Large Scale Monitoring of BroadBand Internet Infrastructure
Contract No. 004336

D 4.5b: “Second Period's Training Material”

<table>
<thead>
<tr>
<th>Contractual Date of Delivery</th>
<th>June 2006</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actual Date of Delivery</td>
<td>August 2006</td>
</tr>
<tr>
<td>Deliverable Security Class</td>
<td>Public</td>
</tr>
<tr>
<td>Editor</td>
<td>Anna Doxastaki</td>
</tr>
<tr>
<td>Contributors</td>
<td>Evangelos Markatos, Herbert Bos, Arne Oslebo</td>
</tr>
</tbody>
</table>

The LOBSTER Consortium consists of:

<table>
<thead>
<tr>
<th>Organization</th>
<th>Country</th>
</tr>
</thead>
<tbody>
<tr>
<td>FORTH-ICS</td>
<td>Greece</td>
</tr>
<tr>
<td>VU</td>
<td>Netherlands</td>
</tr>
<tr>
<td>CESNET</td>
<td>Czech Republic</td>
</tr>
<tr>
<td>UNINETT</td>
<td>Norway</td>
</tr>
<tr>
<td>ENDACE</td>
<td>United Kingdom</td>
</tr>
<tr>
<td>Alcatel</td>
<td>France</td>
</tr>
<tr>
<td>FORTHnet</td>
<td>Greece</td>
</tr>
<tr>
<td>TNO</td>
<td>Netherlands</td>
</tr>
<tr>
<td>TERENA</td>
<td>Netherlands</td>
</tr>
</tbody>
</table>
Table of Contents

Overview ____________________________________________________________ 3
Tutorial Evaluation ___________________________________________________ 4
List of Participants ___________________________________________________ 6
Annex I - Questionnaire ______________________________________________ 8
Annex II – slides of presentations ______________________________________ 9
Overview

The LOBSTER Tutorial took place on the 15\textsuperscript{th} of May 2006, in conjunction with the TERENA Networking Conference (http://www.terena.nl/events/tnc2006/), in Catania, Italy. The speakers of the Tutorial came from some partners of the project, in particular:

- FORTH - Evangelos Markatos
- VU – Herbert Bos
- UNINETT – Arne Oslebo

The tutorial aimed to introduce to the attendants the principles and practice of passive network traffic monitoring, i.e. monitoring based on full packet capture and inspection. The tutorial also covered topics, such as: applications using MAPI, capturing traffic, and overview of traffic analysis.

The Tutorial comprised four talks:

- “Introduction to passive network monitoring and the packet capture (pcap) library”  
  \textit{Evangelos Markatos, (FORTH)}

- “Implementing passive monitoring applications using MAPI”  
  \textit{Arne Øslebo (UNINETT)}

- “Capturing packets with FFPF: Fairly Fast Packet Filters”  
  \textit{Herbert Bos (VU)}

- “An overview of traffic analysis using NetFlow”  
  \textit{Arne Øslebo (UNINETT)}

These presentations are available on line, through the link:  

They also appear in Annex II of this report, in page 9.
Tutorial Evaluation

A questionnaire (please refer to Annex I in page 8) was disseminated during the Tutorial to the participants, who were asked to provide their feedback in terms of the usefulness of the Tutorial. Figure 1 below shows that 43% of the participants thought that the Tutorial was good, while 32% thought it was very good.

![Figure 1: Perceived usefulness of the Tutorial by the participants (in percentages).](image1.png)

**Evaluation of Usefulness:**
1: not useful
2: a little useful
3: moderately useful
4: good
5: very good

The participants were also asked to evaluate the Tutorial in terms of its organization. As it appears in Figure 2 below, about 46% of the participants thought that the organization of the Tutorial was good, while 43% thought that it was very good.

![Figure 2: Evaluation of the Tutorial’s organization by the participants (in percentages).](image2.png)

**Evaluation of Organization:**
1: very poor
2: poor
3: moderate
4: good
5: very good
Finally, there was a question that involved the statement of any comments the participants could have. One of these comments was that the “developing of PCAP APIs” should be a separate topic from traffic analysis (NetFlow), since these generate interest on different groups of people.

Overall, the comments were in the lines of overall satisfaction.

Therefore, the Tutorial was successful since the majority of the participants considered it to be good in terms of usefulness. There was a lively discussion and questions about the Project and in general the attendants demonstrated their interest in the project.

By taking into account the feedback of the participants, the work of the project will address the ideas expressed and the questions raised.

![Photos from the Tutorial](image)

Figure 3: Photos from the Tutorial. From upper left to bottom right:
  a. Herbert Bos (VU) during his talk
  b. Audience at the tutorial
  c. Prof. Markatos talking with Mr. Wim Jansen, LOBSTER Project Officer
  d. Arne Øslebø (UNINETT) during his talk
List of Participants
The number of registered participants exceeded 50, but the actual participants were about 60, since some of them attended at the last minute.

