



**NTNU – Trondheim**  
Norwegian University of  
Science and Technology

# Visualization of large scale Netflow data

**Nicolai Eeg-Larsen**

Submission date: May 2016  
Responsible professor: Yuming Jiang, ITEM  
Supervisor: Arne Øslebø, UNINETT

Norwegian University of Science and Technology  
Department of Telematics



**Title:** Visualization of large scale Netflow data  
**Student:** Nicolai Eeg-Larsen

**Problem description:**

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

This is the second paragraph. Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

**Responsible professor:** Yuming Jiang, ITEM  
**Supervisor:** Arne Øslebø, UNINETT



## Sammendrag

Sikkerheten til nesten all offentlig nøkkel-kryptografi er basert på et vanskelig beregnbarhetsproblem. Mest velkjent er problemene med å faktorisere heltall i sine primtallsfaktorer, og å beregne diskrete logaritmer i endelige sykliske grupper. I de to siste tiårene, har det imidlertid dukket opp en rekke andre offentlig nøkkel-systemer, som baserer sin sikkerhet på helt andre type problemer. Et lovende forslag, er å basere sikkerheten på vanskeligheten av å løse store likningssett av flervariabe polynomlikninger. En stor utfordring ved å designe slike offentlig nøkkel-systemer, er å integrere en effektiv “falluke” (trapdoor) inn i likningssettet. En ny tilnærming til dette problemet ble nylig foreslått av Gligoroski m.f., hvor de benytter konseptet om kvasigruppe-strengtransformasjoner (quasigroup string transformations). I denne masteroppgaven beskriver vi en metodikk for å identifisere sterke og svake nøkler i det nylig foreslalte multivariable offentlig nøkkel-signatursystemet MQQ-SIG, som er basert på denne idéen.

Vi har gjennomført et stort antall eksperimenter, basert på Gröbner basis angrep, for å klassifisere de ulike parametrene som bestemmer nøklen i MQQ-SIG. Våre funn viser at det er store forskjeller i viktigheten av disse parametrene. Metodikken består i en klassifisering av de forskjellige parametrene i systemet, i tillegg til en innføring av konkrete kriterier for hvilke nøkler som bør velges. Videre, har vi identifisert et unødvendig krav i den originale spesifikasjonen, som krevde at kvasigruppene måtte oppfylle et bestemt kriterie. Ved å fjerne denne betingelsen, kan nøkkelerings-algoritmen potensielt øke ytelsen med en stor faktor. Basert på alt dette, foreslår vi en ny og forbedret nøkkel-genereringsalgoritme for MQQ-SIG, som vil generere sterkere nøkler og være mer effektiv enn den originale nøkkel-genereringsalgoritmen.



## Preface

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.



## Acknowledgements

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.



# Contents

|  |             |
|--|-------------|
| <b>List of Figures</b>                         | <b>ix</b>   |
| <b>Listings</b>                                | <b>xi</b>   |
| <b>List of Tables</b>                          | <b>xiii</b> |
| <b>List of Acronyms</b>                        | <b>xv</b>   |
| <b>1 Introduction</b>                          | <b>1</b>    |
| 1.1 Motivation . . . . .                       | 1           |
| 1.2 Scope and Objectives . . . . .             | 1           |
| 1.2.1 Scope . . . . .                          | 1           |
| 1.2.2 Objectives . . . . .                     | 2           |
| <b>2 Background</b>                            | <b>3</b>    |
| 2.1 NetFlow . . . . .                          | 3           |
| 2.1.1 How does it work? . . . . .              | 3           |
| 2.1.2 Main components . . . . .                | 5           |
| 2.1.3 nfdump . . . . .                         | 5           |
| 2.2 Data visualization . . . . .               | 7           |
| 2.2.1 Characteristics . . . . .                | 8           |
| 2.2.2 Visual perception . . . . .              | 8           |
| 2.2.3 Data presentation architecture . . . . . | 9           |
| 2.3 D3.js . . . . .                            | 10          |
| 2.3.1 How does it work? . . . . .              | 10          |
| <b>3 Research</b>                              | <b>15</b>   |
| 3.1 Related work . . . . .                     | 15          |
| 3.2 Initial research . . . . .                 | 16          |
| 3.3 Traits of a DDoS attack . . . . .          | 17          |
| 3.3.1 Raw NetFlow format . . . . .             | 18          |
| <b>4 D3.js and NetFlow</b>                     | <b>21</b>   |

|                   |  |           |
|-------------------|--|-----------|
| 4.1               | Using D3.js . . . . .                                | 21        |
| 4.1.1             | Scope . . . . .                                      | 21        |
| 4.2               | Number of flows to a certain host and port . . . . . | 23        |
| 4.2.1             | Scope in D3.js . . . . .                             | 23        |
| 4.2.2             | Pros and cons . . . . .                              | 26        |
| <b>5</b>          | <b>Discussion</b>                                    | <b>29</b> |
| <b>6</b>          | <b>Challenges</b>                                    | <b>31</b> |
| 6.1               | Large data sets . . . . .                            | 31        |
| 6.1.1             | IP-spectrum . . . . .                                | 31        |
| 6.1.2             | Increasing number of flows . . . . .                 | 31        |
| 6.2               | Live updates . . . . .                               | 32        |
| <b>7</b>          | <b>Concluding Remarks and Future Work</b>            | <b>33</b> |
| 7.1               | Recommendations for Future Work . . . . .            | 33        |
| 7.1.1             | Further testing . . . . .                            | 33        |
| 7.1.2             | Customized searches . . . . .                        | 33        |
| <b>References</b> |  | <b>35</b> |
| <b>Appendices</b> |  |           |
| <b>A</b>          | <b>Appendix A</b>                                    | <b>39</b> |
| <b>B</b>          | <b>Appendix B</b>                                    | <b>49</b> |

# List of Figures

|     |   |    |
|-----|---|----|
| 2.1 | Creating a flow in the NetFlow cache siter . . . . .  | 4  |
| 2.2 | Figure of a simple NetFlow architecture . . . . .   | 5  |
| 2.3 | Example of dataset of random numbers where no pre-attentive processing is done . . . . .  | 6  |
| 2.4 | Example of dataset of random numbers where no pre-attentive processing is done . . . . .  | 6  |
| 2.5 | Example of dataset of random numbers where no pre-attentive processing is done . . . . .  | 7  |
| 2.6 | Example of dataset of random numbers where no pre-attentive processing is done . . . . .  | 9  |
| 2.7 | Example of a dataset of random numbers where pre-attentive processing has been used to distinguish the occurrences of the number five . . . . . | 9  |
| 3.1 | Ten files in the provided files with the most flows [cis16] . . . . .   | 17 |
| 3.2 | Ten files in the provided files with the most flows [cis16] . . . . .   | 18 |
| 3.3 | Ten files in the provided files with the most flows . . . . .   | 19 |
| 4.1 | Ten files in the provided files with the most flows . . . . .   | 22 |
| 4.2 | The smallest files from the provided files . . . . .  | 22 |
| 4.3 | Top ten used destination addresses within the timeframe 1300-1400, 18th of January . . . . .  | 22 |
| 4.4 | Top ten used destination addresses within the timeframe 1300-1400, 18th of January . . . . .  | 23 |
| 4.5 | Top ten used destination addresses within the timeframe 1300-1400, 18th of January . . . . .  | 24 |
| 4.6 | Top ten used destination addresses within the timeframe 1300-1400, 18th of January . . . . .  | 25 |
| 4.7 | Top ten used destination addresses within the timeframe 1300-1400, 18th of January . . . . .  | 25 |



# Listings

|                              |   |
|------------------------------|---|
| 2.1 nfdump example . . . . . | 7 |
|------------------------------|---|



# List of Tables

|     |  |    |
|-----|--|----|
| 4.1 | Pros and cons of a visual solution . . . . .       | 26 |
| 4.2 | Pros and cons of a command line solution . . . . . | 27 |



# List of Acronyms

**API** Application Programming Interface.

**AS** Autonomnous Systems.

**BGP** Border Gateway Protocol.

**CSS** Cascading Style Sheets.

**CSV** Comma Separated Value.

**DDoS** Distributed Denial of Service.

**DoS** Denial of Service.

**DPA** Data Presentation Architecture.

**GB** GigaByte.

**HTML** HyperText Markup Language.

**IP** Internet Protocol.

**IPFIX** IP Flow Information eXport.

**IPv4** Internet Protocol version 4.

**IPv6** Internet Protocol version 6.

**LVM** Logical Volume Manager.

**NTNU** Norwegian University of Science and Technology.

**SVG** Scalable Vector Graphics.

**TPC** Transmission Control Protocol.



# Chapter 1

# Introduction

## 1.1 Motivation

Network security, monitoring and Big Data are becoming major pillars in the increase in data usage. Data traffic is growing at an alarming rate[[CO02], and is not showing any signs of slowing down. A majority of this traffic is harmless and not in the interest for network monitors. As the amount of traffic increases, separating malicious from other activity is becoming more difficult. Today network monitoring can be achieved through using Cisco's NetFlow standard along with tools as nfdump. The visual presentation in such tools are limited, and is not very interactive or intuitive.

Theory on how data is best presented visually dates back decades. The term 'visual presentation' is used to refer to the actual presentation of information through a visible medium such as text or images [Jam07]. It ranges from simple text to body-language, and a good visual communication design is evaluated based on the comprehension by the audience.

Using visualization to represent NetFlow data has not been common, and was first published as a solution after the 2004 ACM Workshop on Visualization and data mining for computer security [LYBL04]. In this paper NVisionIP, a java based solution is proposed. Capable of providing the system state, and possibly with further work find patterns or attacks on the network.

## 1.2 Scope and Objectives

### 1.2.1 Scope

This thesis will cover the basics in how NetFlow data is generated and collected, and also the tools used to manipulate the raw data. As well as showing theory behind general visualization and how to present data such as NetFlow. D3.js is used to create examples using real NetFlow data provided by UNINETT.

## 2 1. INTRODUCTION

Connecting the basics mentioned earlier in this section to a simple solution using real anonymous data and allowing volunteers, with experience using nfdump and existing tools, to provide feedback(see 1.2.2 ).

