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: Otto Wittner, 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: Otto Wittner, 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 likningsett av flervariable 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 foreslåtte 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 nøklene 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 nøkkelgenererings-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

List of Figures

Listings

List of Acronyms

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.

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
Background

1.1 NetFlow

Cisco IOS NetFlow creates an enviroment 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.

1.1.1 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 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+16]

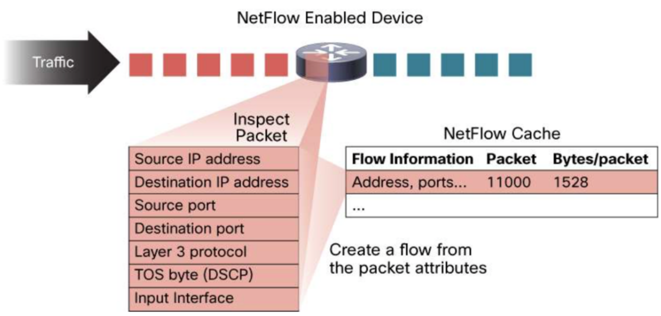


Figure 1.1: Creating a flow in the NetFlow cache siter

- 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 Autonomnous Systems (AS)
- Subnet mask for the source and destination addresses to calculate prefixes

– flags to examine Transmission Control Protocol (TPC) handshakes

Lime inn hvordan det ser ut i kommandolinjen

sitere listen

1.1.2 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 analyze the received flow data in different contexts, such as intrusion or traffic profiling.

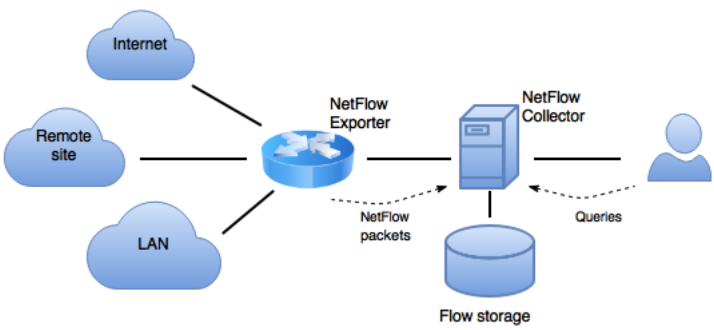


Figure 1.2: Figure of a simple NetFlow architecture

1.1.3 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 1.3 the collection process is depicted, and in figure 1.4 the processing of collected NetFlow data is shown.[nfd16]

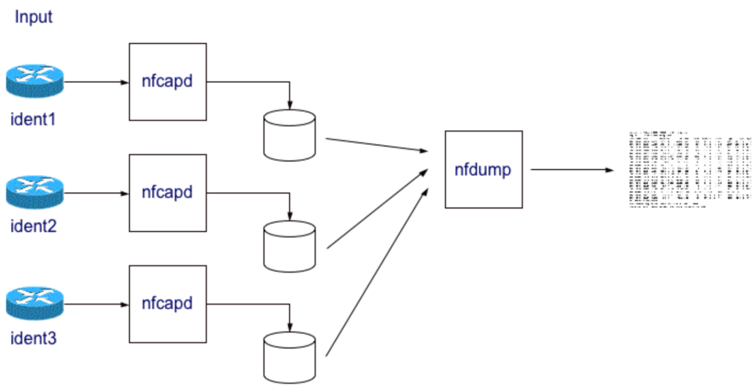


Figure 1.3: Example of dataset of random numbers where no pre-attentive processing is done

