

INFORMATION SOCIETY TECHNOLOGIES (IST)
PROGRAMME



Large Scale Monitoring of BroadBand Internet Infrastructure
Contract No. 004336

D2.2: Prototype implementation

Abstract:

This document describes the implementation of the core components of Lobster.

Contractual Date of Delivery	31 March 2006
Actual Date of Delivery	24 October 2006
Deliverable Security Class	Public

The LOBSTER Consortium consists of:

FORTH	Coordinator	Greece
VU	Principal Contractor	The Netherlands
CESNET	Principal Contractor	Czech Republic
UNINETT	Principal Contractor	Norway
SYMANTEC	Principal Contractor	United Kingdom
ALCATEL	Principal Contractor	France
FORTHnet	Principal Contractor	Greece
TNO	Principal Contractor	The Netherlands
TERENA	Principal Contractor	The Netherlands

Chapter 1

Introduction

The CD that is enclosed with this deliverable contains the core software of the Lobster project. The design of this software was described in Deliverable 2.1 [4]. In the remainder of this chapter we give an overview of the software that is provided. In Chapter 2 we give installation instructions for the software packages.

1.1 MAPI

The Monitoring API (MAPI) was originally designed and implemented in the SCAMPI project as a new API for doing passive monitoring. The main goal when designing MAPI was to create an architecture that added as little overhead as possible on the actual processing of packets. MAPI is also a multiuser API where multiple users can run several monitoring jobs concurrently and it was important to create an architecture where global optimisation based on all monitoring jobs from all users would be possible. It was also a high priority to make MAPI extensible so that new packet processing functions could easily be added and it was important to be able to run the same applications on top of different hardware without having to make any changes to the application.

In LOBSTER, several improvements have been made to MAPI. Support for bi-directional measurement with two adapters in the same host is now supported, proper global optimisation has been added and the performance for offline processing of flows have greatly been improved. Several new measurement functions have also been added and the overall quality of the code has been improved.

This chapter provides a description of the MAPI architecture and how it works and some details on the features added in LOBSTER. A more general overview of how to use MAPI can be found in LOBSTER deliverable D1.2.

1.2 Ruler

The MAPI packet filtering system described in §1.1 is very effective, but in some cases is not flexible enough: only a predefined set of anonymisation operations

can be applied. To complement the MAPI system, the Ruler anonymisation language was developed. It describes anonymisation operations on network packets as pattern matching and rewrite operations.

The packets are selected by matching them against regular expressions, and the rewriting is done by copying selected parts of the matching expression into the resulting packet.

For example, suppose that we want to monitor page requests to a web server. To request a page on web server, a packet is sent that contains the following line:

```
GET /index.html HTTP/1.0
```

Where `/index.html` is the name of the web page in this request. To extract these page requests from a packet stream, we could use the following filter:

```
include "layouts.rli"
```

```
filter zap_url
  head:(Ethernet_header IPv4_header * "GET ")
  url:*
  tail: (0x0D 0x0A *)
=> accept;
```

We require that the packet starts with an Ethernet header and an IP version 4 packet header. The layout of these headers is defined in the include file `layouts.rli` which is not shown here. We match all packets with these headers a prefix of unknown length, the characters 'GET ', another span of bytes of unknown length, a carriage return, a line feed, and the remainder of the packet.

In the example filter the packet is accepted, so it is copied to the output without change. By default, all other packets are not copied to the output.

Instead of just accepting or rejecting a packet, it is also possible to create a new output packet from fragments of the input packet and constant patterns. For example, the filter

```
include "layouts.rli"
```

```
filter zap_url
  head:(Ethernet_header IPv4_header * "GET ")
  url:*
  tail: (0x0D 0x0A *)
=> head "XXX" tail;
```

uses the same match pattern as the previous example, but now only parts of the input packet are copied to the output (the parts that are labelled `head` and `tail`). The requested URL is not copied to the output. Instead, the string `XXX` is inserted in the output packet.

