TNO Information & Communication–Technology

Update Lobster & NERD
Agenda

• Lobster:
  • What is Lobster?
  • Status

• NERD:
Lobster

• Main goal:
  • To develop an advanced European infrastructure for passive network monitoring.

• 9 Partners:
  • ALCATEL, CESNET, ENDANCE, FORTH, FORTHNET, Vrije Universiteit Amsterdam, Terena, TNO, UNINETT

• Duration: oct 2004 – dec 2006
Possible Lobster Applications

- Accurate traffic characterisation for programs using dynamic ports
- Spread of zero-day worms
- European Internet measurement service
- End-to-end performance debugging
- Application performance measurement
- Trace DoS attacks
- Test platform for IPFIX attributes
Lobster

- High speed network monitoring (10Gbps)
- Use of Dedicated programmable hardware (fpga cards – DAG, SCAMPI)
- Monitoring application programming interface (MAPI)
- Multiple network sensor API (distributed MAPI)
- Cross domain monitoring
- Anonymisation framework
- Access control
- Demo applications
Lobster vs. Geant2

- Same ‘member’ community
- Lobster also tries to include commercial members (ISPs)
- Passive monitoring only
- Use same (passive) measurement data
- Equal infrastructure design (Lobster adapts from JRA1)
- Same security applications
- Not only security applications
- Lobster has shorter time span (2 years) (more pressed for demo apps)
Lobster status

• Requirements analysis – done!
  • Req. collection, acceptable use policy for fair sharing

• Monitoring infrastructure design – due Oct '05
  • Anonymisation framework definition, Common access platform definition,
    first-tier encryption definition, integrated architecture definition

• Monitoring infrastructure realisation – due Apr '06
  • Prototype

• Monitoring infrastructure deployment – due Jan '06
  • LOBSTER applications, Monitoring infrastructure
• History
• How does it work?
  • real-time analysis
  • post analysis
  • web user interface
• Status
• Future
  • from application to framework
  • Integration in LOBSTER, Geant2.
History

• 2002: SURFnet and TNO initiated a research project into DoS detection on the SURFnet network
• End of 2002: Prototype (NERD v0.1) finished, based on Caida’s cflowd, flowtools, gnuplot and shell scripts.
• 2004: Design & development of NERD v0.5, removed third party tools by rewriting the daemon
• 2004: Design & development of NERD v1.0, bugfixes on daemon and new user interface
• 2004: Application to be used in Lobster
• 2005: potential security tool used in GN2/JRA2 (SURFnet)
• 2005 March 18: Open source release NERD (1.03beta)
• NERD – Network Emergency Responder & Detector

• Collects NetFlow
• A tool that detects DoS attacks
• raises Alarms
• flexible search through stored NetFlow data
How does it work?

- Input data: NetFlow
- real-time analyse
  - Output: alarms in database
- post analysis
  - Output: flow-tools style data (text)
- web based GUI
The collector

- Collector
  - simple UDP receiver (binds to multiple IP/port)
- Pre-processor
  - source specific functions (ex. filter double flows)
- Data stored on disk
  - for the post analysis
- Data kept in memory
  - for real-time analysis
Real-time analysis

- Every x minutes the Rules (1..n) are executed
- Rule:
  - 1 filter (ex. src_addr = 123.0.0.0/16 and dst_port != 80)
  - 1..n clusters (cluster on dst_ip and count flows)
  - threshold (#flows > 1,000,000)
  - output formatting (alarm in database)
Post analysis

- Executed at users request
- Rule:
  - 1 filter (same as real-time analysis)
  - 1..n clusters (same as real-time analysis)
  - (no threshold)
  - output formatting (flow-tools like text files)
Configuration

• Stored in database
• Rule record = filter + cluster
  • making filters and clusters reusable
  • multi user prepared
GUI screenshots
## Alarms of Wednesday, September 8th 2004

<table>
<thead>
<tr>
<th>Starttime (GMT+1)</th>
<th>Stoptime (GMT+1)</th>
<th>Rulename</th>
<th>Alarem message</th>
<th>Trigger</th>
</tr>
</thead>
</table>
| Wed, Sept 8th, 11:00 | Wed, Sept 8th, 12:31 | Flood detection | **Source IP address 10.0.0.138 with Destination IP address 192.168.0.2 has 12832 connections in 5 minutes** | 9000 connections in 5 minutes  
|                   |                   |                | [Analyse]                                                                      |                                              |
| Wed, Sept 8th, 12:50 | * Not stopped *   | Portscan detection | **Source IP address 127.0.0.1 has 20316 destination ports in 5 minutes**       | 15000 destination ports in 5 minutes         
|                   |                   |                | [Analyse]                                                                      |                                              |
Set Analysis