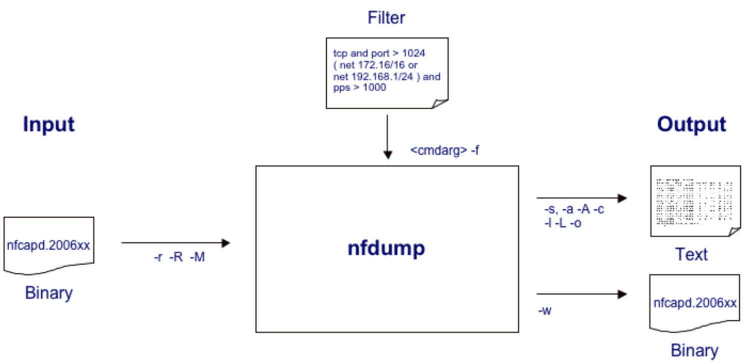


Figure 1.4: Example of dataset of random numbers where no pre-attentive processing is done

v5,v9 and IPfix

- **v5:** NetFlow v5 is definitely the most popular version of Cisco Netflow. It is fixed, maning 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.

– **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. forklare?

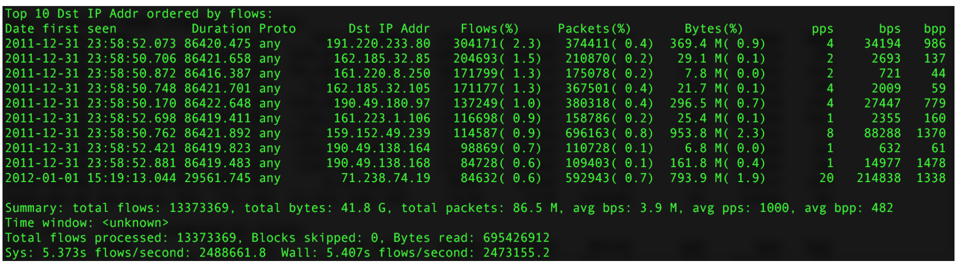
Example of use

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:

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

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 1.5.



```
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      191.220.233.80   304171( 2.3)   374411( 0.4)  369.4 M( 0.9)       4    34194   986
2011-12-31 23:58:50.706 86421.658 any       162.185.32.85   204693( 1.5)   210870( 0.2)   29.1 M( 0.1)       2     2693   137
2011-12-31 23:58:50.872 86416.387 any       161.220.8.250   171799( 1.3)   175078( 0.2)    7.8 M( 0.0)       2      721    44
2011-12-31 23:58:50.748 86421.701 any      162.185.32.105   171177( 1.3)   367501( 0.4)   21.7 M( 0.1)       4     2009    59
2011-12-31 23:58:50.170 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       161.223.1.106   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  1370
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 15:19:13.044 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 bps: 3.9 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 1.5: Example of dataset of random numbers where no pre-attentive processing is done

1.2 Data visualization

Data visualization refers to the techniques used to communicate data or information by encoding it as visual objects[sitere?]. Meaning that information is represented trough 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". Without visual presentation such data is too big to understand. This is the big reason for 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.

1.2.1 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.

1.2.2 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 trough 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.

Figure 1.6: Example of dataset of random numbers where no pre-attentive processing is done

Figure 1.7: Example of a dataset of random numbers where pre-attentive processing has been used to distinguish the occurrences of the number five

1.2.3 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:

- 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

1.3 D3.js

In this paper D3.js [Bos12] 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.

1.3.1 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:

```

var paragraphs = document.getElementsByTagName("p");
for (var i = 0; i < paragraphs.length; i++) {
  var paragraph = paragraphs.item(i);
  paragraph.style.setProperty("color", "white", null);
```

```
}
```

Listing 1.1: HTML example

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

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

Listing 1.2: D3.js example

D3.js also possess 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.

```
<!DOCTYPE html>
<meta charset="utf-8">
<style> /* set the CSS */

body { font: 12px Arial;}

path {
    stroke: steelblue;
    stroke-width: 2;
    fill: none;
}

.axis path,
.axis line {
    fill: none;
    stroke: grey;
    stroke-width: 1;
    shape-rendering: crispEdges;
}

</style>
<body>

<!-- load the d3.js library -->
<script src="http://d3js.org/d3.v3.min.js"></script>

<script>

// Set the dimensions of the canvas / graph
var margin = {top: 30, right: 20, bottom: 30, left: 50},
    width = 600 - margin.left - margin.right,
    height = 270 - margin.top - margin.bottom;