### **1.2.2 Objectives**

#### **O.1: Processing anonymous data**

UNINETT provided one month of data, which is hundreds of GigaByte (GB). A large part of the process is analysing and perform queries using nfdump to be able to extract the exact information needed, as well as sorting it and creating files readable by the D3.js framework.

#### **O.2: Developing working solution**

Objective two concerns the development of a simple visual solution using D3.js. One of the main objectives is to use as much of the theory covered in 2.2 effectively.

#### **O.3 Evaluate solution**

In objective three the main objective is discussing if the solution created in O.2 display enough potential to be able to use visualization as a tool in network monitoring.

# Chapter 2

## Background

### 2.1 NetFlow

Cisco IOS NetFlow creates an environment that have the tools to understand who, what, when, where and how network traffic is flowing. This makes it easier for administrators to utilize the network as optimal as possible. One can determine the source and destination of traffic and use this information to reveal for example DDoS-attacks or spam mail. [PMAØ04]

bruke  
den siste  
refer-  
ansen

#### 2.1.1 How does it work?

Every packet that is forwarded within a router/switch is examined for a set of Internet Protocol (IP) packet attributes. With these attributes one can determine if the packet is unique or similar to other packets.

The attributes used by NetFlow are:

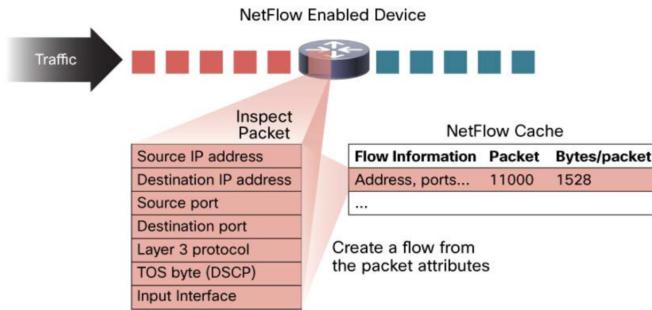
- IP source address
- IP destination address
- Source port
- Destination port
- Layer 3 protocol type
- Class of service
- Router/Switch interface

To group packets into a flow, one compares source/destination IP address, source/destination ports, protocol interface and class of service. Then the packets

## 4 2. BACKGROUND

and bytes are tallied. This method is scalable because a large amount of network information is condensed into a database of NetFlow information called the NetFlow cache.

When the NetFlow cache is created one can use this to understand the network behaviour. The different attributes generate different knowledge about a certain network, and combined they can paint a detailed picture of how the network is working. For example the ports show what application is utilizing the traffic, while the tallied packets and bytes show the amount of traffic. [SST<sup>+</sup>16] [EKMV04]



**Figure 2.1:** Creating a flow in the NetFlow cache siter

- Source address allows the understanding of who is originating the traffic
- Destination address tells who is receiving the traffic
- Ports characterize the application utilizing the traffic
- Class of service examines the priority of the traffic
- The device interface tells how traffic is being utilized by the network device
- Tallied packets and bytes show the amount of traffic

Additional information added to a flow includes:

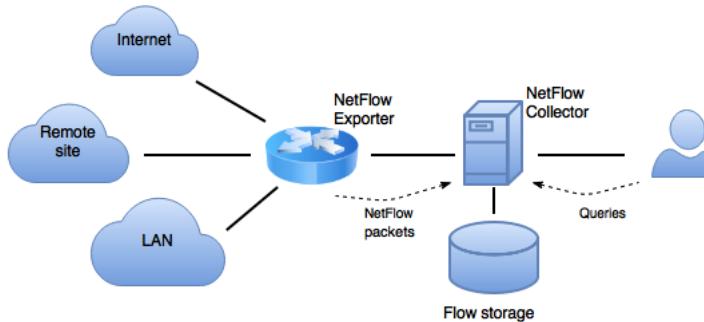
- Flow timestamps to understand the life of a flow; timestamps are useful for calculating packets and bytes per second
- Next hop IP addresses including Border Gateway Protocol (BGP) routing Autonomous Systems (AS)
- Subnet mask for the source and destination addresses to calculate prefixes

- flags to examine Transmission Control Protocol (TCP) handshakes

### 2.1.2 Main components

A typical set-up using NetFlow consists of three main components:

- **Flow Exporter:** aggregates packets into flows and exports flow records towards one or more flow collectors.
- **Flow collector:** is responsible for reception, storage and pre-processing of flow data received from a flow exporter.
- **Analysis application:** an application that analyse the received flow data in different contexts, such as intrusion or traffic profiling.



**Figure 2.2:** Figure of a simple NetFlow architecture

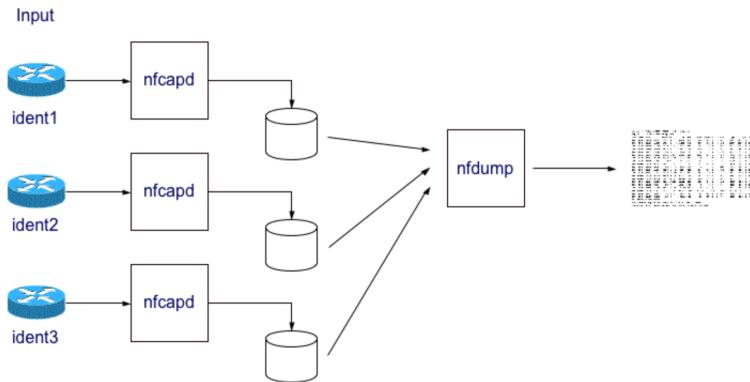
### 2.1.3 nfdump

skrive om ipfix, v5 og v9, muligens i egen seksjon) nfdump collect and process NetFlow data on the command line. It stores NetFlow data in time sliced files. The files are binary and this provides the possibility of either returning the output from nfdump in the same binary form, or as readable text. nfdump has four output formats, raw, line, long and extended. The challenge of representing Internet Protocol version 6 (IPv6) addresses is handled by shrinking them in the normal output. In figure 2.3 the collection process is depicted, and in figure 2.4 the processing of collected NetFlow data is shown.[nfd16]

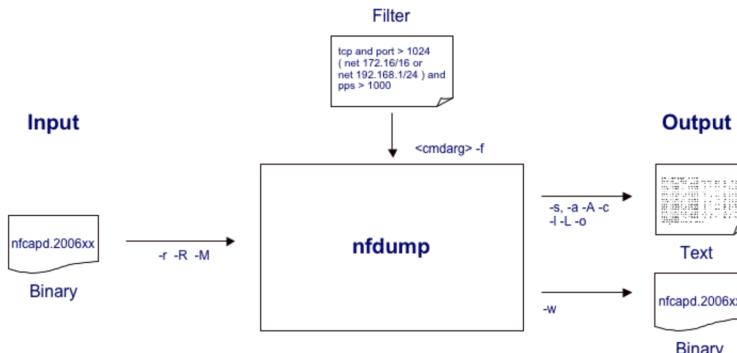
Lime inn  
hvordan  
det ser  
ut i kom-  
mandolin-  
jen

sitere  
listen

## 6 2. BACKGROUND



**Figure 2.3:** Example of dataset of random numbers where no pre-attentive processing is done



**Figure 2.4:** Example of dataset of random numbers where no pre-attentive processing is done

### v5,v9 and IPfix

- **v5:** NetFlow v5 is definitely the most popular version of Cisco Netflow. It is fixed, meaning it always stays the same and makes for a simpler deciphering.
- **v9:** v9 is opposite of its predecessor dynamic. The collector will need to know the format of incoming NetFlow v9 flows, which means v9 templates periodically needs to be sent to the collector to inform of the format which the flows are being exported are. It was made to support technologies as Multi-cast, IPSec and Multi Protocol Label Switching (MPLS). This thanks to the templates. IPv6 support was added as well. [Cla04]

- **IPFIX:** Based on the design of NetFlow v9, IP Flow Information eXport (IPFIX) added support for variable length strings. Making it possible for Application Visibility and Control(AVC) exports in the future.[TB11]

forklare?

### Example of use

An example of a nfdump command used in this project is for example how to extract the number of flows each day, and find the 10 most used destination IP-addresses:

```
1 nfdump -R /data/netflow/oslo_gw/2012/01/01/nfcapd.201201010000:nfcapd
    .201201012355 -n 10 -s dstip -o csv > example.csv
```

**Listing 2.1:** nfdump example

Such a request iterates over a number of files due to the -R command. In this case it is all captures between 00:00 until 23:55 on the first of January 2012. It is limited to the 10 most popular destination IP's with the -n and -s. All of this is stored in a .Comma Separated Value (CSV)file which is optimal for use with the D3.js framework.

This results in the output shown in figure 2.5.

| Top 10 Dst IP Addr ordered by flows: |           |       |                |              |              |               |          |        |      |     |
|--------------------------------------|-----------|-------|----------------|--------------|--------------|---------------|----------|--------|------|-----|
| Date first seen                      | Duration  | Proto | Dst IP         | Addr         | Flows(%)     | Packets(%)    | Bytes(%) | pps    | bps  | bpp |
| 2011-12-31 23:58:52.073              | 86420.475 | any   | 191.229.233.80 | 304171( 2.3) | 374411( 0.4) | 369.4 M( 0.9) | 4        | 34194  | 986  |     |
| 2011-12-31 23:58:56.796              | 86421.658 | any   | 162.185.32.85  | 204693( 1.5) | 210870( 0.2) | 29.1 M( 0.1)  | 2        | 2693   | 137  |     |
| 2011-12-31 23:58:58.872              | 86416.387 | any   | 161.220.8.250  | 171799( 1.3) | 175078( 0.2) | 7.8 M( 0.0)   | 2        | 721    | 44   |     |
| 2011-12-31 23:58:58.748              | 86421.701 | any   | 162.185.32.105 | 171177( 1.3) | 367501( 0.4) | 21.7 M( 0.1)  | 4        | 2009   | 59   |     |
| 2011-12-31 23:58:58.170              | 86422.648 | any   | 190.49.180.97  | 137249( 1.0) | 380318( 0.4) | 296.5 M( 0.7) | 4        | 27447  | 779  |     |
| 2011-12-31 23:58:52.698              | 86419.411 | any   | 161.223.1.108  | 116698( 0.9) | 158786( 0.2) | 25.4 M( 0.1)  | 1        | 2355   | 160  |     |
| 2011-12-31 23:58:50.762              | 86421.892 | any   | 159.152.49.239 | 114587( 0.9) | 696163( 0.8) | 953.8 M( 2.3) | 8        | 88288  | 1379 |     |
| 2011-12-31 23:58:52.421              | 86419.823 | any   | 190.49.138.164 | 98869( 0.7)  | 110728( 0.1) | 6.8 M( 0.0)   | 1        | 632    | 61   |     |
| 2011-12-31 23:58:52.881              | 86419.483 | any   | 190.49.138.168 | 84728( 0.6)  | 109403( 0.1) | 161.8 M( 0.4) | 1        | 14977  | 1478 |     |
| 2012-01-01 15:19:13.644              | 29561.745 | any   | 71.238.74.19   | 84632( 0.6)  | 592943( 0.7) | 793.9 M( 1.9) | 20       | 214838 | 1338 |     |

|   |
|---|
| Summary: total flows: 13373369, total bytes: 41.8 G, total packets: 86.5 M, avg pps: 1000, avg bpp: 482 |
| Time window: <unknown>  |
| Total flows processed: 13373369, Blocks skipped: 0, Bytes read: 695426912                               |
| Sys: 5.373s flows/second: 2488661.8 Wall: 5.407s flows/second: 2473155.2                                |

