not be hardened after additional modifications, the model was eventually rejected.

On the right side of the figure a further model is presented that uses an exponential approximation. In both figures the relationship between the ICMP round-trip time metric, which was measured using the ping utility, and the system state, that is approximated by the web server goodput, are shown.

The generic form of the used exponential model equation is shown in equation 7.1, while the actually used model equation had its parameter c set to 0 due to the fact that under an ideal attack the goodput would eventually reach towards 0.

$$a\exp(bx) + c \tag{7.1}$$



(a) dual linear approximation (b) exponential approximation

Figure 7.2: Flash-crowd models

To further analyze the selected model, various TCP models have been studied that partially explained the behavior between the measured parameters. Figure 7.3 includes a theoretical TCP model developed by Padhye et al., that has been looked at. The model has been taken from [53] and its there presented modeling equations have been implemented in a MATLAB script.

At first sight the model looked quite well, but during further analysis showed to make assumptions that did not hold in the lab network setup. One of those assumptions was that the round-trip time delay is mainly caused by the used network whereas the measurements in the lab network showed that the reason for the delays had to be in the web server since the delays in the network remained nearly constant during the flash-crowd attacks. Therefore the experimental model was preferred over a theoretical one for further measurements, during the prediction and error calculations.

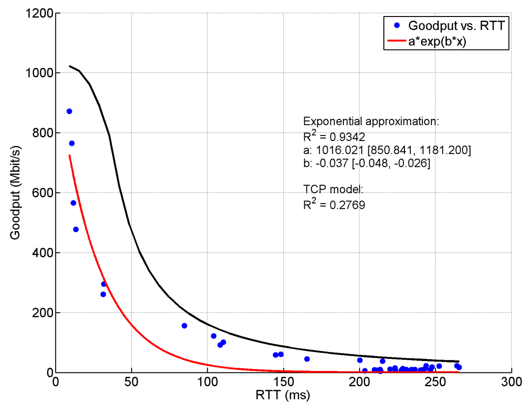


Figure 7.3: Exponential approximation & Theoretical TCP model

7.2.3 Slow POST

As has been explained previously no model that would serve to perform a meaningful impact prediction could be implemented for the slow POST attack. The reason is that a configuration option set on the Internet web server determines the limit where the attack is successful and as such an external attacker would have to probe the system until the attack causes a DoS to determine the configured limit.

Nevertheless the attack is implemented in the audit framework and as such can be used to determine if the web server is vulnerable to the slow POST attack. However the attack will result in a short DoS if the server is vulnerable and its configured limit of parallel connections is reached.

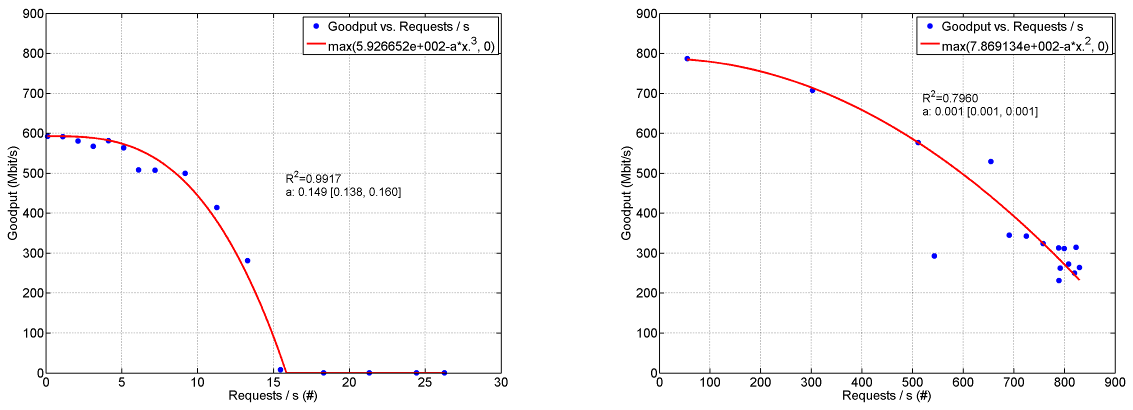
7.2.4 UDP flooding

The UDP flooding attack with large packets did not result in a DoS condition such that it was not further used. For the UDP flooding attack with small attacks no suitable metrics could be found during the metric selection process as has can be seen in table 7.1. As such no model was established for the UDP flooding attack with small packets.

7.2.5 GET flooding

Having a model for the flash-crowd attack the only remaining attacks that have not been looked at are the GET flooding attacks. According to the attack strength measurements in chapter 5, the GET flooding attack without database did not result in a DoS condition. For the GET flooding attack with database that showed to be successful in obtaining a DoS no external measurable metrics could be found as is shown in table 7.1.

Since GET flooding attacks are the most observed attacks in the Internet an alternative model that could also be based on internal metrics was looked for. The internal metric should be chosen such that there exists a chance to approximate the metric using external measurements. Due to the correlation with the attack strength the number of served requests per second, defined in table 4.1, was selected with a Spearman's rank correlation of 0.700 to 0.988, which is a high to very high correlation according to table 6.1. The process of approximating the served requests per second using additional external measurements is however not further explored in this report.



(a) with database (b) without database

Figure 7.4: GET flooding

In figure 7.4 models using polynomials for the GET flooding attacks are displayed. Although only a partial DoS was achieved during the GET flooding attack without database it is still included in the figure.

Chapter 8

Evaluation

In this chapter the results of the evaluation are summarized. The first section introduces the network and system setup variations that have been analyzed during the thesis, presenting the adapted model and error plots. The prediction and its associated error calculations are finally presented in the second part of the chapter.

8.1 Variations in the Setup

In parallel to the previously introduced modeling the evaluation of the models has been prepared, where several network and web server system parameters have been varied. The goal was to analyze the influence on the model caused by the network and system properties presented in the following list.

- Network with low throughput
- Network with latency variations
- Network with additional utilization of the bottleneck link
- System with additional web server utilization
- System with low disk performance
- System with modified web server software

To simulate the influence of a network link with low throughput, the gigabit capable access link of the client system was reduced to a 100 Mbit/s link using the `ethtool` tool, as has been explained in the lab network setup in 4.1. The latency variations in the network were introduced by redirecting the client measurement traffic through the productive network and into a VPN tunnel that ended on the web server system. As such not only the latency variations have been increased but also additional round-trip delay was added to the network connections.

The utilization of the bottleneck link was increased with additional TCP network traffic (also called cross traffic) that has been generated between two additional hosts, as shown in figure

4.1, on either side of the bottleneck link. To generate the simulated TCP traffic the flow-level traffic generator harpoon has been used which is explained in the appendix A.1.1.

To increase the utilization of the web server an additional system was prepared that acted as a normal client using the web server. The client was automated using Selenium which is an automated web application testing system [72]. Using Selenium's IDE, tests can quickly be recorded in the browser and played back at a later stage. The recorded test included the visiting of multiple sub pages and the downloading of files from the web server. The test was continuously repeated during the attack.

The low disk performance of the web server was simulated by using an external USB 2.0 drive instead of the internal hard-drive. The USB drive stored the contents accessible by the web server as well as the complete database. To change between the disk and the USB content a symbolic link was created between the web server content on the USB drive and the content on the usually used disk drive. The database software showed to be picky as it did not accept symbolic links in its data path such that finally the `mount --bind` command, which remounts part of the file hierarchy somewhere else, had to be used for the database data instead of a symbolic link.

To analyze the influence of the web server software the Apache 2 web server [65], that has been used during all previous measurements, was replaced with the lighttpd web server [40], keeping the same web server data.

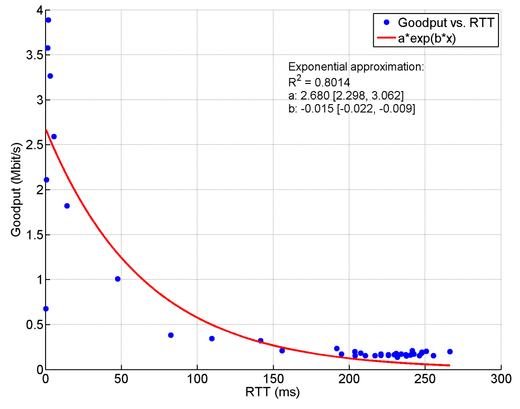
In addition to the changes on the server system a second server system was taken into consideration. Due to the fact that both systems used the same software and configuration and since both servers used recent processors and plenty of memory those variations were not expected to influence the measurements heavily. Also the influence of the used SSD drive could only be little since the speed of the previously used disk drive was already near the bottleneck link speed as is shown in the results below.

The models have been calculated using the complete measurement data, ranging from no attack being present up to at least the point where the given attack strength caused a DoS condition.

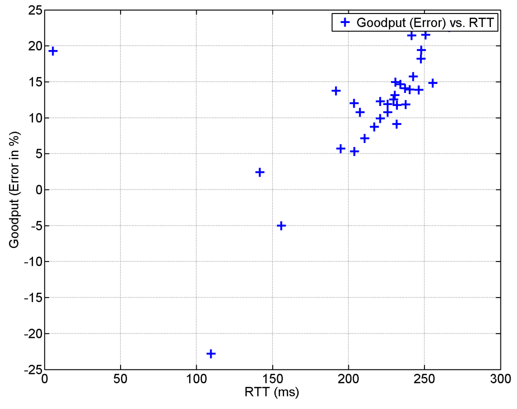
8.1.1 Results

The model plots are presented in figure 8.1 and table 8.1 summarizes the calculated R^2 as well as error values shown to the right of the model plots. The error is calculated between the model and the actually measured data, as a percentage of the maximally observed goodput during each measurement.