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

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

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

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

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

// Adds the svg canvas
var svg = d3.select("body")
    .append("svg")
        .attr("width", width + margin.left + margin.right)
        .attr("height", height + margin.top + margin.bottom)
    .append("g")
        .attr("transform",
              "translate(" + margin.left + "," + margin.top + ")");

// Get the data
d3.csv("data.csv", function(error, data) {
    data.forEach(function(d) {
        d.date = parseDate(d.date);
        d.close = +d.close;
    });

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

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

    // Add the X Axis
    svg.append("g")
        .attr("class", "x axis")
        .attr("transform", "translate(0," + height + ")")
        .call(xAxis);

    // Add the Y Axis
    svg.append("g")
        .attr("class", "y axis")
        .call(yAxis);
```

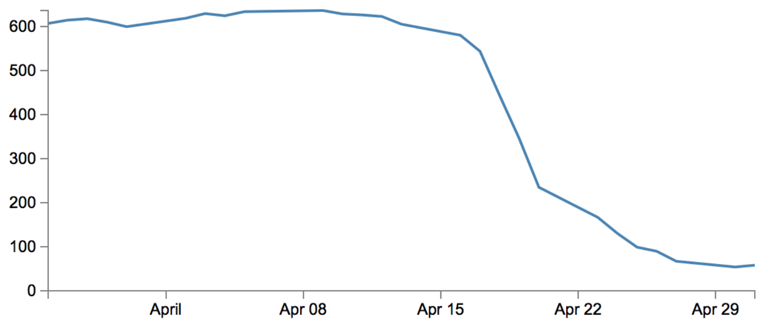
```

});