**Figure 2.5:** Example of dataset of random numbers where no pre-attentive processing is done

## 2.2 Data visualization

Data visualization refers to the techniques used to communicate data or information by encoding it as visual objects. Meaning that information is represented through any visual element such as graphs and plots, but may also take any other visual form. Visualization helps users analyse and interact with data in a whole new way. It makes complex data more accessible, understandable and usable.[Fri08]

In recent years the rate of which data is generated has increased rapidly, and the need for information to be available and comprehensible is growing. All these

new sources of data has created what we refer to as "Big Data" [MBD<sup>+</sup>12]. Without visual presentation such data is too big to understand. This is one of the major reasons visualization is emerging as a big market.

Combining several parameters through visualization could reveal something automated systems might ignore or don't pick up on.

The greatest value of a picture is when it forces us to notice what we never expected to see.

by John Tukey.

### 2.2.1 Characteristics

In his book from 1983, *The Visual Display of Quantitative Information*[TGM83], Edward Tufte defines characteristics any effective graphical representation should contain as:

- show the data
- induce the viewer to think about the substance rather than about methodology, graphic design, the technology of graphic production or something else
- avoid distorting what the data has to say
- present many numbers in a small space
- make large data sets coherent
- encourage the eye to compare different pieces of data
- reveal the data at several levels of detail, from a broad overview to the fine structure
- serve a reasonably clear purpose: description, exploration, tabulation or decoration
- be closely integrated with the statistical and verbal descriptions of a data set.

### 2.2.2 Visual perception

In this paper the correlation between effective visual communication and how it is perceived upon human inspection is important. A humans ability to distinguish between differences in length, shape and color is referred to as "pre-attentive attributes".

A good example of this is imagining finding the number of a certain character in a series of characters. This requires significant time and effort, but if the character were to stand out by being a different size, color or orientation this could be done quickly through pre-attentive processing. Good data visualization takes all of this into consideration and uses pre-attentive processing. In this simple example it is easy to see how pre attentive processing is used to distinguish how many occurrences of the number 5 is in a larger set of random numbers.

```
987349790275647902894728624092406037070570279072
803208029007302501270237008374082078720272007083
247802602703793775709707377970667462097094702780
927979709723097230979592750927279798734972608027
```

**Figure 2.6:** Example of dataset of random numbers where no pre-attentive processing is done

```
987349790275647902894728624092406037070570279072
803208029007302501270237008374082078720272007083
247802602703793775709707377970667462097094702780
927979709723097230979592750927279798734972608027
```

**Figure 2.7:** Example of a dataset of random numbers where pre-attentive processing has been used to distinguish the occurrences of the number five

### 2.2.3 Data presentation architecture

Data Presentation Architecture (DPA) has its purpose to identify, locate, manipulate, format and present data in such a way as to optimally communicate meaning and proffer knowledge[wik16]. This has become an important tool in Business Intelligence, the art of transforming raw data into something useful.

#### Objectives

DPA has two main objectives, which is the following:

## 10 2. BACKGROUND

- To use data to provide knowledge in the most efficient manner possible (minimize noise, complexity, and unnecessary data or detail given each audience's needs and roles)
- To use data to provide knowledge in the most effective manner possible (provide relevant, timely and complete data to each audience member in a clear and understandable manner that conveys important meaning, is actionable and can affect understanding, behaviour and decisions)

### Scope

The actual work of DPA consist of:

- Creating effective delivery mechanisms
- Define relevant knowledge needed by each viewer
- Determine how often the data should be updated
- Determine how often and when the user needs to see the data
- Finding the right data
- Utilizing the best visualizations and presentation formats

## 2.3 D3.js

In this paper D3.js [Bos16] is chosen as the framework to create examples of effective data visualizations due to its dynamical and interactive properties. D3 stands for Data-Driven Documents, and is a Javascript library. D3.js allows users to bind arbitrary data to a Document Object Model. It uses widely implemented Scalable Vector Graphics (SVG), Cascading Style Sheets (CSS) and HyperText Markup Language (HTML)5 standards. D3 is unique in the way it creates SVG objects from large datasets using simple D3.js functions to generate rich text/graphic charts and diagrams.

### 2.3.1 How does it work?

The W3C DOM API is often tiring to use. An example bit of code from [link/kilde] shows how one changes the text color of paragraph elements:

```
,  
2 var paragraphs = document.getElementsByTagName( "p" );  
4 for (var i = 0; i < paragraphs.length; i++) {
```

```

6   var paragraph = paragraphs.item(i);
7   paragraph.style.setProperty("color", "white", null);
8 }
```

**Listing 2.2:** HTML example

In D3.js this could be solved through one line of code:

```
d3.selectAll("p").style("color", "white");
```

**Listing 2.3:** D3.js example

D3.js also possesses dynamic properties which gives the user a powerful tool to create advanced graphics with a small amount of code.

This next snippet of code shows how the D3.js framework simply appends to an existing html object. [sim16]

```

10 <!DOCTYPE html>
11 <meta charset="utf-8">
12 <style> /* set the CSS */
13
14 body { font: 12px Arial; }
15
16 path {
17   stroke: steelblue;
18   stroke-width: 2;
19   fill: none;
20 }
21
22 .axis path,
23 .axis line {
24   fill: none;
25   stroke: grey;
26   stroke-width: 1;
27   shape-rendering: crispEdges;
28 }
29
30 </style>
31 <body>
32
33 <!-- load the d3.js library -->
34 <script src="http://d3js.org/d3.v3.min.js"></script>
35
36 <script>
37
38 // Set the dimensions of the canvas / graph
39 var margin = {top: 30, right: 20, bottom: 30, left: 50},
40     width = 600 - margin.left - margin.right,
41     height = 270 - margin.top - margin.bottom;
```

## 12 2. BACKGROUND

```
// Parse the date / time
44 var parseDate = d3.time.format("%d-%b-%y").parse;

46 // Set the ranges
47 var x = d3.time.scale().range([0, width]);
48 var y = d3.scale.linear().range([height, 0]);

50 // Define the axes
51 var xAxis = d3.svg.axis().scale(x)
52   .orient("bottom").ticks(5);

54 var yAxis = d3.svg.axis().scale(y)
55   .orient("left").ticks(5);

56 // Define the line
57 var valueline = d3.svg.line()
58   .x(function(d) { return x(d.date); })
59   .y(function(d) { return y(d.close); });

62 // Adds the svg canvas
63 var svg = d3.select("body")
64   .append("svg")
65     .attr("width", width + margin.left + margin.right)
66     .attr("height", height + margin.top + margin.bottom)
67   .append("g")
68     .attr("transform",
69       "translate(" + margin.left + ", " + margin.top + ")");

70 // Get the data
71 d3.csv("data.csv", function(error, data) {
72   data.forEach(function(d) {
73     d.date = parseDate(d.date);
74     d.close = +d.close;
75   });
76 });

78 // Scale the range of the data
79 x.domain(d3.extent(data, function(d) { return d.date; }));
80 y.domain([0, d3.max(data, function(d) { return d.close; })]);

82 // Add the valueline path.
83 svg.append("path")
84   .attr("class", "line")
85   .attr("d", valueline(data));

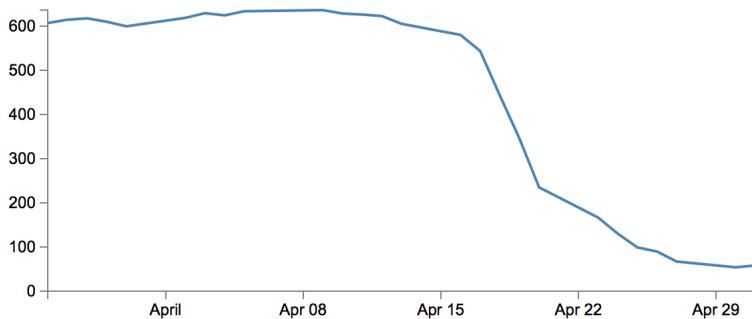
86 // Add the X Axis
87 svg.append("g")
88   .attr("class", "x_axis")
89   .attr("transform", "translate(0, " + height + ")");
90   .call(xAxis);

92 // Add the Y Axis
93 svg.append("g")
```

```
96     .attr("class", "y_axis")
97     .call(yAxis);
98 });
100 </script>
</body>
```

**Listing 2.4:** Example of use of the D3.js framework

legg til kilde på koden her. This graph would be appended to the body element of the html and look like this:



In the code the dynamic properties are visible as the x- and y-axis change its parameters based on the input data.



# Chapter 3

## Research

### 3.1 Related work

In the last decade the importance of security against attacks on large computer systems has grown rapidly. In 2004, the ACM workshop on Visualization and data mining for computer security presented NVisionIP: netflow visualizations of system state for security situational awareness[LYBL04]. This was one of the first tools to visualize NetFlow data. The visualization was based on either number of bytes transmitted or the number of flows to or from the hosts on the network.