<table>
<thead>
<tr>
<th></th>
<th>Name</th>
<th>Institution</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Maurizio Aiello</td>
<td>CNR - IEIIT</td>
</tr>
<tr>
<td>2</td>
<td>Lachlan Andrew</td>
<td>Caltech</td>
</tr>
<tr>
<td>3</td>
<td>Marco Bencivenni</td>
<td>INFN-CNAF</td>
</tr>
<tr>
<td>4</td>
<td>Spiros Bolis</td>
<td>University of Athens</td>
</tr>
<tr>
<td>5</td>
<td>Ronald Boontje</td>
<td>Universiteit van Amsterdam</td>
</tr>
<tr>
<td>6</td>
<td>Joe Burns</td>
<td>Queen's University Belfast</td>
</tr>
<tr>
<td>7</td>
<td>Stefano Catani</td>
<td>University of Trieste</td>
</tr>
<tr>
<td>8</td>
<td>Pavel CELEDA</td>
<td>CESNET z. s. p. o.</td>
</tr>
<tr>
<td>9</td>
<td>Massimiliano CICCARELLI</td>
<td>DATAMAT/ESA</td>
</tr>
<tr>
<td>10</td>
<td>Sandra Denasi</td>
<td>INRIM</td>
</tr>
<tr>
<td>11</td>
<td>Freek Dijkstra</td>
<td>Universiteit van Amsterdam</td>
</tr>
<tr>
<td>12</td>
<td>Anna Doxastaki</td>
<td>FORTH-ICS</td>
</tr>
<tr>
<td>13</td>
<td>Giacomo Fazio</td>
<td>INAF IASF</td>
</tr>
<tr>
<td>14</td>
<td>David Foster</td>
<td>CERN</td>
</tr>
<tr>
<td>15</td>
<td>Giorgio Giorgetti</td>
<td>University of Trieste</td>
</tr>
<tr>
<td>16</td>
<td>Lisa Golka</td>
<td>Univ. of Stuttgart</td>
</tr>
<tr>
<td>17</td>
<td>Paola Grosso</td>
<td>UvA</td>
</tr>
<tr>
<td>18</td>
<td>Mike Harron</td>
<td>University of Hertfordshire</td>
</tr>
<tr>
<td>19</td>
<td>Wim Jansen</td>
<td>European Commission DG INFSO</td>
</tr>
<tr>
<td>20</td>
<td>Peter Juul</td>
<td>Uni-C / Danish NREN</td>
</tr>
<tr>
<td>21</td>
<td>Dariusz Klimowicz</td>
<td>TASK</td>
</tr>
<tr>
<td>22</td>
<td>Markus Krieger</td>
<td>Universitaet Wuerzburg</td>
</tr>
<tr>
<td>23</td>
<td>Felix Kugler</td>
<td>SWITCH</td>
</tr>
<tr>
<td>24</td>
<td>Ladislav Lhotka</td>
<td>CESNET</td>
</tr>
<tr>
<td>25</td>
<td>Eric Lopez</td>
<td>RAICES - CLARA</td>
</tr>
<tr>
<td>26</td>
<td>Evangelos Markatos</td>
<td>FORTH-ICS</td>
</tr>
<tr>
<td>27</td>
<td>Jaime Leonardo Martínez Rodríguez</td>
<td>Red Nacional Academica de Tecnologia Avanzada</td>
</tr>
<tr>
<td>28</td>
<td>Maurizio Molina</td>
<td>DANTE</td>
</tr>
<tr>
<td>29</td>
<td>Herman Moons</td>
<td>University of Leuven</td>
</tr>
<tr>
<td>30</td>
<td>Walter Eduardo Munguia Martínez</td>
<td>RAAP-CLARA</td>
</tr>
<tr>
<td>31</td>
<td>Mscislaw Nakonieczny</td>
<td>TASK</td>
</tr>
<tr>
<td>32</td>
<td>Jiri Novotny</td>
<td>CESNET z.s.p.p</td>
</tr>
<tr>
<td>33</td>
<td>Brian O’Hara</td>
<td>University of Dublin, Trinity College</td>
</tr>
<tr>
<td>34</td>
<td>Aida Omerovic</td>
<td>UNINETT FAS</td>
</tr>
<tr>
<td>35</td>
<td>Chris Oornick</td>
<td>KLPD</td>
</tr>
<tr>
<td>36</td>
<td>Jukka Orjärvi</td>
<td>Oulu Polytech</td>
</tr>
<tr>
<td>37</td>
<td>Vasile Florian Pais</td>
<td>National Institute for Laser, Plasma &amp; Radiation Physics</td>
</tr>
<tr>
<td>38</td>
<td>Michalis Polychronakis</td>
<td>FORTH-ICS</td>
</tr>
<tr>
<td>39</td>
<td>Juergen Rauschenbach</td>
<td>DFN-Verein</td>
</tr>
</tbody>
</table>


<table>
<thead>
<tr>
<th></th>
<th>Name</th>
<th>Institution</th>
</tr>
</thead>
<tbody>
<tr>
<td>40</td>
<td>Genevieve Romier</td>
<td>CNRS</td>
</tr>
<tr>
<td>41</td>
<td>Matti Saarinen</td>
<td>University of Helsinki</td>
</tr>
<tr>
<td>42</td>
<td>Piotr Sasiedzki</td>
<td>Silesian University of Technology Computer Centre</td>
</tr>
<tr>
<td>43</td>
<td>Wolfgang Schrimm</td>
<td>University of Heidelberg</td>
</tr>
<tr>
<td>44</td>
<td>Vladimir Smotlacha</td>
<td>CESNET</td>
</tr>
<tr>
<td>45</td>
<td>rogier spoor</td>
<td>SURFnet</td>
</tr>
<tr>
<td>46</td>
<td>Tomislav Stivojevic</td>
<td>CARNet</td>
</tr>
<tr>
<td>47</td>
<td>Fergal Suipeil</td>
<td>University College Dublin</td>
</tr>
<tr>
<td>48</td>
<td>Zbigniew S. Szewczak</td>
<td>Faculty of Math &amp; Comp. Sc. N. Copernicus Univ.</td>
</tr>
<tr>
<td>49</td>
<td>Derek Taylor</td>
<td>University of Hertfordshire</td>
</tr>
<tr>
<td>50</td>
<td>Pedro Veiga</td>
<td>FCCN</td>
</tr>
<tr>
<td>51</td>
<td>Pavle Vuletic</td>
<td>University of Belgrade</td>
</tr>
<tr>
<td>52</td>
<td>Björn Wiberg</td>
<td>Uppsala University</td>
</tr>
</tbody>
</table>
Annex I - Questionnaire

LOBSTER Tutorial on Passive Network Monitoring

Questionnaire:

- Please rate the usefulness of the Tutorial using the scale:
  1: not useful,  2: a little useful  3: moderately useful,  4: good,  5: very good

```
1 □ 2 □ 3 □ 4 □ 5 □
```

- Which talk(s) did you find the most interesting? (Please write their numbers)
  …………………………………………… …………………………………
  …………………………………………… …………………………………

- Which talk(s) did you find the least interesting? (Please write their numbers)
  …………………………………………… …………………………………
  …………………………………………… …………………………………

- Which topics would you like to see presented in a future LOBSTER Tutorial?
  …………………………………………… …………………………………
  …………………………………………… …………………………………

- Please rate the overall organization of the Tutorial using the scale:
  1: very poor,  2: poor,  3: moderate,  4: good,  5: very good

```
1 □ 2 □ 3 □ 4 □ 5 □
```

- Do you have any general comments for the Tutorial overall?
  …………………………………………… …………………………………
  …………………………………………… …………………………………
  …………………………………………… …………………………………
  …………………………………………… …………………………………
  …………………………………………… …………………………………
Annex II – slides of presentations

1. “Introduction to passive network monitoring and the packet capture (pcap) library”  
   Evangelos Markatos, (FORTH)

2. “Implementing passive monitoring applications using MAPI”  
   Arne Øslebø (UNINETT)

3. “Capturing packets with FFPF: Fairly Fast Packet Filters”  
   Herbert Bos (VU)

4. “An overview of traffic analysis using NetFlow”  
   Arne Øslebø (UNINETT)
Evangelos Markatos  
FORTH-ICS  
markatos@ics.forth.gr

http://www.ics.forth.gr/~markatos  
Institute of Computer Science  (ICS)  
Foundation for Research and Technology – Hellas (FORTH)

Evangelos Markatos  info@ist-lobster.org  
FORTH

---

**Roadmap**

- **Motivation**
  - What is network traffic monitoring?
  - Why is it important?
- **Two methodologies for monitoring**
  - **Active** network monitoring
    - Examples
  - **Passive** network monitoring
    - Examples
- pcap
- Introduce the rest of this tutorial
- Summary and Conclusions

---

Focus of this tutorial
Motivation: A GRAND Challenge in Networking

The first GRAND Challenge in Computer Networking is to

“… develop and deploy the technology to make it possible to record a day in the life of the Internet…”

Committee on Research Horizons in Networking
Clark, Lazowska, Patterson, Paxson, Savage, Zegura, ….
2001
The Challenge

- Why is it important?
  - "a data set with typical days for the next 10 years of the Internet might be a treasure chest for networking researchers"
  - Measurement-based GRAND challenges, such as the human genome, have served to
    - Crystallize research issues, and
    - Mobilize research efforts
  - Good network monitoring data are necessary for operational needs
    - Why is my network slow?
    - Which route do my packets follow?
    - Why is a particular flow missing lots of packets?
    - How much peer-to-peer traffic is there?
- Next GRAND Challenge in Networking Research:
  - Monitor a day in the life of the Internet

Evangelos Markatos  info@ist-lobster.org

Network Traffic Monitoring

- So, do Network Traffic Monitoring
  - To get a better understanding on “what’s on the network”
- What can you do with monitoring?
  - Know the state of the Internet and your network
  - Capacity planning
  - Traffic accounting
    - Which application generates most traffic?
  - Understand the performance of individual applications
    - “why is my application so slooooooooow”?
  - Detect Security threats
    - DoS Attacks, Worm outbreaks

Evangelos Markatos  info@ist-lobster.org
**Roadmap**

- **Motivation**
  - What is network traffic monitoring?
  - Why is it important?