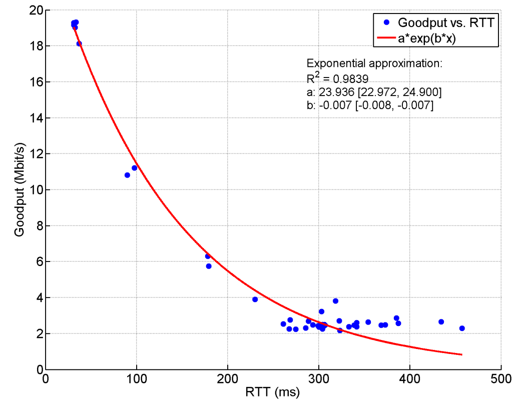
In the presented model plots various bottlenecks that have been introduced by the setup variations can be identified when comparing each of the shown model plots to the original model in figure 7.2b.



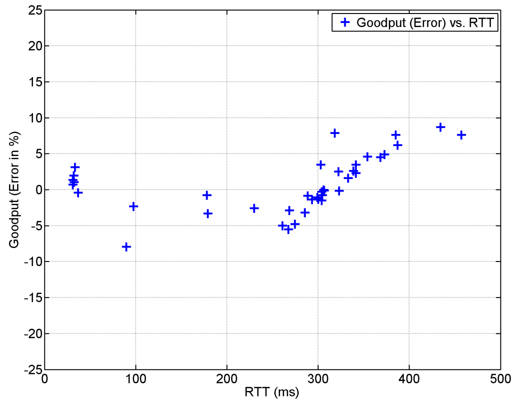
(a) 100 Mbit/s link on client



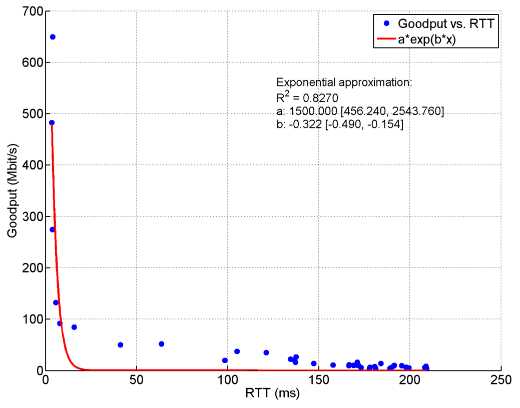
(b) 100 Mbit/s link on client (error)



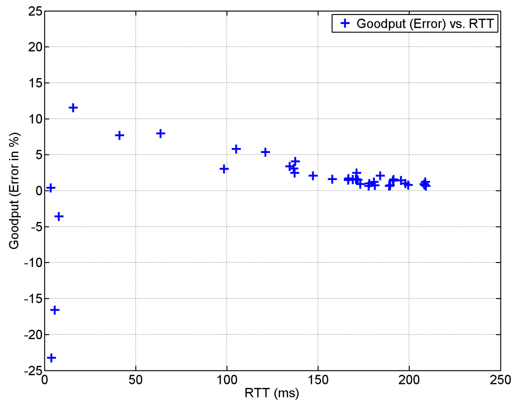
(c) VPN tunnel



(d) VPN tunnel (error)

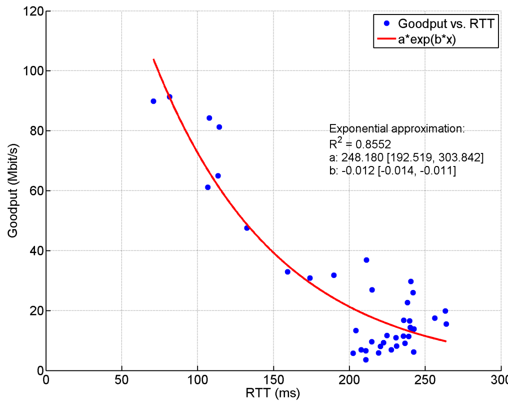


(e) 200Mbit/s cross traffic

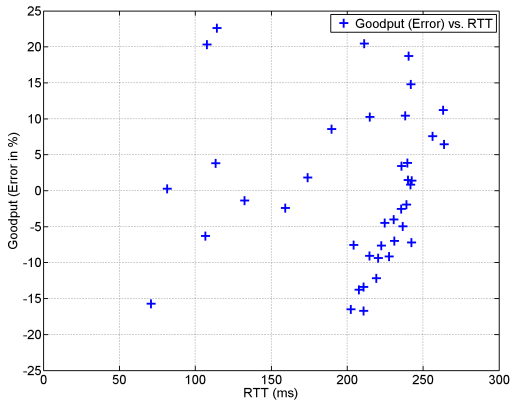


(f) 200Mbit/s cross traffic (error)

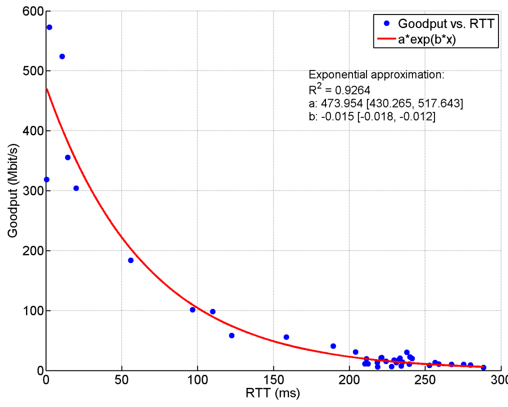
Figure 8.1: Parameter variations during flash-crowd attacks



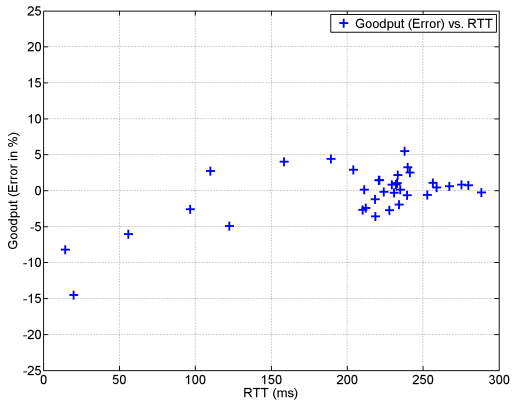
(g) additional client traffic



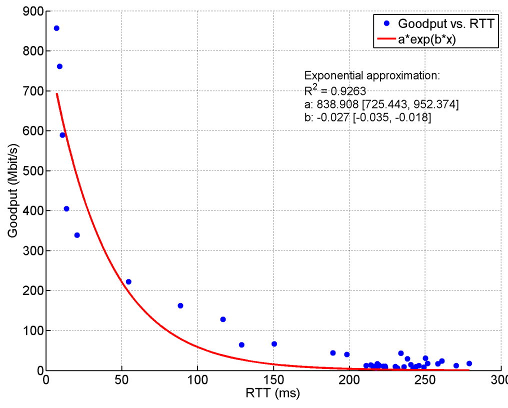
(h) additional client traffic (error)



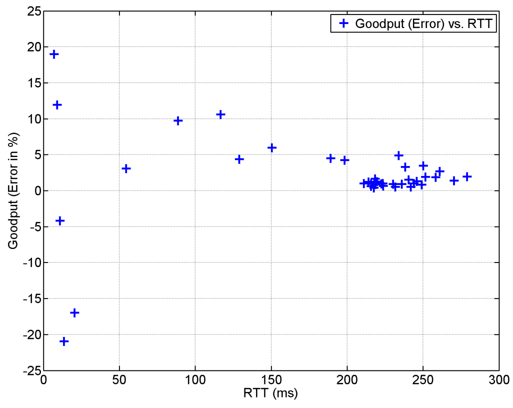
(i) USB drive



(j) USB drive (error)

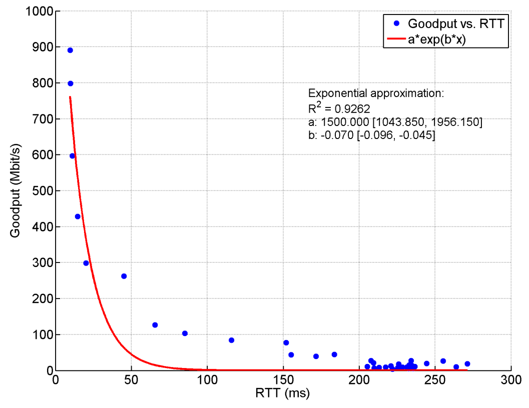


(k) lighttpd

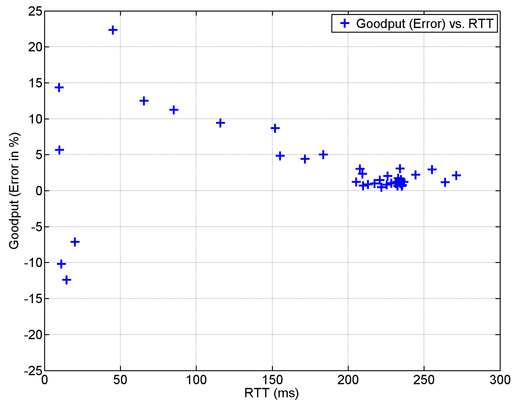


(l) lighttpd (error)

Figure 8.1: parameter variations during flash-crowd attacks (continued)



(m) second server



(n) second server (error)

Figure 8.1: parameter variations during flash-crowd attacks (continued)

Table 8.1: Parameter variation errors

Variation	R^2	Error (%)		
		avg	min	max
100 Mbit/s link on client	0.8014	34.48	2.43	294.25
VPN tunnel	0.9839	3.06	0.04	8.69
200Mbit/s cross traffic	0.8270	4.19	0.38	37.88
additional client traffic	0.8552	8.54	0.27	22.60
USB drive	0.9264	5.41	0.11	47.42
lighttpd	0.9263	3.90	0.38	20.95
second server	0.9262	4.17	0.47	22.32

Starting with the bottleneck caused by the disk performance one notes that during the original measurements the disk has not been observed as a major limiting factor. Indeed, verifying the performance of the disk drive using `hdparm -t /dev/sda` to measure the sustained speed of sequential data reads, shows that the disk drive is capable of reading data with a constant rate of over 125 MB/s and thus near the throughput of the bottleneck link. In contrast the USB drive as shown in figure 8.1i is a clear bottleneck and limits the maximally obtainable goodput during the measurement.

