Network Monitoring for Performance and Security: the LOBSTER Project

Michalis Polychronakis
mikepo@ics.forth.gr

Institute of Computer Science (ICS)
Foundation for Research and Technology – Hellas (FORTH)

3 October 2006
LOBSTER Partners

- **LOBSTER**: Large-scale Monitoring of Broadband Internet Infrastructure

- **Partners:**
  - *Research organizations*
    - ICS-FORTH, Greece (coordinator)
    - Vrije Universiteit, The Netherlands
    - TNO Telecom, The Netherlands
  - *NRNs and ISPs*
    - CESNET, Czech Republic
    - UNINETT, Norway
    - FORTHnet, Greece
    - TERENA, The Netherlands
  - *Industrial Partners*
    - Alcatel, France
    - Symantec, UK

- Successor of the IST SCAMPI project
LOBSTER Overview

Motivation
- Improve our understanding of the traffic on the Internet
- Is my network under attack?
- Are there any compromised computers?
- How much P2P traffic is there?

Main goal:
- Develop an advanced European infrastructure for Internet traffic monitoring
- A network of passive monitors
- Enable easy, flexible, and efficient distributed passive monitoring
Current Sensor Deployment

- Real time traffic monitoring
- Monitoring using traffic traces
Current Sensor Deployment

Monitoring using traffic traces
What has been developed so far?

- A programming environment for distributed passive network monitoring
  - MAPI (Monitoring Application Programming Interface)

- An implementation of MAPI on top of different monitoring cards
  - Regular NICs
  - DAG Cards (Endace)
  - Combo6 Cards (CESNET, Masaryk Univ.)

- Several real-world applications
MAPI: Network Flow Abstraction

- MAPI is based on a generalized **Network Flow** abstraction
  - A network flow is a subset of the traffic

```
Monitored network traffic

... A B C ...

"Incoming web server traffic"

"HTTP GET requests"

"CodeRed worm"
```

- **Packet with destination port 80**
- **Packet with destination port 80 containing the string "GET "**
- **Packet with destination port 80 containing the string "GET /default.ida?NNNNN..."**
MAPI: Predefined Functions

- The traffic of a flow is processed by chaining functions

  - **Filtering**
    - BPF filters
    - Keep packets with a specific string/regexp in their payload
    - Keep all traffic of specific protocols (FTP, GNUTELLA, BitTorrent, …)

  - **Processing**
    - Count the number of packets/bytes
    - Dump packets to disk
    - Packet sampling (probabilistic/deterministic)
    - IP defragmentation and TCP stream reassembly
    - Compute packet digests
    - Anonymize packets
    - IPFIX flow export
      - *many more*… (+ custom functions)
MAPI: Example Code

**Count the number of packets destined to our web servers**

```c
fd = mapi_create_flow("/dev/dag0");
mapi_apply_function(fd, "BPF_FILTER", "tcp and dst port 80");
 fid = mapi_apply_function(fd, "PKT_COUNTER");
mapi_connect(fd);

while(1) {
    sleep(1);
    num_packets = mapi_read_results(fd, fid);
    /* ... */
}
```
MAPI: Distributed Monitoring

- Receive traffic from a local interface
  \texttt{mapi\_create\_flow(“eth0”);}

- Receive traffic from \textbf{multiple remote} interfaces
  \texttt{mapi\_create\_flow(“sensor.uninett.no:/dev/dag0, 139.91.231.2:eth1”);}

- Minimal network overhead
  - Packets are processed locally on each sensor
  - The user application receives only the information \textit{really} needed!
Applications

- **Performance monitoring**
  - **appmon**: per-application traffic categorization
  - **abw**: bandwidth usage monitoring
  - **stager**: generic statistics aggregation and visualization

- **Traffic anonymization**
  - **anontool**: flexible traffic anonymization

- **0-day attack detection**
  - **ear**: worm detection
  - **nem**: polymorphic shellcode detection using emulation
**appmon: Traffic Categorization (1/2)**

- Widely used applications that use dynamic ports
  - Not known in advance
  - **Difficult to monitor**

- We are not sure even for *known* ports any longer…
  - Most of port 80 traffic is not web traffic!

- LOBSTER uses deep packet inspection to accurately attribute traffic to applications
**appmon: Traffic Categorization (2/2)**

LOBSTER P2P Monitoring: traffic from/to Crete School Network Monitor (3 hours)

(1 minute average)