- **Two methodologies for monitoring**
  - **Active** network monitoring
    - Examples
  - **Passive** network monitoring
    - Examples

- **pcap**
- Introduce the rest of this tutorial
- Summary and Conclusions

Evangelos Markatos  info@ist-lobster.org

---

**Active Monitoring**

- Inject packets into the network
- Measure their arrival time, loss rate, etc.
- What can you do with it?
  - Measure **delay** (one-way / two-way)
  - Measure **bottleneck link bandwidth**
  - Find **network topology**
  - **What’s up?** (in my network)
    - Which nodes are up and running?
Evangelos Markatos  info@ist-lobster.org

Ping: is the receiver host alive?

```plaintext
%> ping www.ist-lobster.org
PING www.ist-lobster.org (192.87.30.11): 56 data bytes
64 bytes from 192.87.30.11: icmp_seq=0 ttl=49 time=308.7 ms
64 bytes from 192.87.30.11: icmp_seq=1 ttl=49 time=307.6 ms
64 bytes from 192.87.30.11: icmp_seq=2 ttl=49 time=244.4 ms

--- www.ist-lobster.org ping statistics ---
3 packets transmitted, 3 packets received, 0% packet loss
round-trip min/avg/max = 244.4/286.9/308.7 ms
```

Evangelos Markatos  info@ist-lobster.org

Traceroute: find all intermediate routers between a source and a destination computer

```plaintext
%> traceroute www.ist-lobster.org (from Crete)
traceroute to www.ist-lobster.org (192.87.30.11), 30 hops max, 40 byte packets
1 147.52.17.1 (147.52.17.1) 1.050 ms 0.690 ms 0.592 ms
2 olympos.e43.lanh.uoc.gr (147.52.12.1) 1.626 ms 1.033 ms 0.840 ms
3 heraklio-uch-ATM.grnet.gr (194.177.209.141) 129.528 ms 112.644 ms 123.465 ms
4 heraklio2-to-heraklio.backbone.grnet.gr (194.177.209.77) 124.791 ms 116.749 ms 119.965 ms
5 Syros-to-Heraklio2.backbone.grnet.gr (195.251.27.81) 136.089 ms 104.469 ms 81.116 ms
6 athens3-to-Syros.backbone.grnet.gr (195.251.27.10) 72.664 ms 62.814 ms 67.341 ms
7 grnet.gr1.gr.geant.net (62.40.103.57) 81.392 ms 102.067 ms 79.488 ms
8 gr.de2.de.geant.net (62.40.96.82) 145.367 ms 150.166 ms 142.117 ms
9 de2-2.de1.de.geant.net (62.40.96.54) 145.367 ms 150.166 ms 142.117 ms
10 de.nl1.nl.geant.net (62.40.103.98) 184.078 ms 158.921 ms 160.933 ms
11 surfnet-gw.nl1.nl.geant.net (62.40.103.98) 184.078 ms 158.921 ms 160.933 ms
12 PO11-0-GR1.Amsterdam1.surf.net (145.145.166.33) 145.367 ms 150.166 ms 142.117 ms
13 PO0-0.AR5.Amsterdam1.surf.net (145.145.162.2) 163.605 ms 144.161 ms 177.526 ms
14 145.145.18.46 (145.145.18.46) 178.350 ms 175.365 ms 166.334 ms
15 * 145.145.18.46 (145.145.18.46) 178.079 ms IX
16 * 145.145.18.46 (145.145.18.46) 171.881 ms IX *
17 145.145.18.46 (145.145.18.46) 192.753 ms IX * 180.104 ms IX
```

Evangelos Markatos  info@ist-lobster.org
Roadmap

- Motivation
  - What is network traffic monitoring?
  - Why is it important?
- Two methodologies for monitoring
  - Active network monitoring
    - Examples
  - Passive network monitoring
    - Examples
- pcap
- Introduce the rest of this tutorial
- Summary and Conclusions
Passive Traffic Monitoring

• What is it?
  • Non-intrusive traffic monitoring
    – Much like a **telescope**
    – **Does not inject packets** in the network
  • It **passively captures** information from passing packets such as
    – High-level network flows (CISCO Netflow)
    – Network packet headers (NLANR)
    – Entire network packets (incl. payload)
      • if allowed
      • maybe stripped/anonymized (to be shared with a broader audience)

Passive Monitoring: What can it be used for? Performance

• Traffic Categorization/Accounting:
  – What % of my traffic is due to email?
  – Which subnet generates most outgoing traffic?
• Bandwidth Estimation
  – What % of my bandwidth is available now?
  – What % of my bandwidth is being used?
• Study trends:
  – How does the application mix in the traffic changes with time?
    • ftp in the 80’s, www in the 90’s, p2p in the 00’s
  – How does peer-to-peer traffic changes with time?
• Performance Debugging of individual applications
  – Why is **my** application so sloooow?
Example Application: Traffic Categorization

Passive Monitoring:
What can it be used for? Security

- Intrusion Detection
  - Are any of my computers compromised?
  - Do they participate in a botnet?
  - Is there any attacker trying to intrude into my network?
- Large-scale Attack Detection – Detection of Epidemics
  - DoS Attack detection
    - e.g. Detect sharp increases in TCP/SYN packets
  - Zero-day worm detection
    - e.g. Detect lots of identical packets, never seen before, from several sources to several destinations
    - e.g. Detect worm characteristics
      - such as NOP sleds: long sequences of executable code
- Network Telescopes
  - They monitor unused IP addresses (“dark matter”)
  - Ordinarily, unused IP addresses should not receive traffic
  - Observe victims of DoS attacks
    - “back-scatter” traffic, “background radiation”
  - Observe infected hosts
  - Port scans

Evangelos Markatos  info@ist-lobster.org
What equipment is needed?

- Equipment varies from
  - low-end (for low-speed networks) to
  - Sophisticated equipment (for high-speed networks)
- Low-end passive monitors (100Mbps – 1Gbps)
  - An ordinary PC
  - An ordinary network Interface (i.e. an Ethernet card)
    • put in promiscuous mode
  - Mirror all packets from a router to a port connected to
    the above PC

High-end equipment

- High-end passive monitors (1Gbps – 10Gbps)
  - High-end computer
  - Specialized network interface
    • DAG Cards (Endace)
    • Combo cards (SCAMPI project)
    • Hardware-based filtering capabilities
      • Process packets at line speeds
Software?