The VPN tunnel modification in figure 8.1c introduces additional round-trip time delay as has been expected and noted during the setup.

As can be seen in figure 8.1e the cross traffic variation narrows the available bottleneck link capacity however there is no additional round-trip time introduced, whereas in figure 8.1g, where additional client traffic was generated, also the round-trip time is heavily influenced right from the start of the measurement. This also supports the previously made statement, that the observed round-trip time delay is not due to the lab network but the web server system.

The web server software modification in figure 8.1k shows little change compared to the original mode that used Apache 2 as its server software.

As forecast, the second server measurement in figure 8.1m shows that the previously used disk caused only a slight bottleneck, that is now finally lifted by the usage of the SSD drive with a continuous data read speed of over 220 MB/s. The relevant bottleneck for the measurement is thus the bottleneck link capacity that is just slightly higher than the previously used disk drive.

The average error between the model and the measured data is below 10% for all setup variations except for the case where the access network link speed of the client was reduced to 100 Mbit/s. The high error rate that is observed is due to the fact that the throughput that actually passes the bottleneck link, which clearly is the access link of the client, is reduced much more than expected by a factor of over 200 instead of the assumed factor of around 10. The reason for this discrepancy remains unknown.

Further considering the error plots, it is visible that the error for the client access link variation as well as the error for the additional client traffic variation are not centered and clustered around the zero line. This indicates that the influence caused by those variations will most likely also increase the error rate of the prediction.

8.2 Prediction & Error Calculations

While the previously presented models have been computed using training data that was captured with zero attack strength up to at least the attack strength that caused a DoS condition, the actual predictions would only be based on testing data with limited attack strength.

Thus for the calculations of the prediction accuracy or in other terms the error associated with the strength of the attack, a further test data set, that contained repetitions of the training measurements used to establish the models, was collected. Once again the attack strength of the measurements had to last up to the point of the DoS condition such that the error can be computed for the complete attack strength range.

The results obtained during those calculations thus serve to trade off the required attack strength with the accuracy of the prediction. As such the influence on the web server under consideration as well as any additionally involved systems and networks can be reduced by reducing the accuracy of the prediction.

To calculate the prediction error for a particular attack strength, the test data was shortened such that it only contained the measurements up to the examined attack strength. Using the reduced data, the prediction model was established using the techniques previously presented in chapters 6 and 7. The errors have then been calculated for all existing measurement points comparing the values returned by the prediction model with the corresponding values calculated using the verified model.

To allow the comparison between the different setup variations, the prediction error was calculated as a percentage of the maximally recorded goodput during the measurement, similar to the error calculations presented in the previous section 8.1 during the setup variations.

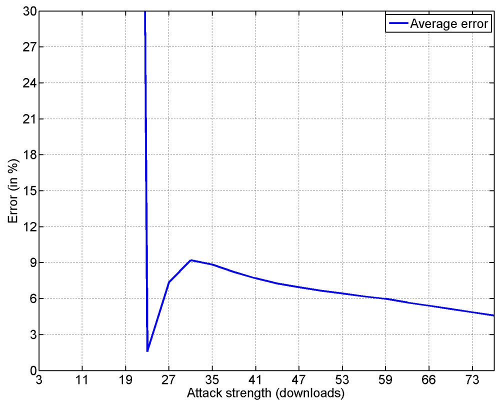
8.2.1 Results

The results of the calculations are given in figure 8.2, where the plots illustrate the progression of the average error, computed according to the previous description, with the increase of the maximally used attack strength for the training data.

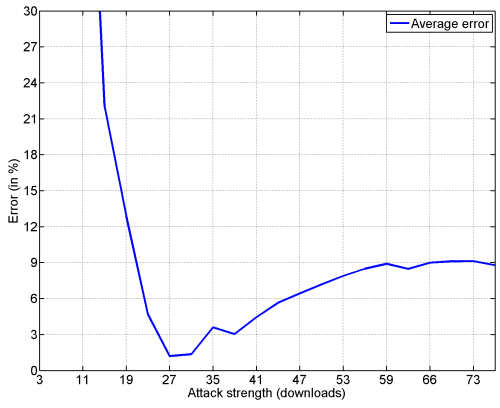
For most of the scenarios the error rate rapidly declines while the maximally used attack strength increases, as has been wished for. After this period that according to the plots lasts up to around a third of the maximally used attack strength, the error rate becomes constant for most measurements. An exception is the VPN tunnel where the error rate starts again to increase before finally becoming constant at an error rate level similar to the other variations involving network components.

As has been suspected previously there are a few variations that incur higher error rates than others. As such all measurements where network parameters were involved experience error rates that are 3 times as high as the error rates of the variations that included only system hardware and software changes.

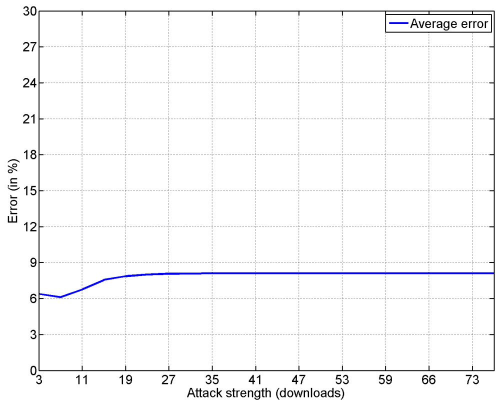
As noted in the «Problem Statement» chapter 2, the goal was to achieve an error rate of less than 10% while keeping the required attack strength within 25 to 50% of the strength required to cause a DoS. According to the calculations this requirements have been fulfilled with an attack strength of around 30% being suitable to achieve an error rate around 2-3% for the software variations and 7-9% for the network variations.



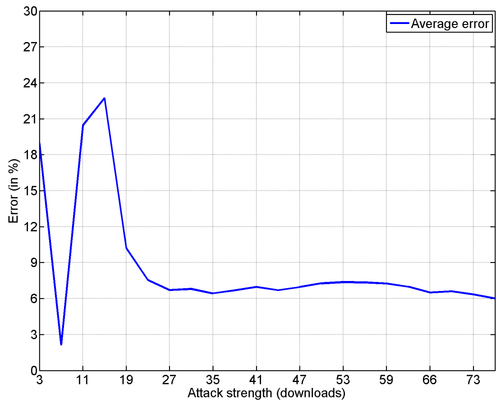
(o) 100 Mbit/s link on client



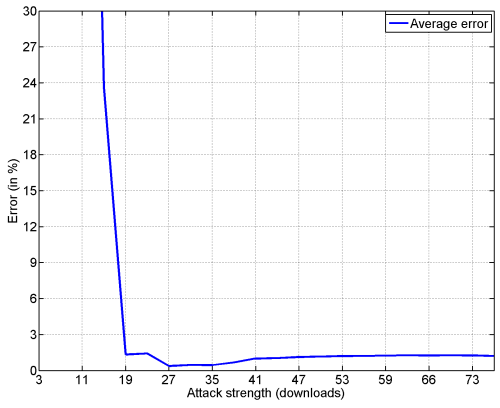
(p) VPN tunnel



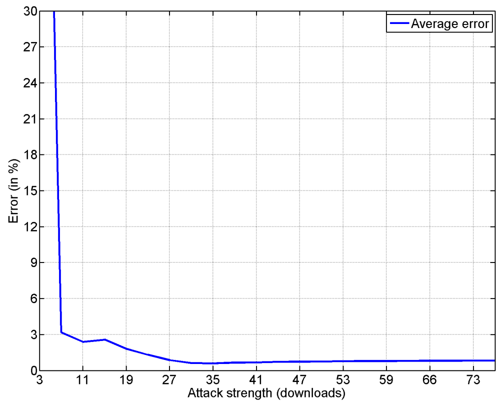
(q) 200Mbit/s cross traffic



(r) additional client traffic

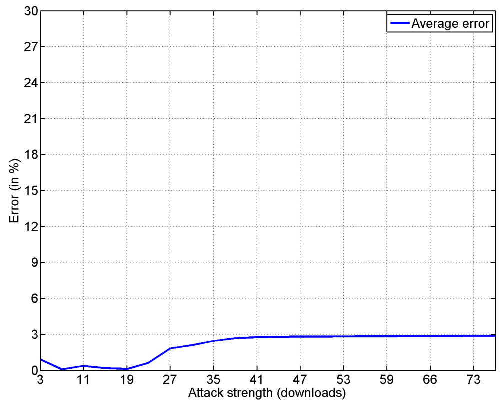


(s) USB drive



(t) lighttpd

Figure 8.2: Prediction errors during flash-crowd attacks



(u) second server

Chapter 9

Audit Framework

The developed auditing method has been implemented in a framework that allows the auditing of any Internet web server using a predefined set of (D)DoS attacks.

