



NTNU – Trondheim
Norwegian University of
Science and Technology

Visualization of large scale Netflow data

Nicolai Eeg-Larsen

Submission date: April 2016
Responsible professor: Firstname Lastname, Affiliation
Supervisor: Firstname Lastname, Affiliation

Norwegian University of Science and Technology
Department of Telematics

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.

Contents

List of Figures	vii
List of Tables	ix
List of Algorithms	xi
1 Background	1
1.1 NetFlow	1
1.1.1 How does it work?	1
1.1.2 Main components	3
1.2 Data visualization	3
1.2.1 Characteristics	4
1.2.2 Visual perception	4
1.2.3 Data presentation architecture	5
1.3 D3.js	6
1.3.1 How does it work?	6
2 Research	11
2.1 Related work	11
2.2 Initial research	12
2.3 Traits of a DDoS attack	13
2.3.1 Raw NetFlow format	15
2.4 Using D3.js	15
2.4.1 Scope	16
2.4.2 Number of flows to a certain host and port	18
3 Challenges	19
3.1 Large data sets	19
References	21

List of Figures

1.1	Creating a flow in the NetFlow cache siter	2
1.2	Figure of a simple NetFlow architecture	3
1.3	Example of dataset of random numbers where no pre-attentive processing is done	5
1.4	Example of a dataset of random numbers where pre-attentive processing has been used to distinguish the occurrences of the number five	5
2.1	Ten files in the provided files with the most flows	15
2.2	Ten files in the provided files with the most flows	16
2.3	The smallest files from the provided files	17
2.4	Top ten used destination addresses within the timeframe 1300-1400, 18th of January	17
2.5	Top ten used destination addresses within the timeframe 1300-1400, 18th of January	18

List of Tables

List of Algorithms

Chapter 1

Background

1.1 NetFlow

Cisco IOS NetFlow creates an environment that has 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 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

2 1. 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.

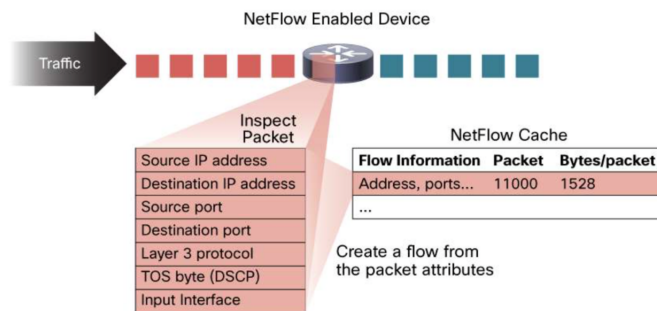


Figure 1.1: Creating a flow in the NetFlow cache siter

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

- flags to examine TCP handshakes

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

sitere
listen

1.1.2 Main components

A typical setup 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.

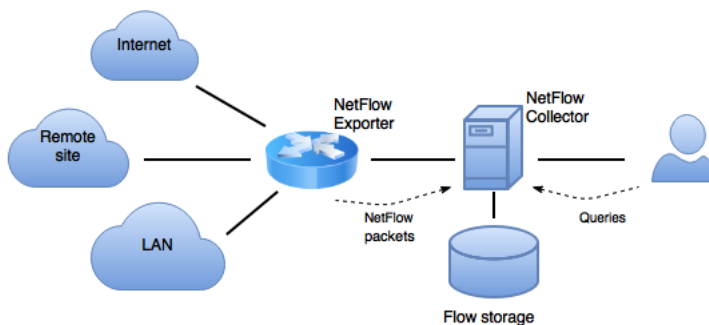


Figure 1.2: Figure of a simple NetFlow architecture

1.2 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 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.

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

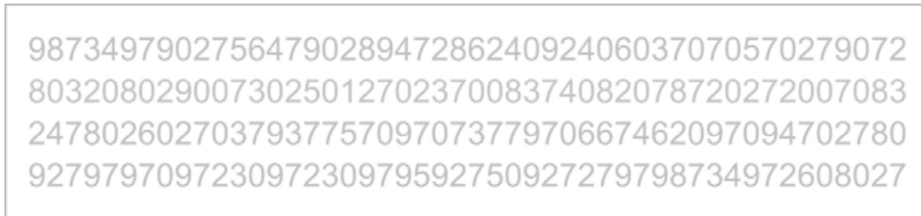
In his book from 1983, *The Visual Display of Quantitative Information* [TGM83], Edward Tufte defines characteristics any effective graphical representation should contain as:

- 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 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.