In [LYL04] they discuss the use of NVisionIP to combat different security concerns. Most of the same attacks covered in this paper are relevant today, only in today's massive amounts of data, they may be way more difficult to discover.

- **Worm infection:** One of the most basic security function one might uncover. Worms usually spread by probing for other hosts. Filtering out hosts transmitting a lot of Flows with a single destination port, one could easily see which machines are infected and should be taken offline.
- **Compromised systems:** If a host is compromised, the attacker might install malware that allows the attacker to control the machine. Following this an attacker might turn a host into a file server. By detecting large volumes of traffic on certain ports one might discover such an attack.
- **Misuse:** Misuse of computer networks in order with terms of use etc.. An example is detecting if certain users have abnormal high volumes of traffic, and by inspecting in more detail one can uncover if this through one single application and not in accordance with the policies of the organization.
- **Port Scans:** When a large number of ports are used at a specific host it is easily identified by NVisionIP.

- **Denial of Service (DoS):** Denial of Service Attacks will be visible through spikes in traffic volume from the host attacking. If a host is attacked the same pattern is visible through high volumes in receiving traffic. Thus peaks in traffic is not necessarily an attack, but might be a result of a new release, or backup etc.

## 3.2 Initial research

In section 2.1 we see how the raw format of the NetFlow packets look. Norwegian University of Science and Technology (NTNU) Logical Volume Manager (LVM) Comparing how understandable this format is comparing to a visual representation will be the main object of this paper(omformulere. ikke helt riktig). How much more effective is visualization compared to the raw format read by machines.

To understand this, experiments will determine how quickly one can distinguish an attack from both the raw format, and the visual representation.

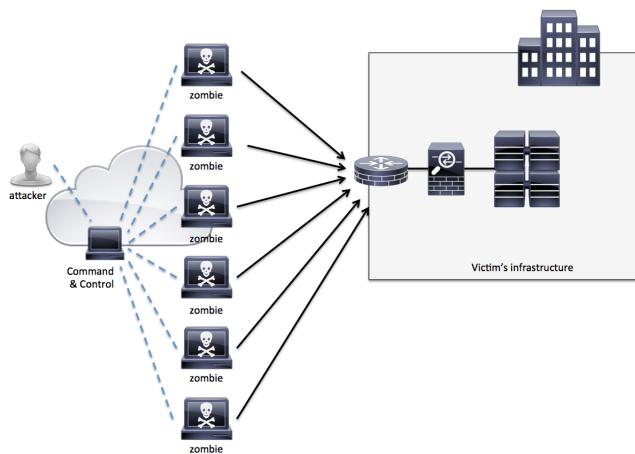
This is where D3.js will come to great use. It can be used to quickly develop simple interactive graphs that can be used to test theories up against each other.

To be able to identify an DDoS attack, one can look at it from two angles. By finding someone whom is attacking, or someone whom is being attacked. In this case we will look at the second scenario. As mentioned earlier simply a peak in flows is not enough grounds to establish an actual attack. First of all, one will need to look for patterns of similar incidents, and what lies behind them.

### 3.3 Traits of a DDoS attack

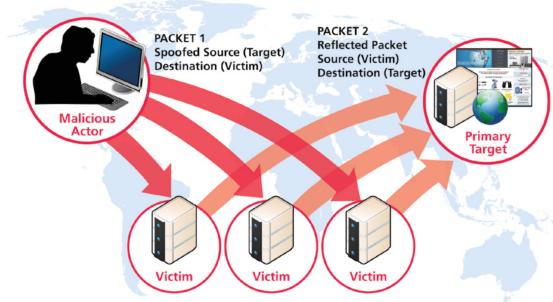
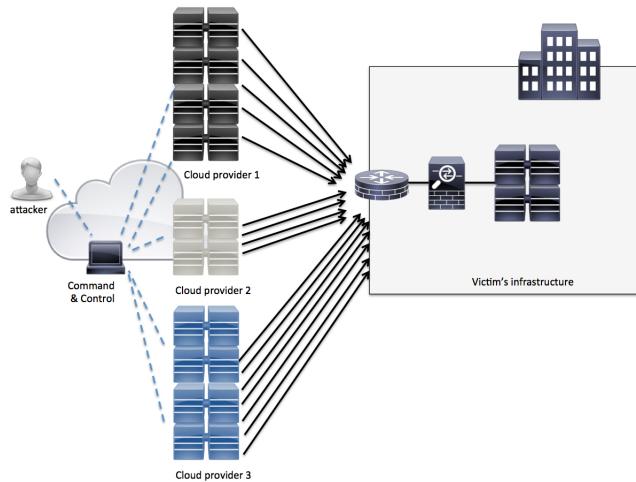
In a Distributed Denial of Service (DDoS) attack there are a large number of hosts performing the attack. In many cases a lot of them are not even aware they are a part on an attack. This is called a botnet, derived from the words robot and network. Using compromised systems, called zombies, gives the attacker control of a large enough amount of hosts to perform a volume-based DDoS attack.

**Figure 3.1:** Ten files in the provided files with the most flows [cis16]



Another new trend that has emerged is using large datacenters or cloud machines to launch these attacks. Either through renting or compromising them. As cloud providers are offering such large amounts of computers, this new platform is not only great for legitimate use, but also cyber-criminals.

Distributed Reflection Denial of Service attacks are becoming more and more popular. DrDoS techniques usually involve multiple victim host machines that unwillingly participate in a DDoS attack on the attacker's primary target. Requests to the victim host machines are redirected, or reflected, from the victim hosts to the target. Anonymity is one advantage of the DrDoS attack method. In a DrDoS attack, the primary target appears to be directly attacked by the victim host servers, not the actual attacker. This approach is called spoofing. Amplification is another advantage of the DrDoS attack method. By involving multiple victim servers, the attacker's initial request yields a response that is larger than what was sent, thus increasing the attack bandwidth.

**Figure 3.2:** Ten files in the provided files with the most flows [cis16]

### 3.3.1 Raw NetFlow format

In the preceding example, there are multiple flows for UDP port 80 (hex value 0050). In addition, there are also flows for TCP port 53 (hex value 0035) and TCP port 80 (hex value 0050).

The packets in these flows may be spoofed and may indicate an attempt to perform these attacks. It is advisable to compare the flows for TCP port 53 (hex value 0035) and TCP port 80 (hex value 0050) to normal baselines to aid in determining whether an attack is in progress.

```

router#show ip cache flow
IP packet size distribution (90784136 total packets):
 1-32 64 96 128 160 192 224 256 288 320 352 384 416 448 480
 .000 .698 .011 .001 .004 .005 .000 .004 .000 .000 .003 .000 .000 .000 .000
 512 544 576 1024 1536 2048 2560 3072 3584 4096 4608
 .000 .001 .256 .000 .010 .000 .000 .000 .000 .000 .000 .000 .000 .000 .000
IP Flow Switching Cache, 4456704 bytes
 1885 active, 63651 inactive, 59960004 added
 129803821 ager polls, 0 flow alloc failures
 Active flows timeout in 30 minutes
 Inactive flows timeout in 15 seconds
IP Sub Flow Cache, 402056 bytes
 0 active, 16384 inactive, 0 added, 0 added to flow
 0 alloc failures, 0 force free
 1 chunk, 1 chunk added
 last clearing of statistics never
Protocol      Total    Flows   Packets  Bytes  Packets  Active(Sec)  Idle(Sec)
-----      Flows     /Sec    /Flow   /Pkt   /Sec    /Flow     /Flow
TCP-Telnet    11393421    2.8      1    48     3.1     0.0     1.4
TCP-FTP       236        0.0      12    66     0.0     1.8     4.8
TCP-FTPD      21         0.0    13726  1294     0.0    18.4     4.1
TCP-WWW       22282        0.0    21    1020    0.1     4.1     7.3
TCP-X          719        0.0      1    40     0.0     0.0     1.3
TCP-BGP        1         0.0      1    40     0.0     0.0    15.0
TCP-Frag      70399        0.0      1    688     0.0     0.0    22.7
TCP-other     47861004   11.8      1    211    18.9     0.0     1.3
UDP-DNS       582        0.0      4    73     0.0     3.4    15.4
UDP-NTP       287252        0.0      1    76     0.0     0.0    15.5
UDP-other     310347        0.0      2    230     0.1     0.6    15.9
ICMP          11674        0.0      3    61     0.0    19.8     15.5
IPv6INIP      15         0.0      1   1132     0.0     0.0    15.4
GRE           4         0.0      1    48     0.0     0.0    15.3
Total:      59957957   14.8      1    196    22.5     0.0     1.5
SrcIf  SrcIPAddress  DstIf      DstIPAddress  Pr SrcP DstP  Pkts
G10/0  192.168.10.201 Gi0/1    192.168.60.102 06 0984 0050   1
G10/0  192.168.11.54  Gi0/1    192.168.60.158 06 0911 0035   3
G10/1  192.168.150.60 Gi0/0    10.89.16.226 06 0016 12CA   1
G10/0  192.168.10.17  Gi0/1    192.168.60.97 11 0B89 0050   1
G10/0  10.88.226.1   Gi0/1    192.168.202.22 11 007B 007B   1
G10/0  192.168.12.185 Gi0/1    192.168.60.239 11 0BD7 0050   1
G10/0  10.89.16.226  Gi0/1    192.168.150.60 06 12CA 0016   1
router#

```

**Figure 3.3:** Ten files in the provided files with the most flows



# Chapter 4

## D3.js and NetFlow

### 4.1 Using D3.js

Earlier in this paper it is mentioned that D3.js will be used to show examples of effective visualization of NetFlow data. It is assumed that the data has already been processed before it is made accessible to these examples. I was supplied with two months of anonymous data from UNINETT to get familiar with NetFlow and be able to use real data for my visualizations. This is anonymized data from January of 2012 from Trondheim and Oslo NetFlow collectors. This means millions of flows.