9.1 Design

The audit tool (`audit.py`) is the central control program that interacts with the client measurement host, which may be the current local machine or a remote one, and the attacking hosts carrying out the selected (D)DoS attack. During the thesis additional measurements have been performed on the web server system such that the framework is also capable of controlling the web server host. The interaction between the different components is illustrated in figure 9.1.

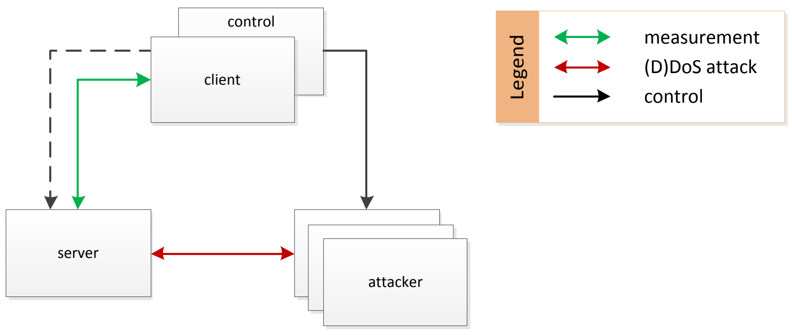


Figure 9.1: (D)DoS attack audit framework

The framework is written in Python and makes use of the «paramiko» [55] SSH2 protocol library during the deployment and collection stage. In the deployment stage the used (D)DoS attack tools are transfered to the attacking hosts and the measurement tools to the client host. After the measurement completes, the measurement data is collected in the collection stage and stored on the control host.

9.2 Requirements

To use the auditing framework a client, to perform the measurement, and at least one attacking host are required, both running a Linux operating system. The tool expects the IP addresses, usernames and passwords for the different systems. As a constraint when multiple attacking hosts are being used, they have to share the same username and password. Depending on the performed attack, the user used to login on the attacking hosts requires super user privileges. To prevent password prompts, when requesting the privileges using `sudo`, from interfering with the attacks, the used systems in the lab environment were configured such that no additional password check was required. To enable super user privileges without password for the user «server» the line shown in 9.1 was added to the `/etc/sudoers` file using `sudo visudo`.

server ALL=(ALL) NOPASSWD: ALL (9.1)

9.3 Configuration Assistant

When the audit tool is launched without parameters, a configuration assistant appears that guides the user through the required steps to configure the framework. The workflow of the configuration assistant is presented in figure 9.2. Since the auditing framework has also been used to perform measurements during the thesis evaluation it allows further configuration options such as the measurement of internal metrics on the web server or the execution of additional scripts, e.g. to generate traffic on the bottleneck link.

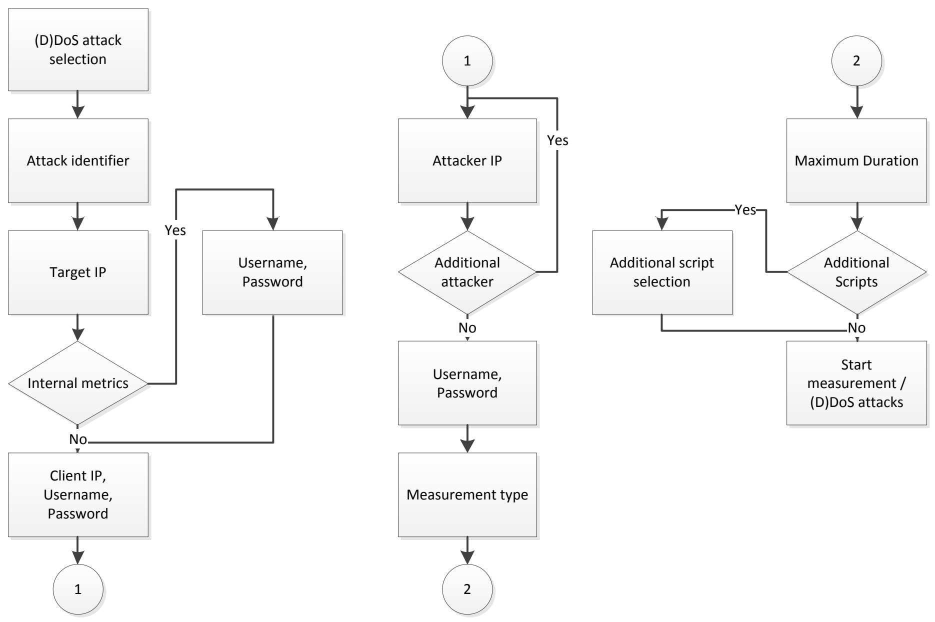


Figure 9.2: Configuration assistant

9.4 Workflow

After the configuration and setup, the initial measurement is run while no attack is present. Afterwards the intermediate data is processed to determine the current server condition before launching the attack. The attack strength is increased gradually until the time limit or the previously set limit of performance deterioration on the web server is reached. When the attack is finished the measurements are completed and the data collected. The collected measurement data is processed and in a last step MATLAB code to further analyze the gathered data is generated.

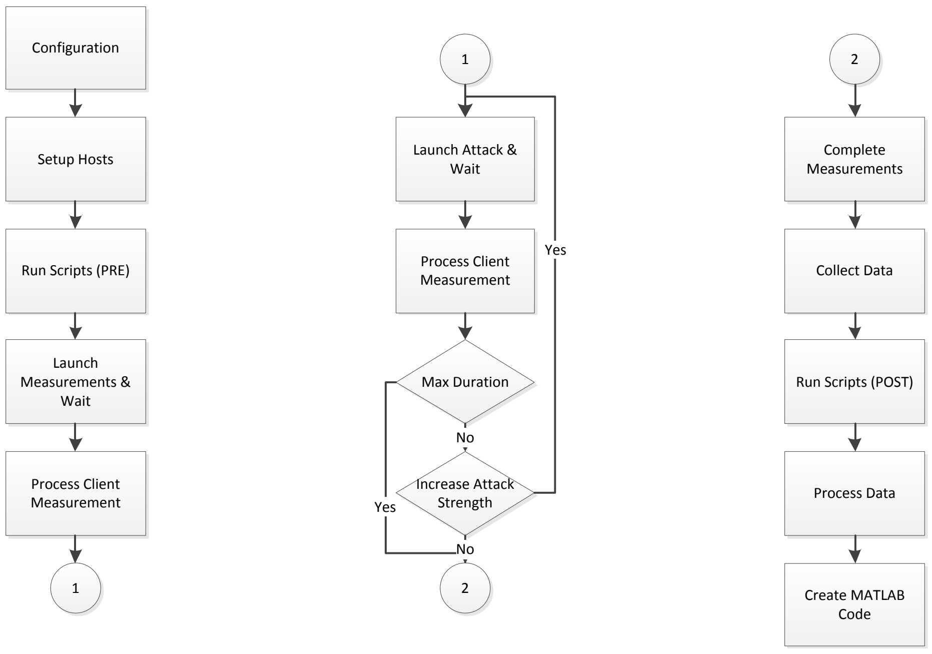


Figure 9.3: Audit framework workflow

9.5 Extensibility

The available (D)DoS attacks during the attack selection process can easily be extended since the available attacks are directly loaded from the file structure. To add an additional attack a new folder, where the name consists of an unused attack number, is added under the directory `tools/attacker`. The folder should contain a `description.txt` file, containing a short title or description on the first line, which is shown during the attack selection process. On the subsequent lines of the file the full description of the attack can be given but will not be shown.

If a file named `attacker.tgz` exists in the folder it will be transfered to the attacking hosts before the attack. In addition each attack requires a `prepare_attack.sh` file, that is also transferred and executed once before the attack. The script is responsible to install the required software on the hosts, extract the uploaded archive, configure the system for the attack and prepare the attack tools. Before the script is executed on the server the first occurrences of `<SERVER_IP>`, `<ATTACKER_ID>`, `<ATTACKER_MAX>` as well as `<ATTACKER_DUR>` and `>STEP_DUR>` are replaced with the target IP, the number of the current attacker and the total number of attackers as well as the total duration of the attack and the duration of the current step. After the script completes its execution, an executable `attack.sh` file should be present in the users home folder, that launches the (D)DoS attack when being executed.

To enable the generation of MATLAB code, the attack folder has to contain a `template.m` file where the variables `<ATTACK_DUR>`, `<STEP_DUR>` and `<ATTACK_ID>` are replaced with their actual values.

Similarly to the (D)DoS attacks, the available scripts that are executed before and after the attack can be extended, by placing a `script_pre.sh` and / or `script_post.sh` in the numbered subdirectories of `tools/additional`.

Chapter 10

Conclusion

10.1 Summary

We developed a new auditing method to externally assess and predict the impact of (D)DoS attacks on web servers based on low strength (D)DoS attack measurements. The method does not require the audited system and its networks to be completely flooded with attack traffic and thus can also be used in a productive environment with critical infrastructure. Using a pre-established model for a (D)DoS attack, its parameters are approximated with the data collected during the assessment. The model can then be used to infer the impact of a similar attack at a stronger attack strength.

To model the impact of (D)DoS attacks, we examined appropriate metrics that characterize the performance of a web server and can be measured externally without privileged access to the system and its network. Based on the results of the measurements selected (D)DoS attacks have been employed to establish predictive impact models which have been evaluated using extensive variations in software and hardware of the web server as well as the intermediate network.