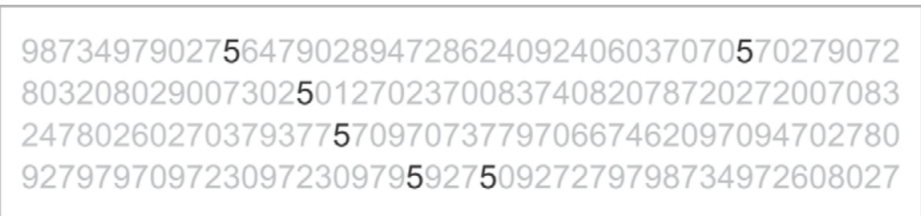


```

987349790275647902894728624092406037070570279072
803208029007302501270237008374082078720272007083
247802602703793775709707377970667462097094702780
927979709723097230979592750927279798734972608027

```

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



```

987349790275647902894728624092406037070570279072
803208029007302501270237008374082078720272007083
247802602703793775709707377970667462097094702780
927979709723097230979592750927279798734972608027

```

Figure 1.4: 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:

6 1. 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

1.3 D3.js

In this paper D3.js 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 SVG, CSS and HTML5 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:

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

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

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

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.

```
1 <!DOCTYPE html>
2 <meta charset="utf-8">
3 <style> /* set the CSS */
4
5 body { font: 12px Arial;}
6
7 path {
8     stroke: steelblue;
9     stroke-width: 2;
10    fill: none;
11 }
12
13 .axis path,
14 .axis line {
15     fill: none;
16     stroke: grey;
17     stroke-width: 1;
18     shape-rendering: crispEdges;
19 }
20
21 </style>
22 <body>
23
24 <!-- load the d3.js library -->
25 <script src="http://d3js.org/d3.v3.min.js"></script>
26
27 <script>
28
29 // Set the dimensions of the canvas / graph
30 var margin = {top: 30, right: 20, bottom: 30, left: 50},
31     width = 600 - margin.left - margin.right,
32     height = 270 - margin.top - margin.bottom;
33
34 // Parse the date / time
35 var parseDate = d3.time.format("%d-%b-%y").parse;
36
37 // Set the ranges
38 var x = d3.time.scale().range([0, width]);
39 var y = d3.scale.linear().range([height, 0]);
40
41 // Define the axes
```

8 1. BACKGROUND

```
43 var xAxis = d3.svg.axis().scale(x)
    .orient("bottom").ticks(5);

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

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

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

61 // Get the data
63 d3.csv("data.csv", function(error, data) {
    data.forEach(function(d) {
65        d.date = parseDate(d.date);
        d.close = +d.close;
67    });

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

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

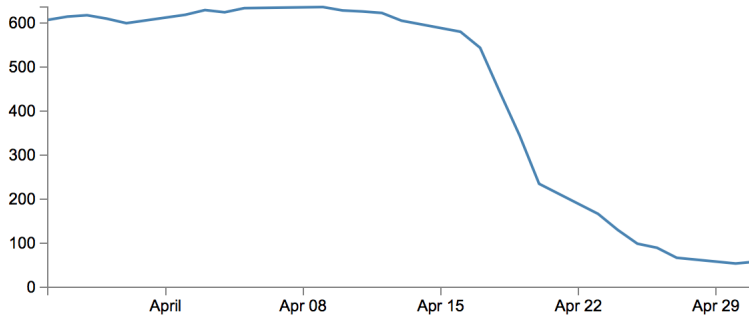
77    // Add the X Axis
79    svg.append("g")
        .attr("class", "x axis")
81        .attr("transform", "translate(0," + height + ")")
        .call(xAxis);

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

89 });

91 </script>
</body>
```


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.

- **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 ..

2.2 Initial research

In section 1.1 we see how the raw format of the NetFlow packets look. 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.

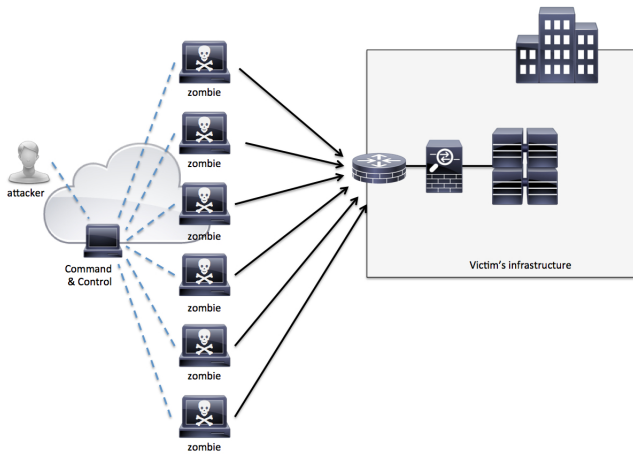
To understand this, experiments will determine how quickly one can distinguish an attack from both the raw format, and the visual representation.

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 up against each other.