- **pcap**: packet capture library from Berkeley
- **MAPI**: Monitoring API (Application Programming Interface)
  - [http://www.ist-lobster.org/downloads](http://www.ist-lobster.org/downloads)
  - Developed within the IST SCAMPI project
    - co-funded by EU
- **Net-flow-related tools**
  - Graphical interfaces

Roadmap

- **Motivation**
  - What is network traffic monitoring?
  - Why is it important?
- **Two methodologies for monitoring**
  - **Active** network monitoring
    - Examples
  - **Passive** network monitoring
    - Examples
- **pcap**
  
  Focus of this tutorial

- Introduce the rest of this tutorial
- Summary and Conclusions
- Packet capture Library
- Developed to capture packets
  - And dump them to disk
- Basic Steps:
  - Open a network interface
  - Get one packet at a time
  - Print it – dump it to the disk

Example: Grab one packet:

```c
main() {
  /* open a network interface */
  descr = pcap_open_live(dev, BUFSIZ, 0, 1, errbuf);
  /* Get next packet */
  packet = pcap_next(descr, &hdr);
  /* print its length */
  printf("Grabbed packet of length %d\n", hdr.len);
}
```
pcap (several packets)

- Example: Grab several packets:

```c
main() {
    /* open a network interface */
    descr = pcap_open_live(dev,BUFSIZ,0, 1,errbuf);
    while (1) {
        /* Grab packets for ever */
        packet = pcap_next(descr,&hdr);
        /* print its length */
        printf("Grabbed packet of length %d\n", hdr.len);
    }
}
```

Evangelos Markatos  info@ist-lobster.org

pcap (several packets)

- Example: Grab several packets:

```c
main() {
    /* open a network interface */
    descr = pcap_open_live(dev,BUFSIZ,0, 1,errbuf);
    pcap_loop (descr,atoi(argv[1]),my_callback, NULL);

    /*callback function */
    void my_callback (u_char *u,const struct pcap_pkthdr* pkthdr,
                     const u_char* packet)
    {
        /* just count the number of packets */
        static int count = 1;
        printf("%d, \n",count);
        count++;
    }

    Grab 100 packets:
    %>a.out 100
```

Evangelos Markatos  info@ist-lobster.org
Example: Suppose that you want to capture only packets destined to your web servers (destination port 80):

```c
main() {
    descr = pcap_open_live(dev,BUFSIZ,0,1,errbuf);
    /* install a filter */
    pcap_compile(descr,&fp,"dst port 80",0,netp)
    pcap_setfilter(descr,&fp)

    pcap_loop (descr,atoi(argv[1]),my_callback, NULL);
}
/*callback function */
void my_callback (u_char *u,const struct pcap_pkthdr* pkthdr, const
    u_char* packet) {
    static int count = 1;
    printf("%d, 
",count);
    count++;
}
```

pcap Limitations

- Excellent for capturing and dumping packets by a single application
- Several applications
  - High overhead – copies all packets to all applications
- Limited functionality
  - No string searching, no packet counting, etc.
Interested in pcap?

- [http://www.tcpdump.org/](http://www.tcpdump.org/)
- [http://www.winpcap.org/](http://www.winpcap.org/)

The rest of the tutorial:

**Part II: Flow-level monitoring**

- **MAPI**
  - Daemon-based packet monitoring
  - Rich functionality (string searching, counters, anonymization, etc.)
- **FFPF (Fairly Fast Packet Filters)**
- **Flow-level passive monitoring**
  - General introduction
    - NetFlow and IPFIX
  - NetFlow-based applications
    - Stager (UNINETT)
    - NERD (TNO)
- **Ruler Anonymization Language**
“Monitor a Day in the Life of the Internet”
Committee on Res. Horizons in Networking

- Traffic Monitoring help us understand what’s on the network
- Passive Network Traffic Monitoring applications for:
  - Performance
    - traffic accounting/categorization
    - Performance debugging
  - Security
    - DoS attack detection,
    - Internet epidemics
    - Intrusion Detection

Evangelos Markatos  info@ist-lobster.org
FORTH
Implementing passive monitoring applications using MAPI

Arne Øslebø
UNINETT
Arne.Oslebo@uninett.no

Outline

• General overview
  – What is MAPI?
  – Example
• Design
• Using MAPI
  – MAPI calls
  – MAPI functions
• Using DiMAPI
• Extending MAPI
**Monitoring Application Programming Interface**

**Design goals:**
- Make it quick and easy to implement new monitoring applications
- Low overhead
- Support for multiple concurrent users and applications
  - Optional support for strong authentication.
- Global optimization
  - Optimize processing of packets based on all applications from all users.
- Transparent support for different hardware adapters
  - NIC, DAG, COMBO6
- Easy to extend

```c
1: fd=mapi_create_flow("/dev/dag0");
2: mapi_apply_function(fd,"BPF_FILTER","src port 1234");
3: ctr_id1=mapi_apply_function(fd,"PKT_COUNTER");
4: mapi_apply_function(fd,"STR_SEARCH","pattern",100,300);
5: ctr_id2=mapi_apply_function(fd,"PKT_COUNTER");
6: mapi_apply_function(fd,"TO_FILE",MFF_PCAP,"worm.trace",0);
7: mapi_connect(fd);
8: 
9: while(1) {
10: ctr_val1=mapi_read_results(fd,ctr_id1);
11: ctr_val2=mapi_read_results(fd,ctr_id2);
12: 
13: printf("BPF match: %llu String match: %llu\n",
14: *ctr_val1,*ctr_val2);
15: sleep(10);
16: }
```
Architecture

http://www.ist-lobster.org

MAPI calls

- mapi_create_flow
- mapi_create_offline_device
- mapi_start_offline_device
- mapi_delete_offline_device
- mapi_apply_function
- mapi_connect
- mapi_read_results
- mapi_get_next_pkt
- mapi_loop
- mapi_read_error
- mapi_close_flow

http://www.ist-lobster.org
• Collection of MAPI functions that can be applied to a MAPI flow
  – BPF, string search, packet counter, jitter, statistical function etc.
• Functions can:
  – filter packets
  – transform packets
  – return or process information from packets
  – read and process results from other functions
  – etc.
• Standard library: 24 functions

---

• Adds or subtracts values from two other MAPI functions
• Arguments:
  – fd and fid for left function: int
  – fd and fid for right function: int
  – type:int (BINOP_ADD,BINOP_SUB)
• Return value:
  – unsigned long long
• Example:
  – ts1=mapi_apply_function(fd,"PKT_INFO",PKT_TS);
  – ts2=mapi_apply_function(fd2,"PKT_INFO",PKT_TS);
  – binop=mapi_apply_function(fd,"BINOP",
    fd2,ts2,fd1,ts1,BINOP_SUB);
BPF_FILTER

- Filters the packets according to a BPF filter expression
- Arguments
  - bpf_filter : char*
    - BPF filter
- Return value:
- Usage:
  - mapi_apply_function(fd,"BPF_FILTER", "port 80");

BUCKET

- Returns results from other MAPI functions in precise periodical intervals
- Arguments
  - interval : char*
  - fd : int
  - function : int
- Return value:
  - void *
- Usage:
  - count=mapi_apply_function(fd,"PKT_COUNTER");
  - mapi_apply_function(fd,"BUCKET", fd,count,"1s");
• Counts number of bytes in a flow
• Arguments
• Return value:
  – unsigned long long
• Usage:
  – mapi_apply_function(fd,"BYTE_COUNTER");