</script>
</body>
```

Listing 1.3: 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 2
Research

2.1 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 too 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 trough 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 trough spikes in traffic volume from the host attacking. If a host is attacked the same pattern is visible trough high volumes in receiving traffic. Thus peaks in traffic is not necessary an attack, but might be a result of a new release, or backup etc.

2.2 Initial research

In section 1.1 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.

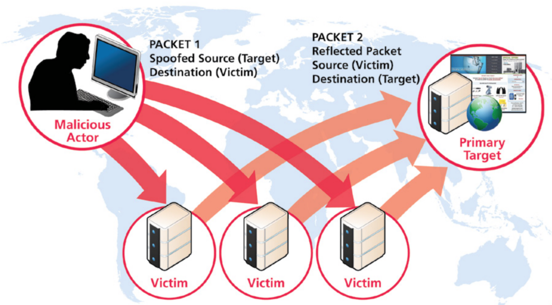
This is were D3.js will come to great use. It can be used to quickly develop simple interactive graphs that can be used to test theories up against each other.

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.

2.3 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. Another new trend that has emerged is using large datacenters or cloud machines to launch these attacks. Either trough 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 is becoming more and more popular. DrDoS techniques usually involve mulitple victim host machines that unwillingly participate in a DDoS attack on the attackers primary target. Requests to the victim host machines are redirected, or re ected, 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.



2.3.1 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.

Chapter 3 D3.js and NetFlow

3.1 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.

3.1.1 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 3.1.

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 3.1: 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 3.2: 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 3.3: 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 3.4: 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.

3.2 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

3.2.1 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.

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 2.1. For example a weekly backup will create similar levels of data usage at specific times each week.

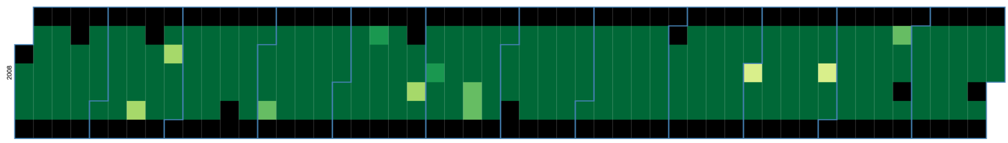


Figure 3.5: Top ten used destination addresses within the timeframe 1300-1400, 18th of January

IP-addresses and ports

For each days 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 trough a heatmap. A heatmap means it distinguishes values trough a color range based on the highest values in the data set.

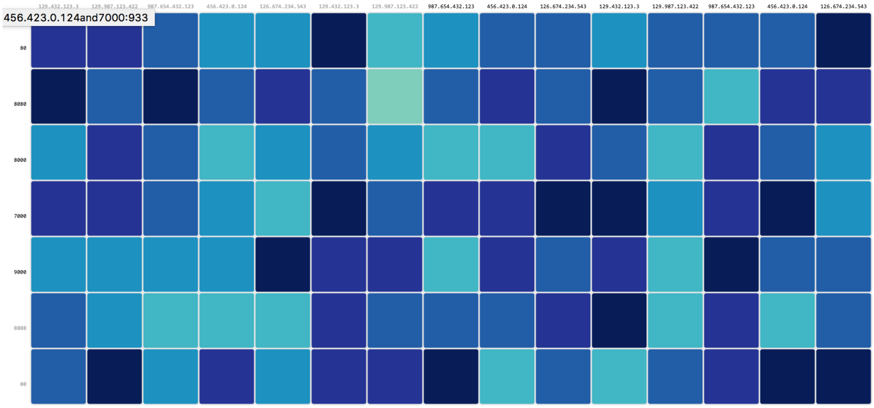


Figure 3.6: Top ten used destination addresses within the timeframe 1300-1400, 18th of January

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.

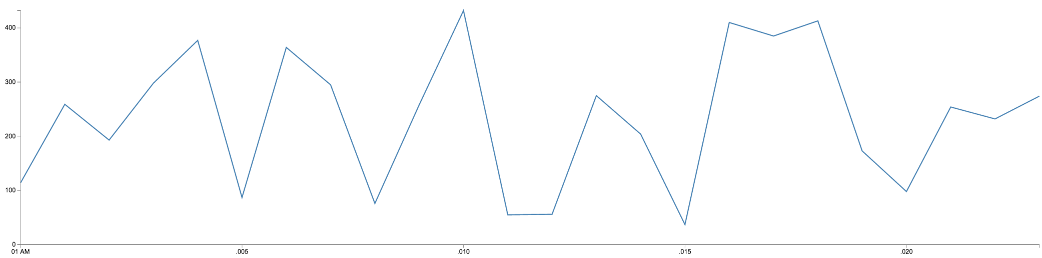


Figure 3.7: Top ten used destination addresses within the timeframe 1300-1400, 18th of January

Chapter 4
Challenges

4.1 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 1.2.1 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.

4.1.1 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 3.2.1 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.

4.1.2 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 3.2

References

[Bos12] Michael Bostock. D3. js. *Data Driven Documents*, 2012.

[cis16] 2016.

[Fri08] Vitaly Friedman. Data visualization and infographics smashing magazine, 2008.

[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.

[nfd16] 2016.

[SST+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.

[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

Appendix A

```
<!DOCTYPE html>
<meta charset="utf-8">
<style> /* set the CSS */

body { font: 12px Arial;}

path {
    stroke: steelblue;
    stroke-width: 2;
    fill: none;
}

.axis path,
.axis line {
    fill: none;
    stroke: grey;
    stroke-width: 1;
    shape-rendering: crispEdges;
}


rect.bordered {
          stroke: #E6E6E6;
          stroke-width:2px;
        }

        text.mono {
          font-size: 6pt;
          font-family: Consolas, courier;
          fill: #aaa;
        }

        text.axis-workweek {
          fill: #000;
        }

        text.axis-worktime {
```

```
            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)}

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

<!-- <div id="option">
    <input name="updateButton"
           type="button"
           value="Previous"
           onclick="previousData()" />
</div>

<div id="option">
    <input name="updateButton"
           type="button"
           value="Next"
           onclick="nextData()" />
</div> -->

<!-- load the d3.js library -->
<script src="http://d3js.org/d3.v3.min.js"></script>
<script type='text/javascript' src='knockout-min.js'></script>