#### 4.1.1 Scope

The vast amounts of data should be presented with such a scope that is intuitive and easily understandable. Considering NetFlow packages is timestamped and sent from one source address to a specific destination address's port, one will have to chose which of these spectrum's to focus on. In the visual solution it is natural to combine these to represent the data.

##### IP spectrum

Choosing the address spectrum as the main focus, one will have to find a way to represent the entire IPv4 spectrum. This is alone a challenge, and when it comes to IPv6, it becomes practically impossible. This results in relying more on the pre-processing of the data and segregating the IP-addresses actually worth noticing. In the data provided by UNINETT it is possible to list for example the top 10 files in size, meaning more flows. In the data provided by UNINETT this search provided the results in figure 4.1.

From this simple preprocessing it is easy to see that in the time period between 1300-1400 on the 18th of January there was a clear peak in the number of flows having all the spots in the top 10. If we compare to the times with the lowest amount there is a different as they are a fraction of the others.

```
[eeglarse@iou2:/data/netflow$ find . -printf '%s %p\n'|sort -nr|head
14838848 ./oslo_gw/2012/01/18/nfcapd.201201181325
14729440 ./oslo_gw/2012/01/18/nfcapd.201201181335
14729284 ./oslo_gw/2012/01/18/nfcapd.201201181310
14720548 ./oslo_gw/2012/01/18/nfcapd.201201181330
14687944 ./oslo_gw/2012/01/18/nfcapd.201201181315
14651908 ./oslo_gw/2012/01/18/nfcapd.201201181340
14566420 ./oslo_gw/2012/01/18/nfcapd.201201181320
14563196 ./oslo_gw/2012/01/18/nfcapd.201201181305
14508804 ./oslo_gw/2012/01/18/nfcapd.201201181345
14472664 ./oslo_gw/2012/01/18/nfcapd.201201181300]
```

**Figure 4.1:** Ten files in the provided files with the most flows

```
[eeglarse@iou2:/data/netflow$ find . -printf '%s %p\n'|sort -nr|tail -100
849408 ./trd_gw1/2012/01/01/nfcapd.201201010920
848160 ./trd_gw1/2012/01/01/nfcapd.201201010745
842856 ./trd_gw1/2012/01/01/nfcapd.201201010725
834212 ./trd_gw1/2012/01/01/nfcapd.201201010655
832340 ./trd_gw1/2012/01/01/nfcapd.201201010640
830364 ./trd_gw1/2012/01/01/nfcapd.201201010555
828856 ./trd_gw1/2012/01/01/nfcapd.201201010845
821940 ./trd_gw1/2012/01/01/nfcapd.201201010900
816012 ./trd_gw1/2012/01/01/nfcapd.201201010905
804624 ./trd_gw1/2012/01/01/nfcapd.201201010735
802596 ./trd_gw1/2012/01/01/nfcapd.201201010545
799164 ./trd_gw1/2012/01/01/nfcapd.201201010635
780600 ./trd_gw1/2012/01/01/nfcapd.201201010650
772644 ./trd_gw1/2012/01/01/nfcapd.201201010610
760424 ./trd_gw1/2012/01/01/nfcapd.201201010705
758864 ./trd_gw1/2012/01/01/nfcapd.201201010720]
```

**Figure 4.2:** The smallest files from the provided files

```
[eeglarse@iou2:~$ cat test_180112.csv |cut -f 5 -d ',' |sort|uniq -c|sort|tail
17762 162.185.32.85
19878 161.222.192.123
21506 191.220.233.80
23995 161.223.1.164
37704 161.223.1.108
39759 159.152.145.176
49316 161.223.1.142
51467 190.49.180.97
61424 161.223.1.106
120976 192.239.62.2
```

**Figure 4.3:** Top ten used destination addresses within the timeframe 1300-1400, 18th of January

Trough this we create a .csv file containing the hour in question going further in detail. Analysing which destination address is the most requested is the next step.

Again one specific address is clearly separated from the others. At this point we have gotten such into detail on the dataset, it is time to find the reason behind the results we have found. These high numbers could be a DDoS attack, or other

```
eeglarse@iou2:~$ cat test_180112.csv |cut -f 4 -d ',' |sort|uniq -c|sort|tail -10
18502 161.222.192.123
18557 191.220.233.80
19338 162.185.32.85
29367 161.223.1.164
46376 192.239.62.2
47139 190.49.180.97
47844 161.223.1.108
50509 161.223.1.142
77527 159.152.145.176
83184 161.223.1.106
```

**Figure 4.4:** Top ten used destination addresses within the timeframe 1300-1400, 18th of January

types of attacks, but not does not necessary ill willed. If we look at the list of top IP-addresses sending packets.

We see that the same IP-address, 192.239.62.2, is here high up as well. Among hundreds of thousands of addresses in the spectrum.

To further investigate the activity on certain IP-addresses, it is possible to get the most popular ports on either one specific IP-address, or a list.

sjekke  
opp med  
portnr og

### Time spectrum

On the other side we have the time spectrum. In this case one looks at the amounts of flows within one time slot. Not down to the different IP-addresses. With the vast amount of IP-adresses this is not a suitable spectrum to present the data to find specific attacks etc.. On the other hand it could be used to monitor amounts of traffic over time or which ports are in use at certain times.

## 4.2 Number of flows to a certain host and port

This example shows how a simple graph can recognize a DDoS attack trough giving the option to see the number of netflows on different hosts and ports.

Rette  
opp

### 4.2.1 Scope in D3.js

In this section three modules of a solution is presented to show different levels of detail. It combines both the time spectrum and IP-spectrum to investigate the NetFlow data in different ways. The data from UNINETT required pre-processing before being made available to the D3.js solution. The bash script used can be found in AppendixB.

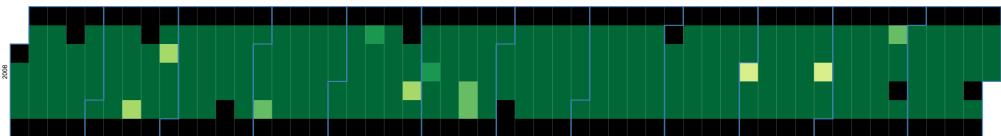
## File structure

With the bash scripts, thousands of files are created. Data is split into as small and many files as possible to reduce loading time at the user side, and to make sure the data is as comprehensible as possible. In section 2.2.2 we described the importance of pre-processing the data to highlight certain aspects of larger data sets. With these scripts we perform advanced preprocessing on extreme amounts of data to bring out the desired output to in the best way find deviations.

Knytte  
dette opp  
til prepro-  
cessing

## Overview

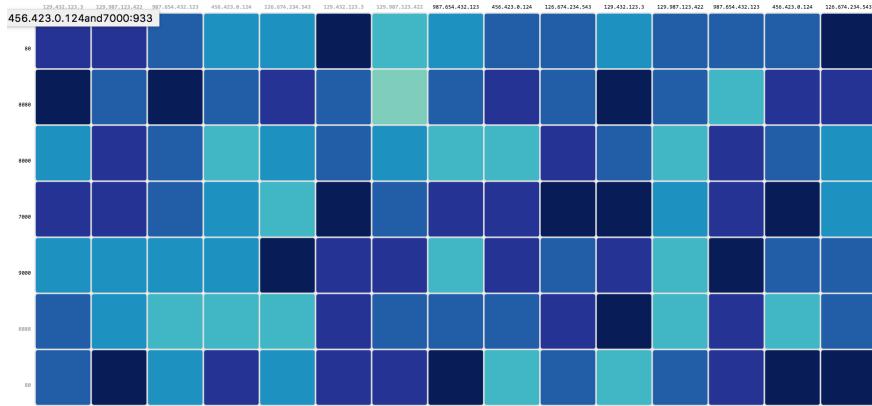
First we have the overview which in this case show an entire year separated into months, weeks and days. The purpose of this is to be able to quickly recognize patterns in the data that correlates to periodically activities as backup or launches of new software as mentioned in 3.1. For example a weekly backup will create similar levels of data usage at specific times each week.



**Figure 4.5:** Top ten used destination addresses within the timeframe 1300-1400, 18th of January

## IP-addresses and ports

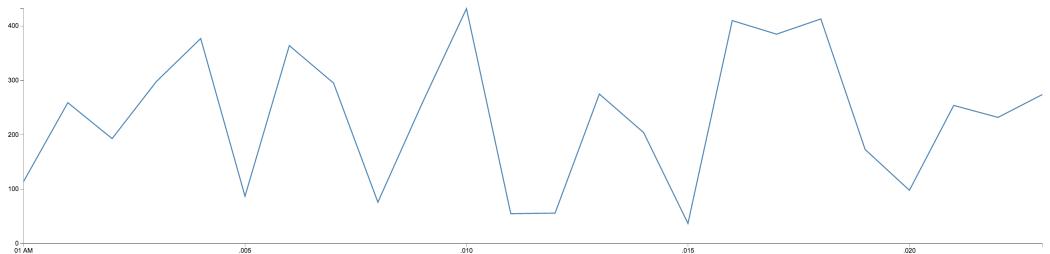
For each day there are millions of different combinations of which IP-addresses and ports that send flows between each other. Through pre-processing it is possible to distinguish which IP-addresses are the most popular each day, and thus find the ports they use the most. This visualization shows the number of flows for each of these combinations through a heatmap. A heatmap means it distinguishes values through a color range based on the highest values in the data set.



**Figure 4.6:** Top ten used destination addresses within the timeframe 1300-1400, 18th of January

### 24-hour chart

When a IP-address and port is selected for a specific day, the next scope is to look at the data in more detail. Using the time-spectrum this graph shows the 24-hour lapse and the amount of flows at each time.



**Figure 4.7:** Top ten used destination addresses within the timeframe 1300-1400, 18th of January

### 4.2.2 Pros and cons

Between the visual solution and the current command line based solution there are pros and cons. Some weighing in more than others depending on the result needed from the solution.