**INCOMING TRAFFIC**

<table>
<thead>
<tr>
<th>IP</th>
<th>Protocol</th>
<th>Traffic</th>
</tr>
</thead>
<tbody>
<tr>
<td>194.63.218.XX</td>
<td>eDonkey</td>
<td>360.13 Kbps</td>
</tr>
<tr>
<td>194.63.217.XX</td>
<td>BitTorrent</td>
<td>246.37 Kbps</td>
</tr>
<tr>
<td>81.186.52.XXX</td>
<td>BitTorrent</td>
<td>193.08 Kbps</td>
</tr>
<tr>
<td>81.186.52.XXX</td>
<td>eDonkey</td>
<td>126.21 Kbps</td>
</tr>
<tr>
<td>194.63.218.XXX</td>
<td>BitTorrent</td>
<td>92.91 Kbps</td>
</tr>
<tr>
<td>194.63.216.XXX</td>
<td>eDonkey</td>
<td>79.09 Kbps</td>
</tr>
<tr>
<td>81.186.52.XXX</td>
<td>BitTorrent</td>
<td>78.95 Kbps</td>
</tr>
<tr>
<td>194.63.237.XXX</td>
<td>SMTP</td>
<td>57.45 Kbps</td>
</tr>
<tr>
<td>81.186.61.XX</td>
<td>BitTorrent</td>
<td>49.04 Kbps</td>
</tr>
<tr>
<td>81.186.54.XXX</td>
<td>BitTorrent</td>
<td>43.20 Kbps</td>
</tr>
</tbody>
</table>

**OUTGOING TRAFFIC**

<table>
<thead>
<tr>
<th>IP</th>
<th>Protocol</th>
<th>Traffic</th>
</tr>
</thead>
<tbody>
<tr>
<td>194.63.235.XX</td>
<td>BitTorrent</td>
<td>885.69 Kbps</td>
</tr>
<tr>
<td>194.63.216.XXX</td>
<td>eDonkey</td>
<td>296.38 Kbps</td>
</tr>
<tr>
<td>194.63.216.XXX</td>
<td>BitTorrent</td>
<td>188.04 Kbps</td>
</tr>
<tr>
<td>81.186.52.XXX</td>
<td>BitTorrent</td>
<td>180.96 Kbps</td>
</tr>
<tr>
<td>81.186.52.XXX</td>
<td>eDonkey</td>
<td>105.18 Kbps</td>
</tr>
<tr>
<td>81.186.52.XXX</td>
<td>BitTorrent</td>
<td>89.37 Kbps</td>
</tr>
<tr>
<td>194.63.217.XXX</td>
<td>eDonkey</td>
<td>83.29 Kbps</td>
</tr>
<tr>
<td>194.63.217.XXX</td>
<td>BitTorrent</td>
<td>81.74 Kbps</td>
</tr>
<tr>
<td>81.186.52.XXX</td>
<td>BitTorrent</td>
<td>36.27 Kbps</td>
</tr>
<tr>
<td>194.63.237.XXX</td>
<td>MAPI</td>
<td>33.44 Kbps</td>
</tr>
</tbody>
</table>
abw: Bandwidth Usage Monitoring

https://perfmon.cesnet.cz/abw-gn2/
**stager**: Statistics Aggregation/Visualization

An IST Project

[http://www.ist-lobster.org](http://www.ist-lobster.org)

[http://stager.uninett.no/](http://stager.uninett.no/)

---

<table>
<thead>
<tr>
<th>Source AS</th>
<th>Destination AS</th>
<th>Octets</th>
<th>Packets</th>
<th>Flows</th>
<th>Packetsize</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number</td>
<td>Name</td>
<td>bit/s</td>
<td>Percent</td>
<td>Packets/s</td>
<td>Percent</td>
</tr>
<tr>
<td>2503</td>
<td>NORDUnet</td>
<td>73.3M</td>
<td>39.71%</td>
<td>121-10⁴</td>
<td>53.30%</td>
</tr>
<tr>
<td>2503</td>
<td>NORDUnet</td>
<td>37.0M</td>
<td>20.44%</td>
<td>73.9-10⁴</td>
<td>21.60%</td>
</tr>
<tr>
<td>2503</td>
<td>NORDUnet</td>
<td>8.33M</td>
<td>4.62%</td>
<td>18.4-10⁴</td>
<td>5.39%</td>
</tr>
<tr>
<td>15659</td>
<td>15659</td>
<td>5.69M</td>
<td>3.08%</td>
<td>17.5-10²</td>
<td>5.10%</td>
</tr>
<tr>
<td>64518</td>
<td>64518</td>
<td>5.07M</td>
<td>2.75%</td>
<td>5.61-10³</td>
<td>1.64%</td>
</tr>
<tr>
<td>1653</td>
<td>SUNET Swedish</td>
<td>3.15M</td>
<td>1.71%</td>
<td>2.54-10⁴</td>
<td>0.74%</td>
</tr>
<tr>
<td>21293</td>
<td>21293</td>
<td>2.86M</td>
<td>1.55%</td>
<td>2.21-10⁴</td>
<td>0.65%</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>2.47M</td>
<td>1.34%</td>
<td>3.51-10⁴</td>
<td>1.03%</td>
</tr>
<tr>
<td>1257</td>
<td>SWIPnet Swedish</td>
<td>2.37M</td>
<td>1.29%</td>
<td>4.24-10⁴</td>
<td>1.24%</td>
</tr>
<tr>
<td>1257</td>
<td>SWIPnet Swedish</td>
<td>2.03M</td>
<td>1.10%</td>
<td>3.15-10³</td>
<td>0.92%</td>
</tr>
</tbody>
</table>
anontool: Traffic Anonymization