<div style="font-size: 50px; text-align: center;margin-top: 20px;margin
    -bottom: 20px"data-bind="text: currentDate()"></div>
<div id="year" style="margin-top=20px"></div>
<div id="area1"></div>
<div id="area2" style="margin-bottom:20px"></div>

<script>

function AppViewModel(){
  this.currentDay = ko.observable(1);
```

```
  availableCountries = ko.observableArray(['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','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');
}

var parseDate = d3.time.format("%d-%b-%y").parse;

ko.applyBindings(AppViewModel);

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


function firstGraph(){
var width = 2000,
    height = 250,
    cellSize = 35;
    date = "01-01-2012" // cell size

var percent = d3.format(".1%"),
    format = d3.time.format("%Y-%m-%d");

var color = d3.scale.quantize()
    .domain([-.5, .5])
    .range(d3.range(11).map(function(d) { return "q" + d + "-11"; }));

var svg = d3.select("#year").selectAll("svg")
    .data(d3.range(2008, 2009))
  .enter().append("svg")
    .attr("width", width)
    .attr("height", height)
    .attr("class", "RdYlGn")
    .append("g")
    .attr("transform", "translate(" + ((width - cellSize * 53) / 2) + "
    ," + (height - cellSize * 7 - 1) + ")");

svg.append("text")
    .attr("transform", "translate(-6," + cellSize * 3.5 + ")rotate(-90)
    ")
    .style("text-anchor", "middle")
    .text(function(d) { return d; });
```

```
var rect = svg.selectAll(".day")
    .data(function(d) { return d3.time.days(new Date(d, 0, 1), new Date
    (d + 1, 0, 1)); })
   .enter().append("rect")
    .attr("class", "day")
    .attr("width", cellSize)
    .attr("height", cellSize)
    .attr("x", function(d) { return d3.time.weekOfYear(d) * cellSize;
    })
    .attr("y", function(d) { return d.getDay() * cellSize; })
    .on("click", function(d){
      heatmapChart("heatmap/"+d+".csv");
      currentDate(d);
      console.log(currentDate());
      updateGraph();
    })
    .datum(format);