#### Visual solution

| Pros  | Cons  |
|---|---|
| <p>Very intuitive and simple to use. Meaning the need for competence within nfdump etc. is not as high, and lowers the threshold to use the tool.</p> <p>Visual interpretation of data is quicker and easier for the human mind to comprehend. Patterns and other aberrations is more likely to be discovered that would not be as visible in a test based solution.</p> <p>As mentioned in 2.2.3, a goal is to show only the data relevant to the user. Meaning the data that is not pertinent the user is never made visible.</p> | <p>A visual solution might limit the possibilities that exist in an advanced command line based solution. Meaning that a visual element is limited to some extent. Not showing the whole picture.</p> |

**Table 4.1:** Pros and cons of a visual solution

#### Command line solution

referere  
til noe  
i back-  
ground  
eller en  
referanse,  
i pkt 2 av  
pros for  
visuell  
løsning

| Pros   | Cons  |
|--|---|
| A command line solution possess a wide range of very specific commands as in nfdump, providing the possibility to do very distinct searches. | Such a tool is time consuming and difficult to master. Being very reliable on the searches being correct and providing the exact answer one is looking for requires a fair amount of skill.<br><br>Patterns are not easy to recognize when using a command line as it might involve several searches to reveal the information desired. |

**Table 4.2:** Pros and cons of a command line solution



# Chapter 5

## Discussion

Diskutere det som kommer opp under testing.



# Chapter 6 Challenges

## 6.1 Large data sets

When visualizing big data the main challenge is to effectively show the core message of the data. Considering one hour of the data provided from UNINETT, there is almost 400,000 different IP-adresses. And the amounts of flows is in the millions.

In section 2.2.1 good visualization is said to be able to present many numbers in a small space, make large data sets coherent, and reveal data at several levels of detail. I chose to create individual modules with D3.js, with each covering a different layer of detail.

### 6.1.1 IP-spectrum

As mentioned the range of the Internet Protocol version 4 (IPv4) is large, and with the emergence of IPv6 there is a challenge present. In 4.2.1 this was resolved with pre-processing of the data for a specific task. In other cases such a limitation on the number of IP-addresses represented wouldn't be satisfying.

### 6.1.2 Increasing number of flows

The amount of data sent these days are expanding quickly. This means the number of flows will follow, and a visual solution will need to be scalable to handle this increase. In the solution in 4.2 the first overview could scale as the heatmap takes in consideration the range of total flows. The next part of the solution is capable of handling larger amounts of flows, but the limitation in the number of ports and IP-addresses could limit certain attacks as ... . As mentioned with the increase in IPv6, the visual representation itself could become less intuitive and simple as the possible combinations of IP-addresses and ports would be too big to yield results(, or reveal attacks). As the last graph is based on simply number of flows over time,

Finne  
konkrete  
eksem-  
pler fra  
tidligere i  
oppgaven

it could prove effective even as the number of flows increase. It could be harder to reveal minor spikes in flows as it will be less distinguished than before.

## 6.2 Live updates

During the development of the solution made in this paper the data has been static and the time used to process the data and present it has not been taken into consideration. If a visual solution is to be used to monitor traffic as it is being collected the processing of the live data needs to be improved to be able to detect attacks and deviations as they are happening. nfdump is a powerful tool, but the amounts of data generated are so large, and will increase further, that the possibility to filter out specific information is to time consuming and require a lot of resources from UNINETT. It was not made a priority in this paper as the functionality of the visual solution up against the working nfdump was the main focus, not if it met requirements in speed or live monitoring.

# Chapter 7

## Concluding Remarks and Future Work

What has been created in this paper is merely a fraction of what is possible with the NetFlow data and D3.js. Further analysis of which information that should be included in the visual presentation could provide solution capable of discovering anomalies not being picked up by current systems.

### 7.1 Recommendations for Future Work

#### 7.1.1 Further testing

The data provided from UNINETT was not confirmed to contain any attacks, but showed although that the visual representation of the data revealed similar patterns as discovered through the current nfdump solution. Further on the visual solution should be tested up against larger sets of data to prove scalability and the ability to reveal patterns over bigger time slots.

#### 7.1.2 Customized searches

Kanskje ta med i further work? As showed in section 4.1.1 the commands in nfdump can reveal detailed information in few commands, but there is still a few searches that is required to obtain the requested data. By creating an Application Programming Interface (API) that simplifies the extraction of the information used in the visual elements, the possibility to display live NetFlow data is obtainable.



## References

- [Bos16] Mike Bostock. D3.js - data-driven documents, 2016.
- [cis16] 2016.
- [Cla04] Benoit Claise. Cisco systems netflow services export version 9. 2004.
- [CO02] K. G. Coffman and A. M. Odlyzko. *Handbook of Massive Data Sets*, chapter Internet Growth: Is There a “Moore’s Law” for Data Traffic?, pages 47–93. Springer US, Boston, MA, 2002.
- [EKMV04] Cristian Estan, Ken Keys, David Moore, and George Varghese. Building a better netflow. *SIGCOMM Comput. Commun. Rev.*, 34(4):245–256, August 2004.
- [Fri08] Vitaly Friedman. Data visualization and infographics smashing magazine, 2008.
- [Jam07] G Harry Jamieson. *Visual communication: More than meets the eye*. Intellect Books, 2007.
- [LYBL04] K. Lakkaraju, W. Yurcik, R. Bearavolu, and A. J. Lee. Nvisionip: an interactive network flow visualization tool for security. In *Systems, Man and Cybernetics, 2004 IEEE International Conference on*, volume 3, pages 2675–2680 vol.3, Oct 2004.
- [LYL04] Kiran Lakkaraju, William Yurcik, and Adam J. Lee. Nvisionip: Netflow visualizations of system state for security situational awareness. In *Proceedings of the 2004 ACM Workshop on Visualization and Data Mining for Computer Security, VizSEC/DMSEC ’04*, pages 65–72, New York, NY, USA, 2004. ACM.
- [MBD<sup>+</sup>12] Andrew McAfee, Erik Brynjolfsson, Thomas H Davenport, DJ Patil, and Dominic Barton. Big data. *The management revolution. Harvard Bus Rev*, 90(10):61–67, 2012.
- [nfd16] 2016.
- [PMAØ04] Michalis Polychronakis, Evangelos P Markatos, Kostas G Anagnostakis, and Arne Øslebø. Design of an application programming interface for ip network monitoring. In *Network Operations and Management Symposium, 2004. NOMS 2004. IEEE/IFIP*, volume 1, pages 483–496. IEEE, 2004.

## 36 REFERENCES

- [sim16] 2016.
- [SST<sup>+</sup>16] Products Services, Cisco Software, Cisco Technologies, Management Instrumentation, Cisco NetFlow, Data Literature, and White Papers. Introduction to cisco ios netflow - a technical overview, 2016.
- [TB11] Brian Trammell and Elisa Boschi. An introduction to ip flow information export (ipfix). *Communications Magazine, IEEE*, 49(4):89–95, 2011.
- [TGM83] Edward R Tufte and PR Graves-Morris. *The visual display of quantitative information*, volume 2. Graphics press Cheshire, CT, 1983.
- [wik16] Data presentation architecture, 2016.

# **Todo list**

|   |    |
|---|----|
| bruke den siste referansen . . . . .  | 3  |
| Lime inn hvordan det ser ut i kommandolinjen . . . . .                              | 5  |
| sitere listen . . . . .   | 5  |
| ( . . . . .   | 5  |
| forklare? . . . . .   | 7  |
| sjekke opp med portnr og . . . . .  | 23 |
| Rette opp . . . . .   | 23 |
| Knytte dette opp til preprocessing . . . . .  | 24 |
| refere til noe i background eller en referanse, i pkt 2 av pros for visuell løsning | 26 |
| Finne konkrete eksempler fra tidligere i oppgaven . . . . .                         | 31 |



# Appendix A

```
1  <!DOCTYPE html>
3  <meta charset="utf-8">
5  <style> /* set the CSS */
7
9   body { font: 12px Arial;}
11
13  path {
15    stroke: steelblue;
16    stroke-width: 2;
17    fill: none;
18 }
19
21
23 .axis path,
24 .axis line {
25   fill: none;
26   stroke: grey;
27   stroke-width: 1;
28   shape-rendering: crispEdges;
29 }
30
32
34 rect.bordered {
35   stroke: #E6E6E6;
36   stroke-width: 2px;
37 }
38
39
41 text.mono {
42   font-size: 6pt;
43   font-family: Consolas, courier;
44   fill: #aaa;
45 }
46
48 text.axis-workweek {
49   fill: #000;
50 }
51
53 text.axis-worktime {
```

```

39         fill: #000;
      }
      .RdYlGn .q0-11{ fill:rgb(165,0,38)}
      .RdYlGn .q1-11{ fill:rgb(215,48,39)}
      .RdYlGn .q2-11{ fill:rgb(244,109,67)}
      .RdYlGn .q3-11{ fill:rgb(253,174,97)}
      .RdYlGn .q4-11{ fill:rgb(254,224,139)}
      .RdYlGn .q5-11{ fill:rgb(255,255,191)}
      .RdYlGn .q6-11{ fill:rgb(217,239,139)}
      .RdYlGn .q7-11{ fill:rgb(166,217,106)}
      .RdYlGn .q8-11{ fill:rgb(102,189,99)}
      .RdYlGn .q9-11{ fill:rgb(26,152,80)}
      .RdYlGn .q10-11{ fill:rgb(0,104,55)}

53 .header {
  height:50px;
  background:#F0F0F0;
  border:1px solid #CCC;
  width:960px;
  margin:0px auto;
}
</style>
<body>

63 <!— <div id="option">
  <input name="updateButton"
    type="button"
    value="Previous"
    onclick="previousData() " />
</div>
69 <div id="option">
71   <input name="updateButton"
    type="button"
    value="Next"
    onclick="nextData() " />
75 </div> —>

77 <!— load the d3.js library —>
<script src="http://d3js.org/d3.v3.min.js"<!--/script--&gt;
79 &lt;script type='text/javascript' src='knockout-min.js'><!--/script--&gt;

81 &lt;div style="font-size: 50px; text-align: center; margin-top: 20px; margin
  -bottom: 20px" data-bind="text: currentDate() "><!--/div--&gt;
82 &lt;div id="year" style="margin-top=20px"<!--/div--&gt;
83 &lt;div id="area1"<!--/div--&gt;
84 &lt;div id="area2" style="margin-bottom:20px"<!--/div--&gt;
85
&lt;script&gt;
87   function AppViewModel(){
     this.currentDay = ko.observable(1);
89
</pre>