Calculating the error rate between prediction and actual measurements the accuracy of the method has been verified resulting in an expected error rate of 10% at a limited attack strength of 30% of the strength causing a DoS.

As a prototype an audit framework was developed, which implements the presented auditing method. It allows anyone to assess the impact of a (D)DoS attack on any web server in the Internet.

10.2 Conclusion

The impact of (D)DoS attacks is highly dependent on the attack type and the targeted property of the system or network. We showed that different models are required to predict the impact of the selected (D)DoS attacks, however not all (D)DoS attacks can be modeled due to different constraints. For once there are attacks that impact the system only marginally or not at all, the reasons are divers, such as the used software on the targeted system or the network layout and infrastructure. On the other hand are the attacks that provoke a sudden (D)DoS condition that is not gradually achieved, mostly due to a configured limit. Also to use a metric in the (D)DoS attack models, certain requirements have to be fulfilled.

We showed that the models are not only dependent on the analyzed (D)DoS attack, but also on the bottlenecks of the employed system and network setup. While the attack defines the general equation of the model, the bottlenecks influence the actual parameters in the model equations.

The developed auditing method serves as an alternative to existing (D)DoS auditing methods that cannot be used in productive environments with critical infrastructure due to the required flooding and resulting outages that cannot be tolerated. Frequent and regular (D)DoS attack auditing allows to improve the host and network stability and prepare for attack mitigation. The preparation for attacks can thus reduce the down-time and resulting losses caused by unexpected (D)DoS attacks.

We showed that is possible to predict the impact of selected (D)DoS attacks on Internet web servers based on a few characteristic metrics which can be measured externally. The usage of externally measurable metrics ensures that any Internet web server can be measured without interfering with the setup of the audited system, e.g. requiring additional tools to be deployed.

10.3 Future Work

10.3.1 Productive Environment

Due to the nature of the work, most of the conducted measurements during this thesis had to be performed in an isolated lab network with limited access to the Internet. Although the effort to simulate a productive network with influences caused by additional clients and network conditions has been taken, it would be interesting to make further evaluations in additional productive environments.

10.3.2 Additional Attacks

Multiple (D)DoS attacks have been modeled and analyzed, having additional attack models would however give important insight into the differences between the attacks and the usage of the presented method especially since multiple attacks showed to be less suitable to be modeled by the presented method.

10.3.3 Additional Metrics

The quality and accuracy of the predictions is based on the (D)DoS attack models, that are themselves based on the selected metrics and their quality. The usage and evaluation of further metrics could thus improve the impact predictions.

10.3.4 Internal Metrics

During the thesis an auditing method that can be performed externally without having access to the targeted systems has been developed. If this restriction is lifted additional internal metrics, that cannot be approximated externally, can be used that may result in improved predictions.

Chapter 11

Zusammenfasung

Im Rahmen der vorgestellten Masterarbeit wurde eine neue Methode der Auditierung entwickelt welche die Auswirkungen von Denial of Service Angriffen auf Internet Web Server anhand externen Messungen voraussagen kann. Bei der entwickelten Methode wird der getestete Server sowie das Netzwerk nicht vollständig ausgelastet und die Methode kann deshalb auch in einer produktiven Umgebung mit kritischer Infrastruktur eingesetzt werden. Die Parameter der vordefinierten Modelle für die Denial of Service Angriffe werden anhand von kurzen und abgeschwächten Messungen bestimmt um anschliessend mit Hilfe der Modelle die Auswirkungen eines ähnlichen Denial of Service Angriffs bei grösserer Stärke vorauszusagen.

Um die Auswirkungen von DoS Angriffen zu modellieren wurden geeignete Metriken die den Zustand eines Web Servers erfassen und welche ohne privilegierten Zugriff auf das System oder Netzwerk von extern messbar sind bestimmt. Basierend auf den ermittelten Metriken wurden ausgewählte (D)DoS Angriffe modelliert und die resultierenden Modelle wurden anhand umfangreicher Variationen auf Software- sowie Hardwareebene des Web Servers und des Netzwerks evaluiert.

Indem die Fehlerrate zwischen der Voraussage und den tatsächlichen Messungen berechnet wurde, wurde die Genauigkeit der entwickelten Methode verifiziert und es zeigte sich, dass sich bei einer DoS Angriffsstärke von 30% des Wertes der zu einem DoS Zustand führt eine Fehlerrate von weniger als 10% erreichen lässt.

Als Prototyp wurde ein Softwarepaket zusammengestellt und erweitert, welches die vorgeschlagene Auditierungsmethode implementiert. Die Software erlaubt es einem beliebigen Benutzer einen Web Server im Internet auf seine Anfälligkeit auf Denial of Service Angriffe zu testen.

Appendix A

Appendix

A.1 Attack Tools

The following section presents some additional information to the used tool during the attacks.

A.1.1 harpoon

harpoon is a tool to generate simulated traffic according to a distribution which can be produced by previously monitoring traffic [62]. The traffic is generated between two systems and has been used to generate cross traffic on the bottleneck link.

```
./run_harpoon.sh [-v<level>] -w<interval> [-c] -f <configuration>
    -v<verbosity level>
    -w<interval duration>
    -c Continuously cycle over list of active connections
    -f <configuration file>
./run_harpoon.sh -v10 -w120 -c -f examples/tcp_server.xml
```

A shortened configuration for client and server can be found in the Configuration section of appendix A.2.1. To determine the number of active connections required to generate a specific amount of traffic harpoon comes with a configuration utility `harpoon_reconf.py`.

```
./harpoon_reconf.py -s tcp_server.xml -c tcp_client.xml -i 120
    -r 200000000
```

A.1.2 Twisted

Twisted is an event driven networking engine written in Python that makes it easy to implement custom network applications [73]. During the thesis Twisted has been used to program a

GET flooding application. The core of an event loop within Twisted is the so called reactor. By default Twisted uses a reactor that relies on the `select(2)` functionality for synchronous I/O multiplexing [23]. In Ubuntu a user is limited to having 1024 concurrent open files which can however be increased using the «ulimit» command [8]. More importantly though the `select` functionality is also limited to 1024 concurrent open files itself. On certain operating systems (e.g. FreeBSD) this setting can be overridden in each program by redefining the `FD_SETSIZE` in the source code of a program. However this is apparently not possible in the used Ubuntu operating system.

Subsequently as an alternative to the default Twisted reactor the pollreactor was used that handles the concurrent connections using polling and as such was able to handle slightly more parallel connections.

A.1.3 netcat (nc)

The netcat utility is a versatile tool when dealing with TCP or UDP and is commonly used as shell-script based HTTP clients [29]. Using the netcat utility previously recorded requests can be retransmitted at very high request rates especially when the returned output is of no interest.

```
./nc 192.168.74.10 80 < requests/1.txt
```

A.1.4 ncat

ncat is part of the Nmap security scanner suite and serves as a replacement for netcat to concatenate and redirect sockets [34]. By default the ncat tool returns the output of the request, however using the `-send-only` option the output can be discarded.

```
./ncat 192.168.74.10 80 < requests/1.txt
./ncat --send-only 192.168.74.10 80 < requests/1.txt
```

A.2 Configurations

In this part an excerpt of the used harpoon configuration as well as the configuration of the pktgen tool are presented.

A.2.1 harpoon

harpoon client configuration

```
<harpoon_plugins>
  <plugin name="TcpClient" objfile="tcp_plugin.so" maxthreads="146" personality="
      client">
       <active_sessions> 146 </active_sessions>
       <interconnection_times>
3.993905
0.293601
2.127093
1.214513
0.409159
0.112100
...
0.402745
0.322681
0.227759
       </interconnection_times>
       <address_pool name="client_source_pool">
            <address ipv4="0.0.0.0" port="0" />
       </address_pool>
       <address_pool name="client_destination_pool">
            <address ipv4="192.168.74.27/32" port="10000" />
       </address_pool>
  </plugin>
</harpoon_plugins>
```

harpoon server configuration

```
<harpoon_plugins>
  <plugin name="TcpServer" objfile="tcp_plugin.so" maxthreads="160" personality="
      server">
      <file_sizes>
18643910
15150
807481
157679
23465
4930
...
39188
4418
68459
      </file_sizes>
      <active_sessions> 160 </active_sessions>
      <address_pool name="server_pool">
          <address ipv4="0.0.0.0" port="10000" />
      </address_pool>
  </plugin>
```

```
</harpoon_plugins>
```

A.2.2 pktgen

pktgen configuration

```
#!/bin/bash

function pgset() {
  local result

  echo $1 > $PGDEV

  result=`cat $PGDEV | fgrep "Result: OK:"`
  if [ "$result" = "" ]; then
    cat $PGDEV | fgrep Result:
  fi
}

function pg() {
  echo inject > $PGDEV
  cat $PGDEV
}

# ---------- Config ----------
# thread config
PGDEV=/proc/net/pktgen/kpktgend_0
 echo "Removing all devices"
 pgset "rem_device_all"
 echo "Adding eth2"
 pgset "add_device eth2"

# device config
# set to 0 when DoS testing
CLONE_SKB="clone_skb 0"
# NIC adds 4 bytes CRC
PKT_SIZE="pkt_size 1500"

# COUNT 0 means forever
COUNT="count 0"
DELAY="delay ${1}"

PGDEV=/proc/net/pktgen/eth2
 echo "Configuring $PGDEV"
 pgset "$COUNT"
 pgset "$CLONE_SKB"
 pgset "$PKT_SIZE"
 pgset "$DELAY"
 pgset "dst <SERVER_IP>"
 pgset "dst_mac <MAC>"

 pgset "udp_src_min 1025"
 pgset "udp_src_max 65000"
 pgset "udp_dst_min 80"
 pgset "udp_dst_max 80"
 pgset "src_min <SRC_IP>"
 pgset "src_max <SRC_IP>"

 pgset "flag UDPSRC_RND"

# Time to run
PGDEV=/proc/net/pktgen/pgctrl

 echo "Running... ctrl^C to stop"
 pgset "start"
 echo "Done"

# Result can be vieved in /proc/net/pktgen/eth2
```