- Command line tool, based on the MAPI anonymization function
- **Flexible:** choose what and how will be anonymized
  - what
    - IP, TCP, UDP, ICMP header fields
    - High-level protocol fields (currently HTTP, FTP, and NetFlow v5/v9)
    - Packet payload
  - how
    - Obfuscation: (MAP, RANDOMIZE, HASH, REPLACE, PATTERN_FILL, …)
    - Remove field
    - Prefix-preserving IP address anonymization
    - Transform *arbitrary* payload parts using regular expressions
- Predefined widely-used anonymization policies for convenience
  http://www.ics.forth.gr/dcs/Activities/Projects/anontool.html
ear: Content-based 0-day Worm Detection
nem: Emulation-based Attack Detection (1/3)

- Network-level 0-day shellcode detector
  - Detects self-modifying polymorphic shellcode

- **Main idea:** execute each network service request as if it were executable code

- **Goal:** identify the execution behavior inherent in polymorphic shellcodes

- Resilient to advanced evasion methods
  - Highly polymorphic and self-modifying code
**nem:** Emulation-based Attack Detection (2/3)

### GET /index.php HTTP/1.1 Host: www.foo.com

```
GET /index.php
HTTP/1.1 Host: www.foo.com
```

### Random code

```
inc edi
inc ebp
push esp
and [edi], ch
imul ebp, [esi+0x64], dword 0x702e7865
push dword 0x54482070
push esp
push eax
das
xor [esi], ebp
xor [eax], esp
...  
```

Usually crashes after a few instructions

---

**benign request**
nem: Emulation-based Attack Detection (2/3)

\x6A\x0F\x59 \xE8\xFF\xFF \xFF\xFF\xC1 ...

\x6A\x0F\x59\xE8\xFF\xFF\xFF\xFF\xFF\xFF\xC1\x5E\x80...

6A07
59
E8FFFFFFFF
FFC1
5E
80460AE0
304C0E0B
E2FA
...

push byte +0x7f
pop ecx
call 0x7
inc ecx
pop esi
add [esi+0xa],0xe0
xor [esi+ecx+0xb],cl
loop 0xe
xor [esi+ecx+0xb],cl
loop 0xe
xor [esi+ecx+0xb],cl
...

Standard behavior

getPC code (for finding its place in memory)

Lots of reads from its own memory locations (corollary of the decryption process)

✗ malicious request!
**nem: Emulation-based Attack Detection (3/3)**

- General info
- Full TCP stream dump...
- ...with the shellcode bytes highlighted
- Complete execution trace of the decryption process
Download Information

- MAPI
  - Version 2.0 beta1 has been recently released!
    http://mapi.uninett.no/

- Applications
  - appmon – included in MAPI distribution
  - stager
    http://stager.uninett.no/
  - abw – included in MAPI distribution
    https://perfmon.cesnet.cz/abw-gn2/
  - anontool
    http://www.ics.forth.gr/dcs/Activities/Projects/anontool.html
thank you!

http://www.ist-lobster.org/
fallback slides
Network Flow Scope Example

```
fd = mapi_create_flow("sensor.uninett.no:/dev/dag0,
                     "mon1.ics.forth.gr:eth0");
mapi_apply_function(fd, "BPF_FILTER", "dst port 80");
```

User

Packet to port 80
MAPI Architecture

Monitoring Sensor

Communication Agent (commd)

DiMAPI stub

Admission Control Daemon (authd)

Monitoring Daemon (mapid)

I/O Bus

Captured Packets

shared memory

UNIX socket / shared memory

TCP socket

TCP socket

User A

Application 1

DiMAPI stub

User B

Application 2

DiMAPI stub
Witty worm Snort signature

```
alert udp any 4000 -> any any (msg:"ISS PAM/Witty Worm Shellcode"; content: "|65 74 51 68 73 6f 63 6b 54 53|"; depth:246;
```

DiMAPI code

```
fd = mapi_create_flow("<scope>");

mapi_apply_function(fd, BPF_FILTER, "udp and src port 4000");

mapi_apply_function(fd, STR_SEARCH, "|65 74 51 68 73 6f 63 6b 54 53|", 0, 246);
```
Distributed Intrusion Detection (2/2)
Remote Packet Capture vs DiMAPI

- **Capture CodeRed worm packets**
  - TCP, port 80, payload: `GET default.ida?NNNNNNNNNNNNN...`

**Remote Libpcap**
Receive all port 80 packets
Then search locally for CodeRed packets

**DiMAPI**
Just receive CodeRed packets
Unknown Traffic

In recent years we practically see either port 80 or unknown traffic.

Probably lots of applications use port 80 to bypass firewall restrictions.

2002: several known ports
2006: only port 80