```

```

availableCountries = ko.observableArray(['129.432.123.3',
91   '129.987.123.422',
   '987.654.432.123','456.423.0.124','126.674.234.543','129.432.123.3',
   '129.987.123.422',
   '987.654.432.123','456.423.0.124','126.674.234.543','129.432.123.3',
   '129.987.123.422',
   '987.654.432.123','456.423.0.124','126.674.234.543']);
availablePorts = ko.observableArray
([['80','8080','8000','7000','9000','8888','80','8080','8000','7000','9000','8888']);

this.chosenIp = ko.observable(availableCountries()[0]);
this.chosenPort = ko.observable(availablePorts()[0]);
this.currentYear = ko.observable(2008);
this.currentMonth = ko.observable(1);
this.currentDate = ko.observable('2008-01-01');
97 }

99 var parseDate = d3.time.format("%d-%b-%Y").parse;
101 ko.applyBindings(AppViewModel);

103 // ** Update data section (Called from the onclick)

105
106 function firstGraph(){
107   var width = 2000,
108     height = 250,
109     cellSize = 35;
110     date = "01-01-2012" // cell size
111
112   var percent = d3.format(".1%"),
113     format = d3.time.format("%Y-%m-%d");
114
115   var color = d3.scale.quantize()
116     .domain([- .5, .5])
117     .range(d3.range(11).map(function(d) { return "#q" + d + "-11"; }));
118
119   var svg = d3.select("#year").selectAll("svg")
120     .data(d3.range(2008, 2009))
121     .enter().append("svg")
122       .attr("width", width)
123       .attr("height", height)
124       .attr("class", "RdYlGn")
125       .append("g")
126       .attr("transform", "translate(" + ((width - cellSize * 53) / 2) +
127         ", " + (height - cellSize * 7 - 1) + ")");
128
129   svg.append("text")
130     .attr("transform", "translate(-6," + cellSize * 3.5 + ") rotate(-90")
131     .style("text-anchor", "middle")
132     .text(function(d) { return d; });

```

```

133 var rect = svg.selectAll(".day")
134   .data(function(d) { return d3.time.days(new Date(d, 0, 1), new Date
135     (d + 1, 0, 1)); })
136   .enter().append("rect")
137   .attr("class", "day")
138   .attr("width", cellSize)
139   .attr("height", cellSize)
140   .attr("x", function(d) { return d3.time.weekOfYear(d) * cellSize; })
141   .attr("y", function(d) { return d.getDay() * cellSize; })
142   .on("click", function(d){
143     heatmapChart("heatmap/" + d + ".csv");
144     currentDate(d);
145     console.log(currentDate());
146     updateGraph();
147   })
148   .datum(format);

149 rect.append("title")
150   .text(function(d) { return d; });

151 svg.selectAll(".month")
152   .data(function(d) { return d3.time.months(new Date(d, 0, 1), new
153     Date(d + 1, 0, 1)); })
154   .enter().append("path")
155   .attr("class", "month")
156   .attr("d", monthPath);

157 d3.csv("year/dji.csv", function(error, csv) {
158   if (error) throw error;

159   var data = d3.nest()
160     .key(function(d) { return d.Date; })
161     .rollup(function(d) { return d[0].High; })
162     .map(csv);

163   rect.filter(function(d) { return d in data; })
164     .attr("class", function(d) { return "day" + color(data[d] % 11); })
165     .select("title")
166       .text(function(d) { return d + ": " + (data[d]); });
167 });

168 function monthPath(t0) {
169   var t1 = new Date(t0.getFullYear(), t0.getMonth() + 1, 0),
170     d0 = t0.getDay(), w0 = d3.time.weekOfYear(t0),
171     d1 = t1.getDay(), w1 = d3.time.weekOfYear(t1);
172   return "M" + (w0 + 1) * cellSize + "," + d0 * cellSize
173     + "H" + w0 * cellSize + "V" + 7 * cellSize
174     + "H" + w1 * cellSize + "V" + (d1 + 1) * cellSize
175     + "H" + (w1 + 1) * cellSize + "V" + 0

```

```

    + "H" + (w0 + 1) * cellSize + "Z";
181 }

183 d3.select(self.frameElement).style("height", "2910px");

185 var margin = { top: 50, right: 0, bottom: 10, left: 60 },
187     width = 2000 - margin.left - margin.right,
188     height = 650 - margin.top - margin.bottom,
189     gridSize = Math.floor(width / 24),
190     legendElementWidth = gridSize,
191     buckets = 9,
192     colors = [ "#ffffd9", "#edf8b1", "#c7e9b4", "#7fcdbb", "#41b6c4", "#1d91c0",
193     "#225ea8", "#253494", "#081d58"], // alternatively
194     colorbrewer.YlGnBu[9]
195     days = availablePorts(),
196     times = availableCountries(),
197     datasets = ["heatmap/data.tsv", "heatmap/data2.tsv"];

198 var svg = d3.select("#area1")
199     .append("svg")
200     .attr("width", width + margin.left + margin.right)
201     .attr("height", height + margin.top + margin.bottom)
202     .append("g")
203     .attr("transform", "translate(" + margin.left + ", " + margin.
top + ")");
204
205 var dayLabels = svg.selectAll(".dayLabel")
206     .data(days)
207     .enter().append("text")
208     .text(function(d) { return d; })
209     .attr("x", 0)
210     .attr("y", function(d, i) { return i * gridSize; })
211     .style("text-anchor", "end")
212     .attr("transform", "translate(-6," + gridSize / 1.5 + ")")
213     .attr("class", function(d, i) { return ((i >= 0 && i <= 4)
? "dayLabel mono axis axis-workweek" : "dayLabel mono axis"); });
214
215 var timeLabels = svg.selectAll(".timeLabel")
216     .data(times)
217     .enter().append("text")
218     .text(function(d) { return d; })
219     .attr("x", function(d, i) { return i * gridSize; })
220     .attr("y", 0)
221     .style("text-anchor", "middle")
222     .attr("transform", "translate(" + gridSize / 2 + ", -6)")
223     .attr("class", function(d, i) { return ((i >= 7 && i <= 16)
? "timeLabel mono axis axis-worktime" : "timeLabel mono axis"); });
224
225 var heatmapChart = function(csvFile) {
    d3.tsv(csvFile,

```

```

227     function(d) {
228         return {
229             day: +d.day,
230             hour: +d.hour,
231             value: +d.value,
232             ip: d.ip,
233             port: d.port
234         };
235     },
236     function(error, data) {
237         var colorScale = d3.scale.quantile()
238             .domain([0, buckets - 1, d3.max(data, function(d) {
239                 return d.value; }))])
240             .range(colors);
241
242         var cards = svg.selectAll(".hour")
243             .data(data, function(d) { return d.day+':'+d.hour; });
244
245         cards.append("title");
246
247         cards.enter().append("rect")
248             .attr("x", function(d) { return (d.hour - 1) * gridSize;
249 })
250             .attr("y", function(d) { return (d.day - 1) * gridSize;
251 })
252             .attr("rx", 4)
253             .attr("ry", 4)
254             .attr("class", "hour bordered")
255             .attr("width", gridSize)
256             .attr("height", gridSize)
257             .style("fill", colors[0])
258             .on("click", function(d){
259                 chosenIp(d.ip);
260                 chosenPort(d.port)
261                 console.log(d.value+" "+chosenIp()+" "+chosenPort())
262             ;
263                 updateGraph();
264             });
265
266         cards.transition().duration(1000)
267             .style("fill", function(d) { return colorScale(d.value);
268 });
269
270         cards.select("title").text(function(d) { return d.ip +'and' +
271             d.port+':'+d.value; });
272
273         cards.exit().remove();
274
275     });
276
277     heatmapChart('/heatmap/2008-01-01.csv');

```

```

273     var datasetpicker = d3.select("#dataset-picker").selectAll(".dataset-button")
274         .data(datasets);
275
276     datasetpicker.enter()
277         .append("input")
278         .attr("value", function(d){ return "Dataset " + d })
279         .attr("type", "button")
280         .attr("class", "dataset-button")
281         .on("click", function(d) {
282             updateGraph();
283         });
284
285 function updateGraph(){
286     d3.select("#area2").selectAll("svg").remove();
287
288     var margin = {top: 30, right: 20, bottom: 30, left: 70},
289     width = 2000 - margin.left - margin.right,
290     height = 500 - margin.top - margin.bottom;
291
292     // Parse the date / time
293
294     // Set the ranges
295     var x = d3.time.scale().range([0, width]);
296     var y = d3.scale.linear().range([height, 0]);
297
298     // Define the axes
299     var xAxis = d3.svg.axis().scale(x)
300         .orient("bottom").ticks(5);
301
302     var yAxis = d3.svg.axis().scale(y)
303         .orient("left").ticks(5);
304
305     // Define the line
306     var valueline = d3.svg.line()
307         .x(function(d) { return x(d.date); })
308         .y(function(d) { return y(d.close); });
309
310     // Adds the svg canvas
311     var svg = d3.select("#area2")
312         .append("svg")
313             .attr("width", width + margin.left + margin.right)
314             .attr("height", height + margin.top + margin.bottom)
315         .append("g")
316             .attr("transform",
317                 "translate(" + margin.left + ", " + margin.top + ")");
318
319     // Get the data
320     d3.csv("chart/" + currentDate() + '_' + chosenIp() + '_' + chosenPort() + '.csv',
321             function(error, data) {

```

46 A. APPENDIX A

```
323     data.forEach(function(d) {  
324         d.date = +d.date;  
325         d.close = +d.close;  
326     });  
  
327     // Scale the range of the data  
328     x.domain(d3.extent(data, function(d) { return d.date; }));  
329     y.domain([0, d3.max(data, function(d) { return d.close; })]);  
  
331     // Add the valueline path.  
332     svg.append("path")  
333         .attr("class", "line")  
334         .attr("d", valueline(data));  
  
335     // Add the X Axis  
336     svg.append("g")  
337         .attr("class", "x axis")  
338         .attr("transform", "translate(0," + height + ")")  
339         .call(xAxis);  
  
341     // Add the Y Axis  
342     svg.append("g")  
343         .attr("class", "y axis")  
344         .call(yAxis);  
  
347     );  
348 };  
349  
350     updateGraph()  
351 };  
  
352 function changeIpOrPort() {  
  
353     // Get the data again  
354     d3.csv("data1.csv", function(error, data) {  
355         data.forEach(function(d) {  
356             d.date = parseDate(d.date);  
357             d.close = +d.close;  
358         });  
  
359         // Scale the range of the data again  
360         x.domain(d3.extent(data, function(d) { return d.date; }));  
361         y.domain([0, d3.max(data, function(d) { return d.close; })]);  
  
362         // Select the section we want to apply our changes to  
363         var svg = d3.select("#area2").transition();  
  
364         // Make the changes  
365         svg.select(".line") // change the line  
366             .duration(750)  
367             .attr("d", valueline(data));  
368         svg.select(".x.axis") // change the x axis
```

```
375     .duration(750)
376     .call(xAxis);
377     svg.select(".y.axis") // change the y axis
378     .duration(750)
379     .call(yAxis);
380   });
381 }
382 firstGraph();
383
384 </script>
385 </body>
```



# Appendix B

## Appendix B

Appendix B contains the scripts used to create .csv files from all the data made available from UNINETT.