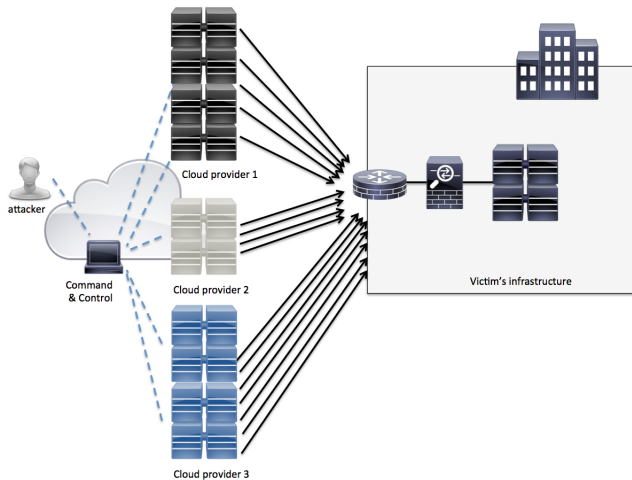
To be able to identify an DDoS attack, one can look at it from two angles. By finding someone whom is attacking, or someone whom is being attacked. In this case we will look at the second scenario. As mentioned earlier simply a peak in flows is not enough grounds to establish an actual attack. First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

2.3 Traits of a DDoS attack

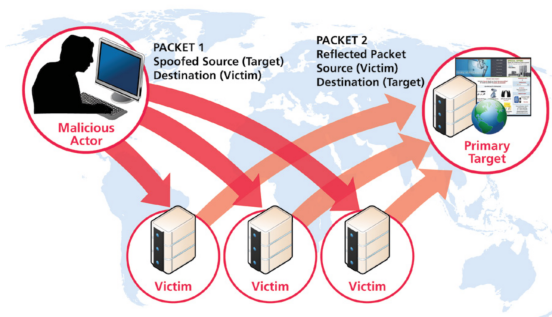
In a Distributed Denial of Service attack there are a large number of hosts performing the attack. In many cases a lot of them are not even aware they are a part on an attack. This is called a botnet, derived from the words robot and network. Using compromised systems, called zombies, gives the attacker 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 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 is becoming more and more popular. DrDoS techniques usually involve multiple victim host machines that unwillingly participate in a DDoS attack on the attackers 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.



2.3.1 Raw NetFlow format

```

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
512 544 576 1024 1536 2048 2560 3072 3584 4096 4608
.000 .001 .256 .000 .010 .000 .000 .000 .000 .000 .000
IP Flow Switching Cache, 4456704 bytes
1885 active, 63651 inactive, 59960004 added
129803821 age 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	Flows /Sec	Packets /Flow	Bytes /Pkt	Packets /Sec	Active(Sec) /Flow	Idle(Sec) /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
Gi0/0 192.168.10.201 Gi0/1 192.168.60.102 06 0984 0050 1
Gi0/0 192.168.11.54 Gi0/1 192.168.60.158 06 0911 0035 3
Gi0/1 192.168.150.60 Gi0/0 10.89.16.226 06 0016 12CA 1
Gi0/0 192.168.10.17 Gi0/1 192.168.60.97 11 0B89 0050 1
Gi0/0 10.88.226.1 Gi0/1 192.168.202.22 11 007B 007B 1
Gi0/0 192.168.12.185 Gi0/1 192.168.60.239 11 0BD7 0050 1
Gi0/0 10.89.16.226 Gi0/1 192.168.150.60 06 12CA 0016 1
router#

```

Figure 2.1: Ten files in the provided files with the most flows

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.

2.4 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 January of 2012 from Trondheim and Oslo NetFlow collectors. This means millions of data entries.

2.4.1 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.

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 2.2.

```
[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 2.2: Ten files in the provided files with the most flows

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.

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


```

[ee@glarse@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 2.3: The smallest files from the provided files

```

[ee@glarse@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 2.4: 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.

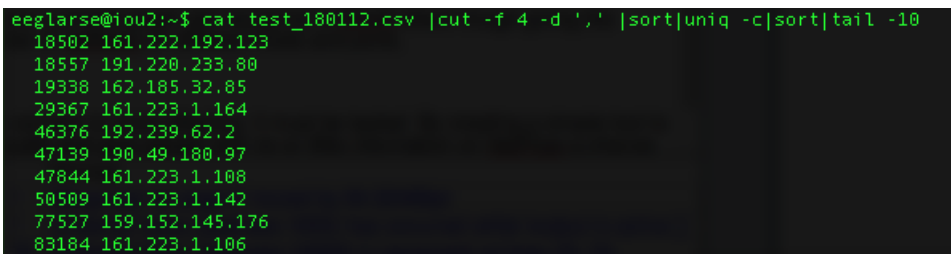
We see that the same IP-address, 192.239.62.2, is here high up as well. Among hundreds of thousands of addresses in the spectrum.

If we were to figure out what application is the reason behind this traffic, one would have to include destination port, and a look at source port might also reveal something.

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.



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

2.4.2 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.

Screenshot

To confirm that such a solution is more effective, it must be tested. By creating a simple tool to convey information visually and give people with no or little information on NetFlow a chance to compare a visual solution to the raw machine readable data, and find out how effective it actually is.

Chapter 3

Challenges

3.1 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.

3.1.1 IP-spectrum

References

- [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.
- [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

Lime inn hvordan det ser ut i kommandolinjen	3
sitere listen	3
sjekke opp med portnr og	17
Screenshot	18