It is also possible to apply further operations on the output packet, including MAPI functions. For example, the filter

```
include "layouts.rli"
```

```
atom hash : 4 :
  [class="MAPI",real_name="HASH",library="mapidstdflib.so"]
```

```
filter hash_url
  head:(Ethernet_header IPv4_header * "GET ")
```

```
url:*  
tail: (0x0D 0x0A *)  
=> head @hash(url) tail;
```

is similar to the previous filter, but replaces the requested URL with a hash value over the web page name that is computed using the MAPI hash function.

The pattern matching and rewriting approach that is used in Ruler allows great flexibility and clarity in specifying an anonymisation policy. This encourages system administrators to tailor the anonymisation policy to their exact needs, and strike the right balance between security and the need for detailed information by the network monitoring application.

Techniques for accepting or rejecting patterns based on regular expressions are well known [1]. For Ruler it is also necessary to *tag* positions in the regular expression, so that selected parts of the matched pattern can be accessed. Algorithms for this problem are much less studied, but fortunately some techniques for implementing this have been published [2].

1.3 Template manager

Next to the two main software packages of the Lobster project, the enclosed CD also contains a third software package called *template manager*. Template Manager is a support program for Ruler. It generates administration code for Ruler, and provides a library of helper functions. This package was not developed in the Lobster project; it is provided as background material. Further information can be found in [3].

CHAPTER 1. INTRODUCTION

Chapter 2

Installation

The software has mainly been developed and tested on Linux systems. However, the software does not depend on any Linux-specific feature, so we expect that the software can be installed on other Unix-like systems with little or no modification.

2.1 MAPI

The MAPI distribution is packaged in the file `mapi-2.0-beta1.tar.gz`.

To install MAPI, take the following steps:

1. Unpack the distribution with `tar xf mapi-2.0-beta1.tar.gz`
2. Enter the distribution directory with `cd mapi-2.0-beta1`
3. Run `./configure`
4. Run `make`
5. Run `su -`
6. Run `make install`
7. If necessary, change `mapi.conf` (usually located in `/etc/mapi/`, depending on the installation path) to reflect your system (e.g., add/remove monitoring interfaces).
8. Run `mapid`
9. Run `mapi commd` (only if distributed monitoring support is enabled).
10. Run some MAPI applications - the 'tests' directory is a good start.

The `configure` script that is run in the first step can be given a number of useful options:

- `--enable-dimapi` Enables support for remote and distributed monitoring.
- `--enable-dag` Enables support for Endace's DAG packet capture cards.
- `--enable-ssl` Enables encryption of DiMAPI traffic.

2.2 Template Manager

Template Manager is a support program for Ruler. It generates administration code for Ruler, and provides a library of helper functions. It is packaged in the file `tmkernel-2.2.0.tar.gz`. To install Template Manager take the following steps:

1. Unpack the distribution with `tar xf tmkernel-2.2.0.tar.gz`
2. Enter the distribution directory with `cd tmkernel-2.2.0`
3. Run `./configure`
4. Run `make`
5. Run `make install`

2.3 Ruler

Ruler is packaged in the file `ruler-0.3.2.tar.gz`. To install Ruler, make sure that you have installed Template Manager (see above), and then take the following steps:

1. Unpack the distribution with `tar xf ruler-0.3.2.tar.gz`
2. Enter the distribution directory with `cd ruler-0.3.2`
3. Run `./configure`
4. Run `make`
5. Run `make install`

Bibliography

- [1] Alfred V. Aho and Jeffrey D. Ullman. *Foundations of Computer Science*. Computer Science Press, 1992.
- [2] Ville Laurikari. NFAs with tagged transitions, their conversion to deterministic automata and application to regular expressions. In *SPIRE*, pages 181–187, 2000.
- [3] C. van Reeuwijk. Rapid and robust compiler construction using template-based metacompilation. In *Proc. of the Compiler Construction conference*, Warsaw, Poland, April 2003.
- [4] The Lobster Consortium. Deliverable D2.1: Preliminary implementation report, January 2005. <http://www.ist-lobster.org/deliverables/D2.1.pdf>.