• Processes the packets of a flow by performing IP defragmentation and TCP stream reassembly
• Arguments
  – threshold : int
    • minimum stream size, default 32Kb
  – timeout : int
    • timeout value for further processing since the first arrival of a packet. Default 30 seconds
• Return value:
• Usage:
  – mapi_apply_function(fd,"COOKING",-1,-1);
DIST

- Returns an array that represents the distribution of results from another MAPI function.
- Arguments
  - `fd` : int
  - `function` : int
  - `min` : char*
  - `max` : char*
  - `interval` : char*
- Return value:
  - `unsigned long long[]`
- Usage:
  - `gap = mapi_apply_function(fd,"GAP");`
  - `mapi_apply_function(fd,"DIST",fd,gap,"1ms","2ms","1us");`

GAP

- Returns the time delay between two consecutive packets in a flow
- Arguments
- Return value:
  - `unsigned long long`
- Usage:
  - `mapi_apply_function(fd,"GAP");`
HASH

- Computes the additive hash function over the packets of a network flow.
- Arguments
  - layer : int
    - link, network, IP or application layer
- Return value:
  - unsigned int
- Usage:
  - mapi_apply_function(fd,"HASH");

HASHSAMP

- Hashing based sampling
- Only processes TCP and UDP packets
- Calculates hash on src/dst IP and src/dst port
- Arguments
  - range : int
  - keep : int
- Return value:
  - struct sample {
      unsigned int source_ip;
      unsigned int dest_ip;
      ...
  }
- Usage:
  - mapi_apply_function(fd,"HASHSAMP",100,1);
PKT_COUNTER

- Counts number of packets in a flow
- Arguments
- Return value:
  - unsigned long long
- Example:
  - mapi_apply_function(fd,"PKT_COUNTER");

PKT_INFO

- Returns information about a packet
- Arguments
  - info : int
    - PKT_TS – packet timestamp
    - PKT_SIZE – packet size
- Return value:
  - unsigned long long
- Usage:
  - mapi_apply_function(fd,"PKT_INFO",PKT_SIZE);
PROTINFO

- Returns a specific protocol field
- Arguments
  - info : int
    - PI_TCPSEQ – TCP Sequence number
    - PI_TCPACK – TCP Ack number
    - Will be extended to support all fields in IP, TCP and UDP
- Return value:
  - unsigned long long
- Usage:
  - mapi_apply_function(fd,"PROTINFO",PI_TCPSEQ);

REGEXP

- Regular expression pattern matching
- Arguments
  - str : char*
- Return value:
- Example:
  - mapi_apply_function(fd,"REGEXP","[Aa]");
RES2FILE

• Stores results from other MAPI function to a file

• Arguments
  – functions : char*
    • list of functions to read results from
    • "<fid>@<fd>,<fid2>@<fd2>"
  – format : char*
    • R2F_RAW, R2F_UULLSTR, R2F_UULLSEC, R2F_STATS
  – header : char*
    • string to store in the beginning of the file
  – filename : char*
  – interval : char*
    • "-1" for always, "0" for once when flow closes, "1s", "1.2ms" etc.

RES2FILE(2)

• Usage:
  – count=mapi_apply_function(fd,"PKT_COUNT);
  – gap=mapi_apply_function(fd,"GAP");
  – sprintf(fids,"%d@%d,%d@%d",count,fd,gap,fd);
  – sprintf(format,"%d,%d",R2F_ULL,R2F_ULLSEC);
  – mapi_apply_function(fd,"RES2FILE",fids,format,
    "Count Gap","test.res","-1")
SAMPLE

- Performs sampling
- Arguments
  - `value : int`
    - for DETERMINISTIC it specifies a sampling interval of 1 out of `value` packets. For probabilistic it specifies the probability in %.
  - `mode : int`
    - DETERMINISTIC or PROBABILISTIC
- Return value:
  - `unsigned long long`
- Usage:
  - `mapi_apply_function(fd,"SAMPLE",100,
    DETERMINISTIC);`

STARTSTOP

- Starts and/or stops measurements at a specific time
- Arguments
  - `start : char*`
  - `stop : char*`
    - So far only relative time from the first captured packet is supported. Will be extended to support absolute time.
    - “0”, “1s”, “10ms”
- Return value:
- Example:
  - `mapi_apply_function(fd,"STARTSTOP","10s","0");`
STAT

- Returns statistical information about results from other MAPI functions
- Arguments
  - fd : int, fid : int
  - skip : int - number of packets to skip before reading first result
- Return value:
  - struct stats {
    unsigned long long count;
    long double sum;
    long double sum2;
    double max;
    double min
  }
- Usage:
  - gap=mapi_apply_function(fd,"GAP");
  - mapi_apply_function(fd,"STAT",fd,gap,1);

STR_SEARCH

- Search the payload of a packet for a specific pattern
- Arguments
  - pattern : char*
    - pattern to search for. Can contain non-printable characters or binary data using Snort syntax.
  - offset : int
    - offset into packet for starting the search
  - depth : int
    - maximum depth of the search
- Return value:
- Usage:
  - mapi_apply_function(fd, "STR_SEARCH", "ab|63 64|",0,1500);
• Signals when a certain threshold is reached.
• Arguments
  – type : int
    • CHAR, INT, ULL
  – fd : int
  – fid : int
  – boundary : int
    • EQ, GT, LT, EQ_D, GT_D, LT_D
  – threshold : unsigned long long
  – timeout : int
  – divider : int
  – count : int
• Return value:

mapi_apply_function(fd, “TOP”, 5, TOPX_TCP, TOPX_TCP_DSTPORT);

• Returns top X values of a field
• Arguments
  – x : int
  – protocol : int
  – field : int
• Return value:
  – unsigned int[]
• Usage:
  – mapi_apply_function(fd,”TOP”, 5, TOPX_TCP, TOPX_TCP_DSTPORT);
TO_BUFFER

- Stores packets of a flow to a buffer so that they can be read using mapi_get_next_packet
- Arguments
- Return value:
  - struct mapipkt {
    unsigned long long ts;
    unsigned short ifindex;
    unsigned caplen;
    unsigned wlen;
    unsigned char pkt;
  }
- Usage:
  - mapi_apply_function(fd,"TO_BUFFER");

TO_FILE

- Stores packets to a file
- Arguments
  - format : int
    - MFF_PCAP, MFF_ERF, MFF_RAW
  - file_name : char*
  - count : unsigned long long
    - number of packets to store. 0 means all packets will be stored.
- Return value:
- Usage:
  - mapi_apply_function(fd,"TO_FILE",MFF_ERF, 1000);
Other libraries

- DAG library
  - Store packets to file using DAG ERF format
  - Filter on interface
- Anonymization library
  - Versatile packet anonymization functions
- IPFIX library
  - Export Netflow v5, v9 or IPFIX flow records
- Protocol tracker library
  - Detect and track protocols that uses dynamic ports
    - P2P, FTP etc.

Compile MAPI

- http://www.ist-lobster.org/downloads/
- Makefile.in
  - DEBUG – turns on debug messages
  - WITH_DAG – adds support for DAG cards
  - WITH_COMBO6 – adds support for COMBO6 cards
  - WITH_ADMISSION_CONTROL – add support for admission control
  - WITH_FUNCTION_STATS – turns on counters for all functions to show how many packets where processed by each function
  - WITH_PRIORITIES – turns on support for flow priorities
  - WITH_MODIFY_PKTS – turns on support for functions that modifies packets.
  - WITH_ANONYMIZATION – compile anonymization library
  - WITH_TRACKING – compile tracking library
  - WITH_IPFIX – compile IPFIX library
Edit mapi.conf and test

http://www.ist-lobster.org

libpath=.
libs=mapistdflib.so:dagflib.so
drvpath=.