A script that creates .csv files for every nfcapd file in a day. This script is run by another short script that runs it 31 times for each day.

```
\label{csv_daily}
104 #!/bin/bash
105 mkdir /home/eeglarose/flowtest/2012_02/$(printf "%02d" $1)
106 clock_converter(){
107     if [ $(($1 % 100)) -gt 59 ]; then
108         return $(($1 % 100))
109     fi
110     if [ $(($1 % 100)) -lt 60 ]; then
111         echo $(printf "%04d" $1)
112         return $(($1 % 100))
113     fi
114 }
115
116 for (( c=0; c<=2355; c += 5 ))
117 do
118     nfdump -r $(printf "%02d" $1)/nfcapd.201202$(printf "%02d" $1)$(
119         clock_converter $c ) -n 10 -s srcip -o csv > /home/eeglarose/
120         flowtest/2012_02/$(printf "%02d" $1)/$(clock_converter $c ).csv
121 done
```

**Listing B.1:** Creates .csv files for every nfcapd file in a day

A script that fetches the total amount of flows for each day and creates a file with the values.

```
120
121
122 clock_converter(){
123     if [ $(($1 % 100)) -gt 59 ]; then
124         return $(($1 % 100))
```

```

126     fi
127     if [ $($((\$1 % 100)) -lt 60 ]; then
128         echo $(printf "%04d" \$1)
129         return $($((\$1 % 100)))
130     fi
131 }
132
133 for (( c=0; c<=2355; c += 5 ))
134 do
135     total_file=$(awk -F' , ' 'NR == 15 { print \$1}' /home/eeglarse/
136     flowtest/2012_02/$(printf "%02d" \$1)/$( clock_converter \$c ).csv)
137     echo $total_file >> testfile2\$1.csv
138 done
139
140 awk '{s+=$1} END {print s}' testfile2\$1.csv >>datefile2.csv

```

**Listing B.2:** Total amount of flows for each day

A script that finds the top 10 used IP-adresses for each day.

```

142 nfdump -R /data/netflow/oslo_gw/2012/01/$(printf "%02d" \$1)/nfcapd
        .201201$(printf "%02d" \$1)0000:nfcapd.201201$(printf "%02d" \$1)2355
        -n 10 -s dstip -o csv > /home/eeglarse/flowtest/top10/$(printf "%02d" \$1).csv

```

**Listing B.3:** Top 10 used IP-adresses for each day

A script that creates a list the top 10 most popular ports, based on the 10 most popular IP-addresses.

```

1 ip_string=''
2 ip=$(awk -F' , ' 'NR == 2 { print \$5}' /home/eeglarse/flowtest/top10/$(
        printf "%02d" \$1).csv)
ip_string+=`dst ip '$ip' or '
4 ip=$(awk -F' , ' 'NR == 3 { print \$5}' /home/eeglarse/flowtest/top10/$(
        printf "%02d" \$1).csv)
ip_string+=`dst ip '$ip' or '
6 ip=$(awk -F' , ' 'NR == 4 { print \$5}' /home/eeglarse/flowtest/top10/$(
        printf "%02d" \$1).csv)
ip_string+=`dst ip '$ip' or '
8 ip=$(awk -F' , ' 'NR == 5 { print \$5}' /home/eeglarse/flowtest/top10/$(
        printf "%02d" \$1).csv)
ip_string+=`dst ip '$ip' or '
10 ip=$(awk -F' , ' 'NR == 6 { print \$5}' /home/eeglarse/flowtest/top10/$(
        printf "%02d" \$1).csv)
ip_string+=`dst ip '$ip' or '
12 ip=$(awk -F' , ' 'NR == 7 { print \$5}' /home/eeglarse/flowtest/top10/$(
        printf "%02d" \$1).csv)
ip_string+=`dst ip '$ip' or '

```

```

14 ip=$(awk -F',' 'NR == 8 { print $5}' /home/eeglarse/flowtest/top10/$(
    printf "%02d" $1).csv)
ip_string+='dst_ip '$ip' or '
16 ip=$(awk -F',' 'NR == 9 { print $5}' /home/eeglarse/flowtest/top10/$(
    printf "%02d" $1).csv)
ip_string+='dst_ip '$ip' or '
18 ip=$(awk -F',' 'NR == 10 { print $5}' /home/eeglarse/flowtest/top10/$(
    printf "%02d" $1).csv)
ip_string+='dst_ip '$ip' or '
20 ip=$(awk -F',' 'NR == 11 { print $5}' /home/eeglarse/flowtest/top10/$(
    printf "%02d" $1).csv)
ip_string+='dst_ip '$ip
22

24 nfdump -R /data/netflow/oslo_gw/2012/01/$(printf "%02d" $1)/nfcapd
    .201201$(printf "%02d" $1)0000:nfcapd.201201$(printf "%02d" $1)2355
    -n 10 -s dstport $iplist -o csv > /home/eeglarse/flowtest/top10/
    top10port/$(printf "%02d" $1).csv

```

A script that uses the 10 most popular IP-adresses and their corresponding ports to find the number of flows sent to each port on each IP-address.

```

2 #!/bin/bash
4 for (( i = 1; i < 31; i++ )); do
5     iplist=()
6     ip=$(awk -F',' 'NR == 2 { print $5}' /home/eeglarse/flowtest/top10/$(
7         printf "%02d" $1).csv)
8     iplist[0]=$ip
9     ip2=$(awk -F',' 'NR == 3 { print $5}' /home/eeglarse/flowtest/top10/$(
10        printf "%02d" $1).csv)
11     iplist[1]=$ip2
12     ip=$(awk -F',' 'NR == 4 { print $5}' /home/eeglarse/flowtest/top10/$(
13        printf "%02d" $1).csv)
14     iplist[2]=$ip
15     ip=$(awk -F',' 'NR == 5 { print $5}' /home/eeglarse/flowtest/top10/$(
16        printf "%02d" $1).csv)
17     iplist[3]=$ip
18     ip=$(awk -F',' 'NR == 6 { print $5}' /home/eeglarse/flowtest/top10/$(
19        printf "%02d" $1).csv)
20     iplist[4]=$ip

```

52 B. APPENDIX B

```

ip=$(awk -F' , ' 'NR == 10 { print $5}' /home/eeglarose/flowtest/top10/$
    (printf "%02d" $i).csv)
22 iplist[8]=$ip
ip=$(awk -F' , ' 'NR == 11 { print $5}' /home/eeglarose/flowtest/top10/$
    (printf "%02d" $i).csv)
24 iplist[9]=$ip
portlist=()
26 ip=$(awk -F' , ' 'NR == 2 { print $5}' /home/eeglarose/flowtest/top10/
    top10port/$(printf "%02d" $i).csv)
portlist[0]=$ip
28 ip=$(awk -F' , ' 'NR == 3 { print $5}' /home/eeglarose/flowtest/top10/
    top10port/$(printf "%02d" $i).csv)
portlist[1]=$ip
30 ip=$(awk -F' , ' 'NR == 4 { print $5}' /home/eeglarose/flowtest/top10/
    top10port/$(printf "%02d" $i).csv)
portlist[2]=$ip
32 ip=$(awk -F' , ' 'NR == 5 { print $5}' /home/eeglarose/flowtest/top10/
    top10port/$(printf "%02d" $i).csv)
portlist[3]=$ip
34 ip=$(awk -F' , ' 'NR == 6 { print $5}' /home/eeglarose/flowtest/top10/
    top10port/$(printf "%02d" $i).csv)
portlist[4]=$ip
36 ip=$(awk -F' , ' 'NR == 7 { print $5}' /home/eeglarose/flowtest/top10/
    top10port/$(printf "%02d" $i).csv)
portlist[5]=$ip
38 ip=$(awk -F' , ' 'NR == 8 { print $5}' /home/eeglarose/flowtest/top10/
    top10port/$(printf "%02d" $i).csv)
portlist[6]=$ip
40 ip=$(awk -F' , ' 'NR == 9 { print $5}' /home/eeglarose/flowtest/top10/
    top10port/$(printf "%02d" $i).csv)
portlist[7]=$ip
42 ip=$(awk -F' , ' 'NR == 10 { print $5}' /home/eeglarose/flowtest/top10/
    top10port/$(printf "%02d" $i).csv)
portlist[8]=$ip
44 ip=$(awk -F' , ' 'NR == 11 { print $5}' /home/eeglarose/flowtest/top10/
    top10port/$(printf "%02d" $i).csv)
portlist[9]=$ip
46 for (( s = 0; s < 10; s++ )); do
    for (( j = 0; j < 10; j++ )); do
48     $(nfdump -R /data/netflow/oslo_gw/2012/01/$(printf "%02d" $i)/
        nfcapd.201201$(printf "%02d" $i)0000:nfcapd.201201$(printf "%02d"
        $i)2355 -n 10 -s dstport -o csv 'dst ip ${iplist[$s]} and dst port
        ${portlist[$j]}' -o csv)
        done
    done
done

```