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

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

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

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

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

function monthPath(t0) {
  var t1 = new Date(t0.getFullYear(), t0.getMonth() + 1, 0),
      d0 = t0.getDay(), w0 = d3.time.weekOfYear(t0),
      d1 = t1.getDay(), w1 = d3.time.weekOfYear(t1);
  return "M" + (w0 + 1) * cellSize + "," + d0 * cellSize
      + "H" + w0 * cellSize + "V" + 7 * cellSize
      + "H" + w1 * cellSize + "V" + (d1 + 1) * cellSize
      + "H" + (w1 + 1) * cellSize + "V" + 0
```

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

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


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

      var svg = d3.select("#area1")
          .append("svg")
          .attr("width", width + margin.left + margin.right)
          .attr("height", height + margin.top + margin.bottom)
          .append("g")
          .attr("transform", "translate(" + margin.left + "," + margin.top + ")");

      var dayLabels = svg.selectAll(".dayLabel")
          .data(days)
          .enter().append("text")
            .text(function (d) { return d; })
            .attr("x", 0)
            .attr("y", function (d, i) { return i * gridSize; })
            .style("text-anchor", "end")
            .attr("transform", "translate(-6," + gridSize / 1.5 + ")")
            .attr("class", function (d, i) { return ((i >= 0 && i <= 4) ? "dayLabel mono axis axis-workweek" : "dayLabel mono axis"); });

      var timeLabels = svg.selectAll(".timeLabel")
          .data(times)
          .enter().append("text")
            .text(function(d) { return d; })
            .attr("x", function(d, i) { return i * gridSize; })
            .attr("y", 0)
            .style("text-anchor", "middle")
            .attr("transform", "translate(" + gridSize / 2 + ", -6)")
            .attr("class", function(d, i) { return ((i >= 7 && i <= 16) ? "timeLabel mono axis axis-worktime" : "timeLabel mono axis"); });

      var heatmapChart = function(csvFile) {
        d3.tsv(csvFile,
```

```
        function(d) {
          return {
            day: +d.day,
            hour: +d.hour,
            value: +d.value,
            ip: d.ip,
            port: d.port
          };
        },
        function(error, data) {
          var colorScale = d3.scale.quantile()
              .domain([0, buckets - 1, d3.max(data, function (d) { return d.value; })])
              .range(colors);

          var cards = svg.selectAll(".hour")
              .data(data, function(d) {return d.day+':'+d.hour;});

          cards.append("title");

          cards.enter().append("rect")
              .attr("x", function(d) { return (d.hour - 1) * gridSize; })
              .attr("y", function(d) { return (d.day - 1) * gridSize; })
              .attr("rx", 4)
              .attr("ry", 4)
              .attr("class", "hour bordered")
              .attr("width", gridSize)
              .attr("height", gridSize)
              .style("fill", colors[0])
              .on("click", function(d){
                chosenIp(d.ip);
                chosenPort(d.port)
                console.log(d.value+" "+" "+chosenIp()+" "+chosenPort());
                updateGraph();
              });

          cards.transition().duration(1000)
              .style("fill", function(d) { return colorScale(d.value); });

          cards.select("title").text(function(d) { return d.ip +'and' + d.port+':'+d.value; });

          cards.exit().remove();

        });
      };

      heatmapChart('/heatmap/2008-01-01.csv');
```

```
      var datasetpicker = d3.select("#dataset-picker").selectAll(".dataset-button")
        .data(datasets);

      datasetpicker.enter()
        .append("input")
        .attr("value", function(d){ return "Dataset " + d })
        .attr("type", "button")
        .attr("class", "dataset-button")
        .on("click", function(d) {
          updateGraph();
        });

function updateGraph(){
d3.select("#area2").selectAll("svg").remove();

var margin = {top: 30, right: 20, bottom: 30, left: 70},
width = 2000 - margin.left - margin.right,
height = 500 - margin.top - margin.bottom;

// Parse the date / time


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

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

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

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

// Adds the svg canvas
var svg = d3.select("#area2")
    .append("svg")
        .attr("width", width + margin.left + margin.right)
        .attr("height", height + margin.top + margin.bottom)
    .append("g")
        .attr("transform",
              "translate(" + margin.left + "," + margin.top + ")");

// Get the data
d3.csv("chart/"+currentDate()+'_'+chosenIp()+'_'+chosenPort()+'.csv',
    function(error, data) {
```

```
    data.forEach(function(d) {
        d.date = +d.date;
        d.close = +d.close;
    });

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

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

    // Add the X Axis
    svg.append("g")
        .attr("class", "x axis")
        .attr("transform", "translate(0," + height + ")")
        .call(xAxis);

    // Add the Y Axis
    svg.append("g")
        .attr("class", "y axis")
        .call(yAxis);

      });
    };

    updateGraph()
  };

function changeIpOrPort() {

    // Get the data again
    d3.csv("data1.csv", function(error, data) {
        data.forEach(function(d) {
        d.date = parseDate(d.date);
        d.close = +d.close;
      });

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

    // Select the section we want to apply our changes to
    var svg = d3.select("#area2").transition();

    // Make the changes
        svg.select(".line")   // change the line
            .duration(750)
            .attr("d", valueline(data));
        svg.select(".x.axis") // change the x axis
```

```
            .duration(750)
            .call(xAxis);
        svg.select(".y.axis") // change the y axis
            .duration(750)
            .call(yAxis);

    });
  }

firstGraph();

</script>
</body>
```

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.

```
#!/bin/bash
mkdir /home/eeglarse/flowtest/2012_02/$(printf "%02d" $1)
clock_converter(){
  if [ $(($1 % 100)) -gt 59 ]; then
    return $(($1 % 100))
  fi
  if [ $(($1 % 100)) -lt 60 ]; then
    echo $(printf "%04d" $1)
    return $(($1 % 100))
  fi
  }

for (( c=0; c<=2355; c += 5 ))
do
   nfdump -r $(printf "%02d" $1)/nfcapd.201202$(printf "%02d" $1)$(
     clock_converter $c ) -n 10 -s srcip -o csv > /home/eeglarse/
     flowtest/2012_02/$(printf "%02d" $1)/$( clock_converter $c ).csv
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.