[driver]
device=eth0
driver=mapiincdrv.so

[driver]
device=/dev/dag0
driver=mapiaggedrv.so
description=DAG 4.3GE
alias=trd-ntnu

[format]
format=MFF_DAG_ERF
driver=mapiagedrv.so

[format]
format=MFF_PCAP
driver=mapiincdrv.so

info@ist-lobster.org
Arne Østebye, UNINETT

Example MAPI application

http://www.ist-lobster.org

- Implement an application that can measure some qualities of a TCP stream
- We want to look at the following attributes:
  - Packets per second
  - Bit per second
  - Jitter
  - TCP congestion window
- Results should be written to a file for further study.
- Syntax:
  - tcpanalyze <filename> <src_ip> <src_port> <dst_ip> <dst_port>

info@ist-lobster.org
Arne Østebye, UNINETT
Relevant MAPI functions

- PKT_COUNTER
  - To calculate packets per second
- BYTE_COUNTER
  - To calculate bit per second
- GAP
  - To measure jitter
- PROTINFO
  - TCP sequence and ack numbers
- BINOP
  - To calculate TCP congestion window
- STAT
  - Calculate statistical values
- RES2FILE
  - To store results to file

tcpanalyze.c

- Create a new subdirectory
  - mkdir tcpanalyze
  - cd tcpanalyze
- Create file tcpanalyze.c
- Include necessary headers:

```c
#include <stdlib.h>
#include <stdio.h>
#include "mapi.h"
#include "stdlib/pktinfo.h"
#include "stdlib/res2file.h"
#include "stdlib/protinfo.h"
```
int main(int argc, char *argv[]) {
    char *file=argv[0];

    int snd=mapi_create_flow("eth0");
    if(snd==-1) {
        printf("Could not create flow\n");
        exit(-1);
    }
    int rec=mapi_create_flow("eth0");

    char filter[1024];
    snprintf(filter,1024,\"tcp and src host %s and src port %s and
dst host %s and dst port %s\",
            argv[1],argv[2],argv[3],argv[4]);
    mapi_apply_function(snd,"BPF_FILTER",filter);

    snprintf(filter,1024,\"tcp and src host %s and src port %s and
dst host %s and dst port %s\",
            argv[3],argv[4],argv[1],argv[2]);
    mapi_apply_function(rec,"BPF_FILTER",filter);
**Makefile**

SOURCES=\$(wildcard *.c)
TARGETS=\$(SOURCES:.c=)

CFLAGS=-g -O2 \$(C_WARNINGS) -DDEBUG -I..
LDFLAGS= ../mapi.so -lpcap -lpthread

all: \$(TARGETS)

\%:\% .mapi.so ../mapi.h
	\$(CC) \$(CFLAGS) \( \$(LDflags) \)

clean:
	rm -rf \$(TARGETS)

---

**DiMAPI**

- What do you do if you have multiple probes and want to measure the TCP quality at all of them?
- Answer: use DiMAPI!
- DiMAPI is the distributed version of MAPI
- The SCOPE abstraction:
  - SCOPE is a set of lines (interfaces) to monitor
  - `mapi_create_flow("host1:eth2, host2:/dev/dag0, host3:eth1")`;
  - `mapi_read_result` returns an array of results
- Fully compatible with MAPI
  - All previous functions work as usual
Implementing new MAPI functions

- All code in one single source file
- Header file only needed if function returns complex data.
- Script automatically creates source file for libraries
- MAPI function interfaces:
  - instance
  - init
  - process
  - get_result
  - reset
  - cleanup
  - client_init
  - client_get_result
  - client_cleanup

info@ist-lobster.org  Arne Østebo, UNINETT  41

typedef struct mapidflib_function_def {
    char* libname;
    char* name;
    char* descr;
    char* argdescr;
    char* devtype;
    mapi_result_method_t restype;
    int shm_size;
    short modifies_pkts;
    mapidflib_optimize_t optimize;
    int (*instance)(....);
    int (*init)(....);
    int (*process)(....);
    int (*get_result)(....);
    int (*change_args)(....);
    int (*reset)(....);
    int (*cleanup)(....);
    int (*client_init)(....);
    int (*client_read_result)(....);
    int (*client_cleanup)(....);
} mapidflib_function_def_t;
static mapidflib_function_def_t finfo={
    /* libname
    "PKT_COUNTER", // name
    "Counts number of packets\n    Return value: unsigned long long",
    // descr
    "", // argdescr
    MAPI_DEVICE_ALL, // devtype
    MAPIRES_SHM, // Use shared memory to return results
    sizeof(unsigned long), // shm size
    0, // modifies_pkts
    MAPIOPT NONE, // global optimization
    NULL, // instance
    NULL, // init
    pktc_process,
    NULL, // get_result,
    pktc_reset,
    NULL, // cleanup
    NULL, // client_init
    NULL, // client_read_result
    NULL // client_cleanup
};

static int pktc_process(
    mapidflib_function_instance_t *instance,
    const unsigned char* dev_pkt,
    const unsigned char* link_pkt,
    mapi_pkthdr_t* pkt_head)
{
    (*(unsigned long long*)instance->result.data)++;
    return 1;
}

static int pktc_reset(
    mapidflib_function_instance_t *instance)
{
    (*(unsigned long long*)instance->result.data)=0;
    return 0;
}
Current status of MAPI

- Still under development
- Stable if used as intended
- Can segfault if wrong arguments are passed
- Working hard on making it more robust
- Available from:
  - http://www.ist-lobster.org/downloads/

http://www.ist-lobster.org
Streamline
[tutorial]

Willem de Bruijn
Herbert Bos
Vrije Universiteit Amsterdam

Goals
Goals
- Major:
  - faster
  - more flexible
  traffic processing
- Minor:
  - new networking subsystem
  - use new hardware

Difficulties
- hardware:
  - widening memory gap
  - fast photons, slow silicon
- software/OS:
  - copies
  - context switching
  - memory allocation
- demand:
  - need for expensive processing (like IDS)
Goals (derived)

- cater to
  - individual packets
  - streams
- domain
  - monitoring
  - transmission
- exploit hardware
- retain what is useful

What are we building?

flows
different barriers
Streamline design

- single framework for pkt processing that
  - uses all levels in the processing hierarchy
  - is language neutral
  - offers advanced processing in NIC
  - supports stateless and stateful filters
  - is backward compatible with pcap (while also supporting much more powerful packet languages)
  - helps users to build complex monitoring applications by ‘clicking components together’

Streamline design

- specify processing in terms of a DAG
- Streamline handles instantiation
  - for each function in DAG, Streamline investigates potential placements
  - collective requirements determine definite placement
- Heuristic: lower is better
Buffers

- PacketBuf
  - circular buffer with N slots
  - e.g., large enough to hold packet

- IndexBuf
  - circular buffer with N slots
  - pointers to packets in PacketBuf
Buffers

- PacketBuf
  - circular buffer with N slots
  - e.g., large enough to hold packet
- IndexBuf
  - circular buffer with N slots
  - pointers to packets in PacketBuf
Buffer management

➔ what to do if writer catches up with slowest reader?