A.3 Code

A.3.1 MATLAB

Table A.1: Description of used MATLAB files

Scripts:	
attack_common.m	Script that handles the common processing and plotting.
attack_common_2.m	Script that handles the preprocesses «tcpdump» data.
attacks/	
attack_<id>_<description>.m	Script that sets the attack parameters and settings to be used when plotting.
models/	
model_flash_crowd.m	Calculate Flash-crowd model.
model_flash_crowd_error.m	Calculate Flash-crowd error of prediction.
model_get_flood_wi_mysql.m	Calculate GET flood with MySQL model.
model_get_flood_wo_mysql.m	Calculate GET flood without MySQL model.
model_get_flood_wo_mysql_error.m	Calculate GET flood without MySQL error of prediction.
model_udp_flood.m	Calculate UDP Flooding model.
tcp_bw.m	MATLAB implementation of formulas in «Modelling TCP throughput: A simple model and its empirical validation» by J. Padhye, et al, SIGCOMM 1998.
tcp_bw_mathis.m	MATLAB implementation of formulas in «The macroscopic behavior of the TCP congestion avoidance algorithm» by Mathis, et al. CCR 27(3), July 1997.
script_tcp_bw.m	Calculate and plot the TCP bandwidth according to the above formulas.

Functions:

Data access & Preprocessing:

stats_<type>.m	Process the saved measurement data. The types are «access_log», «connections», «connections_ss», «curl», «iostat», «iperf», «mem», «mem_free», «monitor», «mpstat», «netstat_s» and «rtt».
read_curl_file.m	Helper function to process the output of the «curl» program.
new_netstat_format.m	Helper function to identify «nestat» output layout.

Aggregation:

eval_<type>.m	Aggregate the preprocessed data and prepare for plotting. The types are «connections», «connections_ss», «cpu», «curl», «io», «iperf», «memory», «memory_free», «netstat_s» and «rtt».

Calculations & Plotting:

calc_attack_strength.m	Extract the attack strength from the measurement.
calc_stat.m	Calculate various statistics on the preprocessed measurement data.
calc_strength.m	Process the output of the «pktgen» program to calculate the actual attack strength.
calc_strength_curl.m	Process the output of the «curl» program to calculate the actual attack strength.
metric_analysis.m	Perform the metric selection analysis.
percentage.m	Calculate a percentage using the given values.
rsquare.m	Calculate R^2 using y and $\bar{y}$.
plot_stat.m	Plot the calculated statistics.
jbfill.m	Fill a region between two vectors with a color.

Misc:

figpref.m	Set figure preferences.
print_figure.m	Print figure to a file.
mat2unixtime.m	Convert MATLAB date to Unixtime.
unixtime2mat.m	Convert Unixtime to MATLAB date.
grep.m	Unix-like grep implementation in MATLAB.
uniquify.m	Ensures a given set of values is unique.
read_first_line.m	Access the first line of a file.
read_last_line.m	Access the last line of a file.
xml2struct.m	Read an XML file into a MATLAB structure.

A.3.2 Framework

Table A.1: Description of developed measurement framework

./	
audit.py	
DoS_attacker.py	
DoS_client.py	
DoS_config.py	
DoS_server.py	
paramiko_helper.py	Helper functions for the paramiko SSH2 protocol library.
post_process.py	Data processing and MATLAB code generation.
validator.py	Helper function to validate entered IP addresses.
./tools/	
additional/	
<id>/	
description.txt	Title and description of additional task to execute before and/or after measurements.
script_pre.sh	Script executed before measurement.
script_post.sh	Script executed after measurement.
attacker/	
<id>/	
attacker.tgz	Archive to be uploaded to the attackers.
attack.sh	Script executed on the attacker to start the attack (has to be contained in the attacker.tgz archive or created by the prepare_attack.sh script).
description.txt	Title and description of attack.
template.m	MATLAB code template.
prepare_archive.sh	Script to prepare attacker.tgz archive.
prepare_attack.sh	Script executed on the attacker before the measurement.
client/	
client.tgz	Archive to be uploaded / moved to the client.
measure_client.py	Script executed on the client to start the measurement (has to be contained in the client.tgz archive or created by the prepare_client.sh script).
prepare_archive.sh	Script to prepare client.tgz archive.
prepare_client.sh	Script executed on the client before the measurement.
server/	
measure_server.py	Script executed on the server to start the measurement.
prepare_server.sh	Script executed on the server before the measurement.

Appendix B

Time Schedule

13.02.2012	study related work including impact metrics, (D)DoS attacks
week 2	additional related work, summarize impact metrics, prepare time schedule
week 3	determine suitable network, end-to-end and (D)DoS attack tools
week 4	implement required tools / prepare lab
week 5	lab measurements
week 6	evaluate correlation between metrics and attack parameters using statistics
week 7	implement required tools / lab measurements
week 8	preliminary model
week 9	discussion with supervisor (major aspects of ongoing work)
week 11	
week 13	final model
week 15	evaluate finding in testbed scenario
week 16	preliminary report
week 17	discussion with supervisor (table of content / preliminary written report)
week 18	verify findings in real world scenario
week 23	final report
week 24	prepare final presentations
12.08.2012	

Bibliography

[1] Dmitri Alperovitch. Revealed : Operation Shady RAT. pages 1–14. URL http://www.mcafee.com/us/resources/white-papers/wp-operation-shady-rat.pdf.

[2] Arbor Networks. Worldwide Infrastructure Security Report. Technical report, 2011.

[3] I.K. Bansal, K.K. Vasishtha, T.C. Gyanani, M. C. Sharma, Snehlata Shukla, and Sunaina Kumar. Correlation - Its Interpretation and Importance. In *Statistical Techniques of Analysis*. Indira Gandhi National Open University (IGNOU), 2008. URL http://www.egyankosh.ac.in/handle/123456789/25740.

[4] J. G. Beerends and L. A. R. Yamamoto. Impact of network performance parameters on the end-to-end perceived speech quality. Technical report, 1997.

[5] Tom Beigbeder, Rory Coughlan, Corey Lusher, John Plunkett, Emmanuel Agu, and Mark Claypool. The effects of loss and latency on user performance in unreal tournament 2003®. In *Proceedings of ACM SIGCOMM 2004 workshops on NetGames '04 Network and system support for games - SIGCOMM 2004 Workshops*, number May 2002, page 144, New York, New York, USA, 2004. ACM Press. ISBN 158113942X. doi: 10.1145/1016540.1016556. URL http://portal.acm.org/citation.cfm?doid=1016540.1016556.

[6] Anna Bouch, Allan Kuchinsky, and Nina Bhatti. Quality is in the Eye of the Beholder : Meeting Users ' Requirements for Internet Quality of Service. Technical report, 2000.

[7] Tom Brewster. Dealing With The DDoS Dealers, 2012. URL http://www.techweekeurope.co.uk/news/ddos-market-84390. (last visit: Aug. 2012).

[8] Canonical Ltd. limit - get and set user limits. URL http://manpages.ubuntu.com/manpages/lucid/man3/ulimit.3.html. (last visit: Jul. 2012).

[9] Patrikakis Charalampos, Michalis Masikos, and Olga Zouraraki. Distributed Denial of Service Attacks. *The Internet Protocol Journal*, 7(4):13–35, 2004.

[10] Neal Charbonneau, Rukaiya Dahodwala, Raza Kanjee, and Aditya Kashiappa. Experimental and Analytical Analysis of TCP and UDP Protocols. pages 1–10, 2009. URL http://www.ece.umassd.edu/faculty/lxing/Homepage/ECE560-S11-Homepage/Project/SampleReports/ECE560-S09-Group1.pdf.

[11] Roman Chertov, Sonia Fahmy, and Ness B. Shroff. Emulation versus Simulation: A Case Study TCP-Targeted Denial of Service Attacks. *2nd International Conference on Testbeds and Research Infrastructures for the Development of Networks and Communities, 2006. TRIDENTCOM 2006.*, pages 316–325, 2006. doi: 10.1109/TRIDNT.2006.1649164. URL `http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1649164`.

[12] Junghoo Cho and Hector Garcia-Molina. Effective page refresh policies for Web crawlers. *ACM Transactions on Database Systems*, 28(4):390–426, December 2003. ISSN 03625915. doi: 10.1145/958942.958945. URL `http://portal.acm.org/citation.cfm?doid=958942.958945`.

[13] CrySyS - Laboratory of Cryptography and Systems Security. Targeted attacks, 2012. URL `http://www.crysys.hu/targeted-attacks.html`. (last visit: Jul. 2012).

[14] James Cullimore. Anonymous DDoS Mastercard Site In WikiLeaks Revenge Attack, 2010. URL `http://www.itproportal.com/2010/12/08/anonymous-ddos-mastercard-site-wikileaks-revenge-attack/`. (last visit: Jul. 2012).