```


clock_converter(){
  if [ $(($1 % 100)) -gt 59 ]; then
    return $(($1 % 100))
  fi
```

```
  if [ $(($1 % 100)) -lt 60 ]; then
    echo $(printf "%04d" $1)
    return $(($1 % 100))
  fi
  }


for (( c=0; c<=2355; c += 5 ))
do
    total_file=$(awk -F',' 'NR == 15 { print $1}' /home/eeglarse/flowtest/2012_02/$(printf "%02d" $1)/$( clock_converter $c ).csv)
    echo $total_file >> testfile2$1.csv
done

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.

```

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.

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

```
ip_string+='dst ip '$ip' or '
ip=$(awk -F',' 'NR == 9 { print $5}' /home/eeglarse/flowtest/top10/$(
    printf "%02d" $1).csv)
ip_string+='dst ip '$ip' or '
ip=$(awk -F',' 'NR == 10 { print $5}' /home/eeglarse/flowtest/top10/$(
    printf "%02d" $1).csv)
ip_string+='dst ip '$ip' or '
ip=$(awk -F',' 'NR == 11 { print $5}' /home/eeglarse/flowtest/top10/$(
    printf "%02d" $1).csv)
ip_string+='dst ip '$ip


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.

```

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

```
    ip=$(awk -F',' 'NR == 11 { print $5}' /home/eeglarse/flowtest/top10/$(printf "%02d" $i).csv)
    iplist[9]=$ip
    portlist=()
    ip=$(awk -F',' 'NR == 2 { print $5}' /home/eeglarse/flowtest/top10/top10port/$(printf "%02d" $i).csv)
    portlist[0]=$ip
    ip=$(awk -F',' 'NR == 3 { print $5}' /home/eeglarse/flowtest/top10/top10port/$(printf "%02d" $i).csv)
    portlist[1]=$ip
    ip=$(awk -F',' 'NR == 4 { print $5}' /home/eeglarse/flowtest/top10/top10port/$(printf "%02d" $i).csv)
    portlist[2]=$ip
    ip=$(awk -F',' 'NR == 5 { print $5}' /home/eeglarse/flowtest/top10/top10port/$(printf "%02d" $i).csv)
    portlist[3]=$ip
    ip=$(awk -F',' 'NR == 6 { print $5}' /home/eeglarse/flowtest/top10/top10port/$(printf "%02d" $i).csv)
    portlist[4]=$ip
    ip=$(awk -F',' 'NR == 7 { print $5}' /home/eeglarse/flowtest/top10/top10port/$(printf "%02d" $i).csv)
    portlist[5]=$ip
    ip=$(awk -F',' 'NR == 8 { print $5}' /home/eeglarse/flowtest/top10/top10port/$(printf "%02d" $i).csv)
    portlist[6]=$ip
    ip=$(awk -F',' 'NR == 9 { print $5}' /home/eeglarse/flowtest/top10/top10port/$(printf "%02d" $i).csv)
    portlist[7]=$ip
    ip=$(awk -F',' 'NR == 10 { print $5}' /home/eeglarse/flowtest/top10/top10port/$(printf "%02d" $i).csv)
    portlist[8]=$ip
    ip=$(awk -F',' 'NR == 11 { print $5}' /home/eeglarse/flowtest/top10/top10port/$(printf "%02d" $i).csv)
    portlist[9]=$ip
    for (( s = 0; s < 10; s++ )); do
        for (( j = 0; j < 10; j++ )); do
          $(nfdump -R /data/netflow/oslo_gw/2012/01/$(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
```