- **slow reader preference**
  - drop new packets
    (traditional way of dealing with this)
  - overall speed set by slowest reader

- **fast reader preference**
  - overwrite existing packets
  - application responsible for keeping up
    - check whether packets have been overwritten
    - different drop rates for different apps

---

Basic abstraction: flow
flow: "a stream of data that matches arbitrary user criteria"
How to use it?

- we build complex applications out of simple components
- different types of use
  - administrators
    - CLI/GUI: click applications together
    - program filters/functions in high-level languages
  - programmers
    - write code against the API
    - write new functions to be used by Streamline

Part I: the administrator
Extensible

- modular framework
- language agnostic
- plug-in filters

CLI language

- flowgraph language
  - boolean constructs
    - \( a \& b \)
    - \( a | b \)
    - \( !a \)
  - stream:
    - \( a \rightarrow b \)
    - \( a -80 \rightarrow b \)
  - grouping:
    - \( a \rightarrow (b | c) \)
Example expressions

- **trivial:**
  \[(\text{generator,expression="1:0"}) > (\text{accept,export,name=mybuffer})\]

\[(\text{pcapin,expression=infectedhttp.pcap}) > (\text{packetcount}) > (\text{sampler,expression=50}) > (\text{packetcount})\]

- **cleaning streams:**
  \[(\text{pcapin,expression="infectedhttp.pcap"}) > (\text{tcp}) > (\text{tcpreassemble}) > (\text{ips_prospector}) > (\text{outfiles,expression=cleantcp})\]

- **classification:**
  \[(\text{pcapin,expression="infectedhttp.pcap"}) > (\text{sourceport}) > (\text{tcp}) > (\text{tcpreassemble}) > (\text{inspect})\]

- **transcoding:**
  \[(\text{ipv4_receive,expression="t 127.0.0.1 5050"}) > (\text{rot13}) > (\text{ipv4_transmit,expression="t 127.0.0.1 5051"})\]

Example 1: GUI for creating flowgraphs
Example 1: GUI for creating flowgraphs

As you can see in the diagram, the GUI is designed to create flowgraphs. The interface includes options for various devices and processes, such as Accept, Demo, and Device, which are likely used to control and configure the flowgraph. The flowgraph itself shows connections between different nodes, indicating the flow of data or processes through the system.

Features like hovering over nodes to display configuration and manual bindings enhance the user experience, making it easier to understand and manage the complex interactions within the flowgraph.
Example 1: GUI for creating flowgraphs

mapping on hardware

Streamline is responsible for mapping the flowgraph on the underlying hardware.
Copying

Three flavours of packet processing

- **Regular**
  - copy only when needed
  - may be slow depending on access pattern

- **Zero copy**
  - *never copy*
  - may be slow depending on access pattern

- **Copy once**
  - copy always

Part II: the savvy administrator
Example 2: FPL-3

- new pkt processing language: FPL-2
  - for IXP, kernel and userspace
  - simple, efficient and flexible
  - simple example: filter all webtraffic
    \[
    \textbf{IF} \ (\text{PKT.IP\_PROTO} == \text{PROTO\_TCP}) \ \\land \ (\text{PKT.TCP\_PORT} == 80)) \ \textbf{THEN} \ \textbf{RETURN} \ 1; \]

- more complex example: count pkts in all TCP flows
  \[
  \textbf{IF} \ (\text{PKT.IP\_PROTO} == \text{PROTO\_TCP}) \ \textbf{THEN} \\
  \quad \text{R[0]} = \text{Hash[ 14, 12, 1024];} \\
  \quad \text{M[ R[0] ]++;} \\
  \textbf{FI} \]

FPL-3

- all common arithmetic and bitwise operations
- all common logical ops
- all common integer types
  - for packet
  - for buffer (useful for keeping state!)
- statements
  - Hash
  - External functions
    - to call hardware implementations
    - to call fast C implementations
  - If … then … else
  - For … break; … Fof
  - Return
Example application: dynamic ports

1. // R[0] stores no. of dynports found (initially 0)
2. IF (PKT.IP_PROTO==PROTO_TCP) THEN
3. IF (PKT.TCP_DPORT==554) THEN
4. M[R[0]]=EXTERN("GetDynTCPDPortFromRTSP",0,0);
5. R[0]++;
6. ELSE // compare pkt's dst port to all ports in array – if match, return pkt
7. FOR (R[1]=0; R[1] < R[0]; R[1]++)
8. IF (PKT.TCP_DPORT == M[R[1]]) THEN
9. RETURN TRUE;
10. FI
11. ROF
12. FI
12. RETURN FALSE;

Other example: Ruler

- regular expression matching and rewriting language
  ➔ see the Lobster workshop tomorrow
trivial example

```c
int main (int argc, char** argv)
{
    int fd, bd, len;
    char *pkt;

    // open FFPF and start processing
    streamline_init();
    fd = streamline_request_insert("(generator, expression="1:0") > (packetcount, export, name=mybuf)";

    // access the data
    bd = open_stream("mybuf", O_RDONLY | OF_INDIRECT);  // extended, but posix compatible API
    while (1) {
        sleep(1);
        len = read_stream(bd, &pkt, 1500);  // direct data access
        printf("read a packet of %d bytes\n", len);
    }
    close_stream(bd);

    // close FFPF
    streamline_request_remove(fd);
    streamline_exit();
    return 0;
}
```
Concluding remarks

- achieved: fast flexible packet processing
- minimising copying/context switching
- cater to different users
- exploit advanced hardware
- new languages

more information

http://ffpf.sourceforge.net/
An overview of traffic analysis using NetFlow

Arne Øslebø
UNINETT
Arne.Oslebo@uninett.no

Outline

- What is Netflow?
- Available tools
- Collecting
- Processing
  - Detailed analysis
    - security incidents
  - Long term statistics
    - network trends
    - accounting
- Presentation
  - Stager
What is NetFlow?

• Cisco technology
  – 1996
• IPFIX definition:
  • A set of IP packets passing an observation point in a network during a certain time interval. All packets belonging to a particular flow have a set of common properties.
• Flow Key
  • Each of the properties that are used for defining a flow
• Flow record
  • Measured properties of a flow

NetFlow v5

• Flow key:
  – Source IP address
  – Destination IP address
  – Source port
  – Destination port
  – Layer 3 protocol type
  – TOS
  – Input interface
• Flow record
  – Source and destination IP address
  – Next hop router's IP address
  – Input and output interface index
  – Packets and bytes in the flow
  – sysUptime at start and end of flow
  – TCP/UDP source and destination port number
  – Type of service
  – TCP flags
  – IP protocol
  – Source and destination AS number
  – Source and destination address prefix mask bits
Netflow tools

- flow-tools
  - Stager
  - FlowViewer
  - FlowScan
  - CUFlow
- nfdump
  - nfsen
  - Stager
- flowd
- flamingo
- JKFlow
- http://www.switch.ch/tf-tant/floma/software.html