[15] Dancho Danchev. DDoS for hire services offering to "take down your competitor's web sites" going mainstream, 2012. URL `http://blog.webroot.com/2012/06/06/ddos-for-hire/`. (last visit: Jul. 2012).

[16] die.net. ethtool - Display or change ethernet card settings, . URL `http://linux.die.net/man/8/ethtool`. (last visit: Jul. 2012).

[17] die.net. free - Display amount of free and used memory in the system, . URL `http://linux.die.net/man/1/free`. (last visit: Jul. 2012).

[18] die.net. iostat - Report Central Processing Unit (CPU) statistics and input/output statistics for devices, partitions and network filesystems (NFS), . URL `http://linux.die.net/man/1/iostat`. (last visit: Jul. 2012).

[19] die.net. mpstat - Report processors related statistics, . URL `http://linux.die.net/man/1/mpstat`. (last visit: Jul. 2012).

[20] die.net. netstat - Print network connections, routing tables, interface statistics, masquerade connections, and multicast memberships, . URL `http://linux.die.net/man/8/netstat`. (last visit: Jul. 2012).

[21] die.net. ntpdate - set the date and time via NTP, . URL `http://linux.die.net/man/8/ntpdate`. (last visit: Jul. 2012).

[22] die.net. proc - process information pseudo-file system, . URL `http://linux.die.net/man/5/proc`. (last visit: Jul. 2012).

[23] die.net. select, pselect, FD_CLR, FD_ISSET, FD_SET, FD_ZERO - synchronous I/O multiplexing, . URL `http://linux.die.net/man/2/select`. (last visit: Jul. 2012).

[24] Stephen Dill, Ravi Kumar, Kevin S. Mccurley, Sridhar Rajagopalan, D. Sivakumar, and Andrew Tomkins. Self-similarity in the web. *ACM Transactions on Internet Technology*, 2(3):205–223, August 2002. ISSN 15335399. doi: 10.1145/572326.572328. URL `http://portal.acm.org/citation.cfm?doid=572326.572328`.

[25] Thomas Dubendorfer, Arno Wagner, and Bernhard Plattner. An economic damage model for large-scale Internet attacks. In *13th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises*, pages 223–228. IEEE Comput. Soc, 2004. ISBN 0-7695-2183-5. doi: 10.1109/ENABL.2004.11. URL `http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1376837http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1376837`.

[26] S. Floyd and V. Jacobson. Random early detection gateways for congestion avoidance. *IEEE/ACM Transactions on Networking*, 1(4):397–413, 1993. ISSN 10636692. doi: 10.1109/90.251892. URL `http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=251892`.

[27] Forrester Research Inc. DDoS : A Threat You Can't Afford To Ignore. Technical report, 2009.

[28] Raja Sekhar Reddy Gade, Hari Vellalacheruvu, and Sanjeev Kumar. Performance of Windows XP, Windows Vista and Apple's Leopard Computers under a Denial of Service Attack. In *2010 Fourth International Conference on Digital Society*, pages 188–191. IEEE, February 2010. ISBN 978-1-4244-5805-9. doi: 10.1109/ICDS.2010.39. URL `http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5432801`.

[29] Giovanni Giacobbi. The GNU Netcat, 2006. URL `http://netcat.sourceforge.net/`. (last visit: Jul. 2012).

[30] Sebastien Godard. SYSSTAT. URL `http://sebastien.godard.pagesperso-orange.fr/`. (last visit: Jul. 2012).

[31] Joao Hespanha, Stephan Bohacek, Katia Obraczka, and Junsoo Lee. Hybrid Modeling of TCP Congestion Control. pages 291–304, 2001. URL `http://www.springerlink.com/index/jvvn1jp1wug4vnw5.pdf`.

[32] John D. Howard. An analysis of security incidents on the Internet 1989-1995. 1997. URL `http://oai.dtic.mil/oai/oai?verb=getRecord&metadataPrefix=html&identifier=ADA389085`.

[33] Alefiya Hussain, John Heidemann, and Christos Papadopoulos. A framework for classifying denial of service attacks. In *Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications - SIGCOMM '03*, page 99, New York, New York, USA, 2003. ACM Press. ISBN 1581137354. doi: 10.1145/863955.863968. URL `http://portal.acm.org/citation.cfm?doid=863955.863968`.

[34] Insecure.Com LLC. Ncat. URL `http://nmap.org/ncat/`. (last visit: Jul. 2012).

[35] Intel Corporation. Intel® Core^{TM} i3-540 Processor, 2010. URL `http://ark.intel.com/products/46473`. (last visit: Jul. 2012).

[36] Intel Corporation. Intel® Core^{TM} i7-2600K Processor, 2011. URL `http://ark.intel.com/products/52214/`. (last visit: Jul. 2012).

[37] Iranian National Computer Emergency Response Team. Identification of a New Targeted Cyber-Attack, 2012. URL `http://www.certcc.ir/index.php?name=news&file=article&sid=1894`. (last visit: Jul. 2012).

[38] Kaspersky Lab. Kaspersky Lab and ITU Research Reveals New Advanced Cyber Threat, 2012. URL http://www.kaspersky.com/about/news/virus/2012/Kaspersky_Lab_and_ITU_Research_Reveals_New_Advanced_Cyber_Threat. (last visit: Jul. 2012).

[39] Suraiya Khan and Issa Traore. Queue-based analysis of DoS attacks. In *Information Assurance Workshop, 2005. IAW'05. Proceedings from the Sixth Annual IEEE SMC*, pages 266–273. IEEE, 2005. ISBN 0780392906. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1495962.

[40] Jan Kneschke. lighttpd. URL http://www.lighttpd.net/. (last visit: Jul. 2012).

[41] Alex Lane. Anonymous DDoS forces Virgin Media offline in revenge for Pirate Bay blockade, 2012. URL recombu.com/digital/news/anonymous-ddos-forces-virgin-media-offline_M10406.html. (last visit: Jul. 2012).

[42] Carolyn Duffy Marsan. Verisign expands cloud-based DDoS protection, 2011. URL http://www.networkworld.com/news/2011/050911-verisign-ddos.html. (last visit: Jul. 2012).

[43] Mathematics in Education and Industry. Spearman's rank correlation. Technical report. URL www.mei.org.uk/files/pdf/Spearmanrcc.pdf.

[44] Jelena Mirkovic and Peter Reiher. A taxonomy of DDoS attack and DDoS defense mechanisms. *ACM SIGCOMM Computer Communication*, 2004. URL http://dl.acm.org/citation.cfm?id=997150.997156.

[45] Jelena Mirkovic, Alefiya Hussain, Brett Wilson, Roshan Thomas, Stephen Schwab, Sonia Fahmy, Roman Chertov, and Peter Reiher. DDoS Benchmarks and Experimenter's Workbench for the DETER Testbed. In *In Proceedings of 3rd International IEEE/CreateNet Conference on Testbeds and Research Infrastructures for the Development of Networks and Communities (TridentCom)*, 2007. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4444680.

[46] Jelena Mirkovic, Peter Reiher, Brett Wilson, Sonia Fahmy, Roshan Thomas, Wei-min Yao, and Stephen Schwab. Towards User-Centric Metrics for Denial-Of-Service Measurement. In *Proceedings of the 2007 workshop on Experimental computer science*, number June, 2007. ISBN 9781595937513. doi: 10.1145/1281700.1281708. URL http://portal.acm.org/citation.cfm?id=1281708.

[47] Jelena Mirkovic, Sonia Fahmy, Peter Reiher, and Roshan K. Thomas. How to Test DoS Defenses. *2009 Cybersecurity Applications & Technology Conference for Homeland Security*, pages 103–117, March 2009. doi: 10.1109/CATCH.2009.23. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4804432.

[48] Jelena Mirkovic, Alefiya Hussain, Sonia Fahmy, Peter Reiher, and R.K. Thomas. Accurately Measuring Denial of Service in Simulation and Testbed Experiments. *IEEE Transactions on Dependable and Secure Computing*, 6(2):81–95, April 2009. ISSN 1545-5971. doi: 10.1109/TDSC.2008.73. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4689468.

[49] Nortel Networks. QoS Performance requirements for UMTS. Technical Report May, 1999.

[50] Robert Olsson. pktgen the linux packet generator. Technical report, 2004.

[51] Oracle Corporation. MySQL: The world's most popular open source database. URL http://www.mysql.com/. (last visit: Jul. 2012).

[52] Packet Storm. Tor's Hammer - Slow POST Denial Of Service Testing Tool, 2011. URL http://packetstormsecurity.org/files/98831/Tors-Hammer-Slow-POST-Denial-Of-Service-Testing-Tool.html. (last visit: Mar. 2012).

[53] Jitendra Padhye, Victor Firoiu, Don Towsley, and Jim Kurose. Modeling TCP throughput. *ACM SIGCOMM Computer Communication Review*, 28(4):303–314, October 1998. ISSN 01464833. doi: 10.1145/285243.285291. URL http://portal.acm.org/citation.cfm?doid=285243.285291.

[54] PLANET Technology Corporation. GSD-805. URL http://www.planet.com.tw/en/product/product_ov.php?id=19375. (last visit: Jul. 2012).

[55] Robey Pointer. paramiko: SSH2 protocol for python. URL http://www.lag.net/paramiko/. (last visit: Jul. 2012).

[56] RSnake and John Kinsella. Slowloris HTTP DoS. URL http://ha.ckers.org/slowloris/. (last visit: Jul. 2012).

[57] Syed Balal Rumy. DDOS Attack Taxanomy. URL http://ccnpsecurity.blogspot.com/2012/01/ddos-attack-taxanomy.html. (last visit: Mar. 2012).