IPv4 source address & IPv4 destination address

Records: 165395  Min val: 0.0.23.52  Max val: 223.255.225.21

flows

Analyse
## Edit Filters

<table>
<thead>
<tr>
<th>Name &amp; Description</th>
<th>Filter expression</th>
<th>IPv4 destination IP address</th>
<th>Destination port</th>
</tr>
</thead>
<tbody>
<tr>
<td>All traffic</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Don’t filter out anything</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>No Big Servers</td>
<td>IPv4 destination IP address</td>
<td>145.7.192.133</td>
<td></td>
</tr>
<tr>
<td>Whitelist for flood detection</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Worm Whitelist</td>
<td>Destination port</td>
<td>1214</td>
<td></td>
</tr>
<tr>
<td>Whitelist for worm detection</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>No Normal Mailservers</td>
<td>Destination port</td>
<td>6881</td>
<td></td>
</tr>
<tr>
<td>Whitelist for mailservers</td>
<td>IPv4 destination IP address</td>
<td>145.7.191.15</td>
<td></td>
</tr>
</tbody>
</table>

- [Add Expression](#)
- [Add Expression](#)
- [Add Expression](#)
- [Add Expression](#)
- [Add Expression](#)

**Apply**
<table>
<thead>
<tr>
<th>Name &amp; Description</th>
<th>Cluster</th>
<th>Filter</th>
<th>Threshold</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Portscan Detection</strong></td>
<td>Dest. Ports</td>
<td>All Traffic</td>
<td>number of flows &gt; 15000</td>
</tr>
<tr>
<td>Checks the number of dest. ports</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Flood Detection</strong></td>
<td>Dest. IP addresses</td>
<td>No Big Servers</td>
<td>number of flows &gt; 9000</td>
</tr>
<tr>
<td>Checks the number of connections</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Worm Detection</strong></td>
<td># Packets &amp; Dest. Port</td>
<td>Worm Whitelist</td>
<td>number of flows &gt; 1000</td>
</tr>
<tr>
<td>Combines #packets and dest. port</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Open Relay Detection</strong></td>
<td>Dest. IP &amp; Dest. Port</td>
<td>No Normal MailServers</td>
<td>number of flows &gt; 100</td>
</tr>
<tr>
<td>Checks for mass mailing</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Status

• Beta testers wanted!

• Short term todo list:
  • web-site/ subversion
  • documentation/ white paper
  • more intuitive interface
  • ipv6 & netflow v9 bug fixes

• Mid-long term (next year)
  • worm detection
  • 3D data representation (student)
  • Flexible data analysis (connection to ROOT/ MatLab etc.)
  • Integration into JRA1/ Lobster architecture
  • from application to framework…
From application to framework

- Other (data) sources
  - tcpdump, hardware cards, snort, firewall,
  - pcap/ raw format, alarms/ logging, XML,
- Combining different data
  - ex. fw or httpd log with network data for worm detection
- Other data output
  - graphs, alarms, top 10 list, XML reports, NetFlow
- Modular building blocks
  - basic function blocks
- Offer APIs for self-made feature extraction
Framework

Data source (1..n)
- router
- tcpdump
- hardware cards
- snort
- the weather
- your special device
Data
- netflow
- snmp
- pcap files
- raw data
- XML
- your special data (or format)
Collector
- simple receive (udp)
- pull mechanism
- snmp client
- sftp/ssh
- you own special method (MARP)
Pre-process
- filter
- sanity check
- buffering
- reassembling
Filter (1)
- Multiple fields
- multiple fields
- source or data specific
Logic (1..n)
- has memory
- group data
- count
- math. Func.
- bpa
Detection (0..n)
- exceed threshold
- filter
Response (0..n)
- send email
- start ext. script
- create ACL
- tune sensor!
- tune own config!
- start analyse
Output (1..n)
- flow-tools reports
- RRD
- html
- XML (DMIF)
- send as netflow

Rule (1..n)
- multiple rules
- time driven
- (interval, abs. time)
- continuously
- every x of data
- best effort

Data source (1..n)
- temporary data
- learned data
- Output data -->
- can be shared for other rules
Configuration Example

- **NetFlow**
- **Raw packets**

Filter:
- Logic (DoS detection algorithm)
- Detection (classify)
- Response
- Output formatting

Output formatting:
- Post-process (find evidence)
- Graph/RRD/html
- e-mail
- Alarms Database
- Evidence file

Path I:
- Logic (average)
- Output formatting
- Graph/RRD/html

Path II:
- Output formatting
- Storage for post-analysis

Path III:
- Raw data
• Lobster site: www.ist-lobster.org

• NERD: www.nerdd.org (will be up soon)
• info@nerdd.org

• Hans Hoogstraaten
• J.M.Hoogstraaten@telecom.tno.nl

• TNO - To apply scientific knowledge with the aim of strengthening the innovative power of industry and government - www.tno.nl