NetFlow collector

- For example: flow-capture or nfcapd
- Files rotated every \( n \) minutes
- Script can be started when file is rotated
- flow-capture supports gzip
  - 2:1 compression ratio
- UNINETT collects:
  - 27 routers, 207 interfaces
  - sampled netflow – 1:100 sampling rate
  - >30GB every day
- flow-tools
  - flow-cat, flow-filter, flow-nfilter and flow-stat
- nfdump
- src/dst IP, src/dst AS, src/dst port, protocol etc.

```
flow-cat ft-v05.2006-05-12.08* | flow-filter -il |
flow-stat -f12 -S2
```

<table>
<thead>
<tr>
<th>#</th>
<th>protocol</th>
<th>flows</th>
<th>octets</th>
<th>packets</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td></td>
<td>378694</td>
<td>5668763771</td>
<td>8060656</td>
</tr>
<tr>
<td>17</td>
<td></td>
<td>541505</td>
<td>352973483</td>
<td>1745931</td>
</tr>
<tr>
<td>50</td>
<td></td>
<td>1578</td>
<td>172325976</td>
<td>424695</td>
</tr>
<tr>
<td>47</td>
<td></td>
<td>641</td>
<td>55589150</td>
<td>165236</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td>14176</td>
<td>5970023</td>
<td>69637</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>30</td>
<td>2124</td>
<td>59</td>
</tr>
<tr>
<td>103</td>
<td></td>
<td>28</td>
<td>1064</td>
<td>28</td>
</tr>
<tr>
<td>41</td>
<td></td>
<td>4</td>
<td>464</td>
<td>4</td>
</tr>
</tbody>
</table>

- Find traffic to/from specific IP address
- Example is anonymized

```
nfdump -r 2006-05-12.0800.nfdump -o 'fmt:%td %sa %da %byt' 'ip 166.81.189.132'
```

<table>
<thead>
<tr>
<th>Duration</th>
<th>Src IP Addr</th>
<th>Dst IP Addr</th>
<th>Bytes</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.952</td>
<td>212.187.175.58</td>
<td>166.81.189.132</td>
<td>552</td>
</tr>
<tr>
<td>0.952</td>
<td>166.81.189.132</td>
<td>212.187.175.58</td>
<td>6563</td>
</tr>
<tr>
<td>0.180</td>
<td>166.81.189.132</td>
<td>190.88.151.169</td>
<td>468</td>
</tr>
<tr>
<td>0.184</td>
<td>190.88.151.169</td>
<td>166.81.189.132</td>
<td>1446</td>
</tr>
<tr>
<td>0.524</td>
<td>68.129.203.157</td>
<td>166.81.189.132</td>
<td>536</td>
</tr>
<tr>
<td>0.520</td>
<td>166.81.189.132</td>
<td>68.129.203.157</td>
<td>762</td>
</tr>
<tr>
<td>1.048</td>
<td>192.127.132.177</td>
<td>166.81.189.132</td>
<td>2569</td>
</tr>
<tr>
<td>1.044</td>
<td>166.81.189.132</td>
<td>192.127.132.177</td>
<td>102321</td>
</tr>
</tbody>
</table>

---

**info@ist-lobster.org**

Ane Øislebe, UNINETT
• Who sendt the most traffic to this IP address?

```
nfdump -r 2006-05-12.0800.nfdump -n5 -ssrcip/bytes 'dst ip 166.81.189.132'
```

<table>
<thead>
<tr>
<th>Duration Proto</th>
<th>Src IP Addr</th>
<th>Flows</th>
<th>Packets</th>
<th>Bytes</th>
</tr>
</thead>
<tbody>
<tr>
<td>910.083</td>
<td>190.88.151.169</td>
<td>152</td>
<td>976</td>
<td>180764</td>
</tr>
<tr>
<td>904.953</td>
<td>166.81.191.137</td>
<td>14</td>
<td>655</td>
<td>117439</td>
</tr>
<tr>
<td>902.915</td>
<td>139.65.84.134</td>
<td>172</td>
<td>1413</td>
<td>106792</td>
</tr>
<tr>
<td>848.974</td>
<td>192.127.132.177</td>
<td>35</td>
<td>1253</td>
<td>92450</td>
</tr>
<tr>
<td>876.597</td>
<td>166.81.98.117</td>
<td>36</td>
<td>364</td>
<td>56876</td>
</tr>
</tbody>
</table>

info@ist-lobster.org  
Arne Øslebe, UNINETT

---

• Long term statistics
• Easy to use web frontend
  • Text based reports and graphs
• Support for different types of network statistics
  • Netflow
  • SNMP  
  [http://software.uninett.no](http://software.uninett.no)
  • Mping
• Easy to add new reports
  • Templates and plugins
• Access control
  • Observation points and reports
• Handle database problems
  • Store raw data to disk
  • Generate reports later when problems are resolved
• Avoid multiple instances of the same backend
  • Wait for previous instances that are still processing the raw data
  • Detect dead locks and memory starvations
Our NetFlow setup

- 27 routers
- 207 interfaces
- >30Gb of raw Netflow data every day
- 400.000 new entries in the db every hour
- >450 millions entries in a single table
- >700Gb database size

Performance

Data from January 17 between 08:00-09:00

<table>
<thead>
<tr>
<th></th>
<th>PC1</th>
<th>PC2</th>
<th>PC3</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Netflow size</strong></td>
<td>537MB</td>
<td>161MB</td>
<td>399MB</td>
<td>1097MB</td>
</tr>
<tr>
<td><strong>Sequentially</strong></td>
<td>9min 17s</td>
<td>3min 8s</td>
<td>5min 51s</td>
<td>18min 16s</td>
</tr>
<tr>
<td><strong>No insert in DB</strong></td>
<td>7min 11s</td>
<td>1min 48s</td>
<td>4min 52s</td>
<td>13min 51s</td>
</tr>
<tr>
<td><strong>Simultaneously</strong></td>
<td>9min 21s</td>
<td>3min 7s</td>
<td>5min 56s</td>
<td>18min 24s</td>
</tr>
<tr>
<td><strong># of new DB entries</strong></td>
<td>164702</td>
<td>69549</td>
<td>184706</td>
<td>418957</td>
</tr>
<tr>
<td><strong># of entries/second</strong></td>
<td>295.69</td>
<td>369.94</td>
<td>526.23</td>
<td>382.26</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Report</th>
<th>Time</th>
<th># of entries in table</th>
</tr>
</thead>
<tbody>
<tr>
<td>IP Protocol overview report</td>
<td>614ms</td>
<td>17,228,300</td>
</tr>
<tr>
<td>IP Protocol overview report (previous timeperiod)</td>
<td>524ms</td>
<td></td>
</tr>
<tr>
<td>IP Protocol detailed report</td>
<td>509ms</td>
<td></td>
</tr>
<tr>
<td>Top src port overview</td>
<td>1221ms</td>
<td>454,034,919</td>
</tr>
<tr>
<td>Top src port overview (previous timeperiod)</td>
<td>717ms</td>
<td></td>
</tr>
<tr>
<td>Top src port detailed</td>
<td>498ms</td>
<td></td>
</tr>
<tr>
<td>Top src port plot (one day, two ports and obs. points)</td>
<td>1590ms</td>
<td></td>
</tr>
</tbody>
</table>
Some links

- http://www.switch.ch/tf-tant/floma/software.html
- http://software.uninett.no/
- http://www.splintered.net/sw/flow-tools/
- http://nfdump.sourceforge.net/

info@ist-lobster.org  Arne Øslebe, UNINETT