[58] Monika Sachdeva, Krishan Kumar, Gurvinder Singh, and Kuldip Singh. Performance Analysis of Web Service under DDoS Attacks. In *2009 IEEE International Advance Computing Conference*, number March, pages 1002–1007. IEEE, March 2009. ISBN 978-1-4244-2927-1. doi: 10.1109/IADCC.2009.4809152. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4809152http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4809152.

[59] Monika Sachdeva, Gurvinder Singh, and Krishan Kumar. An emulation based impact analysis of DDoS attacks on web services during flash events. *2011 2nd International Conference on Computer and Communication Technology (ICCCT-2011)*, pages 479–484, September 2011. doi: 10.1109/ICCCT.2011.6075134. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6075134.

[60] David E. Sanger. Obama Order Sped Up Wave of Cyberattacks Against Iran, 2012. URL http://www.nytimes.com/2012/06/01/world/middleeast/obama-ordered-wave-of-cyberattacks-against-iran.html?_r=2&hp&pagewanted=all. (last visit: Jul. 2012).

[61] Nathan Sheldon, Eric Girard, Seth Borg, Mark Claypool, and Emmanuel Agu. The effect of latency on user performance in Warcraft III. In *Proceedings of the 2nd workshop on Network and system support for games - NETGAMES '03*, pages 3–14, New York, New York, USA, 2003. ACM Press. ISBN 1581137346. doi: 10.1145/963900.963901. URL http://portal.acm.org/citation.cfm?doid=963900.963901.

[62] Joel E. Sommers. Harpoon: A Flow-level Traffic Generator, 2005. URL `http://cs.colgate.edu/~jsommers/harpoon/`. (last visit: Jul. 2012).

[63] Daniel Stenberg. curl - transfer a URL. URL `http://curl.haxx.se/docs/manpage.html`. (last visit: Jul. 2012).

[64] Tenable Network Security. Nessus. URL `http://www.nessus.org/products/nessus`. (last visit: Jul. 2012).

[65] The Apache Software Foundation. Apache HTTP Server Project. URL `http://httpd.apache.org/`. (last visit: Jul. 2012).

[66] The Eclipse Foundation. Jetty. URL `http://www.eclipse.org/jetty/`. (last visit: Aug. 2012).

[67] The MathWorks Inc. MATLAB - The Language of Technical Computing. URL `http://www.mathworks.com/products/matlab/index.html`. (last visit: Jul. 2012).

[68] The National Laboratory for Advanced Network Research. Iperf. URL `http://iperf.sourceforge.net/`. (last visit: Jul. 2012).

[69] The PHP Group. PHP. URL `http://www.php.net/`. (last visit: Jul. 2012).

[70] The Tcpdump Group. tcpdump. URL `http://www.tcpdump.org/`. (last visit: Jul. 2012).

[71] The Tor Project Inc. Tor Project: Anonymity Online. URL `https://www.torproject.org/`. (last visit: Aug. 2012).

[72] ThoughtWorks Inc. Selenium HQ: Web application testing system. URL `http://seleniumhq.org/`. (last visit: Aug. 2012).

[73] Twisted Matrix Labs. Twisted. URL `http://twistedmatrix.com/trac/`. (last visit: Jul. 2012).

[74] Rangarajan Vasudevan, Z. Morley Mao, Oliver Spatscheck, and Jacobus Van der Merwe. MIDAS: An Impact Scale for DDoS attacks. *2007 15th IEEE Workshop on Local & Metropolitan Area Networks*, pages 200–205, June 2007. doi: 10.1109/LANMAN.2007.4295999. URL `http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4295999`.

[75] VMware Inc. VMware vSphere Hypervisor. URL `http://www.vmware.com/products/vsphere-hypervisor/overview.html`. (last visit: Aug. 2012).

[76] Wikipedia - The Free Encyclopedia. Hacktivism, . URL `http://en.wikipedia.org/wiki/Hacktivism`. (last visit: Jul. 2012).

[77] Wikipedia - The Free Encyclopedia. Coefficient of determination, . URL `http://en.wikipedia.org/wiki/Coefficient_of_determination`. (last visit: Jul. 2012).

[78] Wikipedia - The Free Encyclopedia. Flash mob, . URL `http://en.wikipedia.org/wiki/Flash_mob`. (last visit: Jul. 2012).

[79] Wikipedia - The Free Encyclopedia. mpstat, . URL `http://en.wikipedia.org/wiki/Mpstat`. (last visit: Jul. 2012).

[80] Wikipedia - The Free Encyclopedia. Non-linear least squares, . URL http://en.wikipedia.org/wiki/Non-linear_least_squares. (last visit: Jul. 2012).

[81] Wikipedia - The Free Encyclopedia. Slowloris, . URL http://en.wikipedia.org/wiki/Slowloris. (last visit: Jul. 2012).

[82] Wikipedia - The Free Encyclopedia. p-value, . URL http://en.wikipedia.org/wiki/P_value. (last visit: Aug. 2012).

[83] Wireshark Foundation. tshark, . URL http://www.wireshark.org/docs/man-pages/tshark.html. (last visit: Jul. 2012).

[84] Wireshark Foundation. Wireshark, . URL http://www.wireshark.org/. (last visit: Jul. 2012).

[85] Wolfram Research Inc. Spearman Rank Correlation Coefficient. URL http://mathworld.wolfram.com/SpearmanRankCorrelationCoefficient.html. (last visit: Jul. 2012).

[86] Paul Wood, Gerry Egan, Kevin Haley, Tuan-Khanh Tran, Orla Cox, Hon Lau, Candid Wueest, David McKinney, Tony Millington, Benjamin Nahorney, Joanne Mulcahy, John Harrison, Thomas Parsons, Andrew Watson, Mathew Nisbet, Nicholas Johnston, Bhaskar Krishnappa, Irfan Asrar, Sean Hittel, Eric Chien, Eric Park, Mathew Maniyara, Olivier Thonnard, Pierre-Antoine Vervier, Martin Lee, Daren Lewis, and Scott Wallace. Internet Security Threat Report. Technical Report April, Symantec Corporation, 2012. URL http://www.symantec.com/threatreport/.

[87] Lenny Zeltser. An Ad for DDoS Services - Network, Phone, Competition, 2008. URL https://isc.sans.edu/diary.html?storyid=5380. (last visit: Jul. 2012).

Master Thesis Task Assignment for Cyrill Bannwart (D-ITET)

Predicting the Impact of Denial of Service Attacks

Main advisor:	Dr. Vincent Lenders (armasuisse)
Advisor ETH:	David Gugelmann (ETH Zürich)
Supervisor:	Prof. Dr. B. Plattner (ETH Zürich)
Start Date:	February 13th, 2012
End Date:	August 12th, 2012

1 Introduction

Denial of service attacks are still representing a major threat in the Internet. The potential damage that denial of service attacks may exert on critical infrastructures is however often not a priori known because current impact analysis techniques require the target systems to be flooded in order to observe their behaviour under massive load. Massive flooding of critical systems for testing purposes is however problematic in practice because one cannot tolerate the resulting outage of the system.

2 Thesis goals

The goal of this thesis is to develop a measurement-based prediction model to assess the impact of denial of service attacks on Internet servers. The model should trade off accuracy for probe traffic volume to make its application suitable in operational systems. The model should further take the view of an outsider, i.e., the prediction should be achieved by regular clients without any privileged administrator access to the servers or the networks under consideration.

The required set of tools to apply the model in practice should be evaluated and/or implemented by the student. The goal at the end is to have a simple and generic tool suite that can be used by others to test any server (in the same spirit as the nessus tool suite to assess software vulnerabilities of running servers).

The model and tools should finally be evaluated by predicting the vulnerability of real servers in lab conditions (~10 server instances).

3 Tasks

The tasks of this thesis are the following:

- Review the literature on denial of service (DoS) and distributed denial of service (DDoS) attacks and identify related work that aims at predicting the impact of DoS and DDoS attacks on Internet servers.
- Familiarize yourself with existing network and end-to-end measurement tools.
- Develop a cross-layer model to externally predict the collateral damage of denial of service attacks based on probe traffic sent from client nodes.
- Evaluate and implement appropriate tools to support your model.

- Test and evaluate your model/tools in a lab environment with your own deployment of target servers.

4 Deliverables

- At the end of the second week, a detailed time schedule of the thesis must be given to and discussed with the main advisors.
- At the end of the first month, the student has to propose a methodology and preliminary model answering the second and third bullet, respectively.
- At the end of the second month, a short discussion of 15 minutes with the supervisor and the advisors will take place. The student has to talk about the major aspects of the ongoing work using slides.
- At the end of month four, another meeting with the supervisor will take place. At this point, the student should already have a preliminary version of the written report or at least a table of content to hand in to the supervisor. This preliminary version should be brought along to the short discussion.
- At the end of the thesis, a presentation of 15 minutes must be given at armasuisse and at ETH (in English) during a CSG group meeting. The presentations should give an overview as well as the most important details of the work. If possible, a demonstrator should be presented (offline after the talk).
- The final report should be written in English but may be written in German. It should contain a summary written in both English and German, the assignment and the time schedule. Its structure should include an introduction, an analysis of related work, and a complete documentation of all used hardware/software tools. Additionally, the report must contain the signed confirmation regarding plagiarism as last page.
- The final report and developed tools/software must be handed in electronically at the end of the thesis. Additionally, four written copies of the final report should be delivered to the main advisor along with a CD that includes developments undergone during the thesis.