Scalable Analysis for Network Monitoring and Forensics Purposes

Jérôme François
Introduction

Some facts
Motivation

Traffic analysis

Anomaly detection
Botnet detection

Topology analysis

Bad behaviors in Internet
Detection
Evaluation

Conclusion
1. Introduction
   Some facts
   Motivation

2. Traffic analysis
   Anomaly detection
   Botnet detection

3. Topology analysis
   Bad behaviors in Internet
   Detection
   Evaluation

4. Conclusion
- 2,2 billions users, 200 millions servers
  - Cisco measured and forecasted Internet traffic (1000 PB/day)
DDoS Attacks

Largest Single DDoS Attack Observed per Survey
Year in Gbps

Source: Arbor Networks, Inc.
Web based attacks

Average Web-based attacks per day, by month, 2009–2010

Source: Symantec Corporation
Some facts about botnets

- **Botnets**

- **Botnet monitoring** *(Measurement, Detection, Disinfection and Defence, ENISA report 2011):*
  - Shadowserver Foundation: 5000-6000 alive botnets (100000-250000 bots) simultaneously in 2005
  - Conficker working group: 1 000 000 - 3 000 000 alive zombies (2009)
  - Securelist.com: 3 600 000 zombies within US only (2009)
1. Introduction
   - Some facts
   - Motivation

2. Traffic analysis
   - Anomaly detection
   - Botnet detection

3. Topology analysis
   - Bad behaviors in Internet
   - Detection
   - Evaluation

4. Conclusion
Why attacks are powerful?

**Motivation**
- challenging aspects / attacker competitiveness... past trend, too risky today
- **win money!**
  - abuse (spam, click fraud)
  - attack the competitors (steal information, disrupt services)
  - $15 = 10,000 bots (source: Symantec)
  - Zeus botnet: **70$ million** stolen from victim bank accounts
  - → costs: 388 billions $ (source: Symantec 2010)

**And also:**
- more complex attack mechanisms
- more available bandwidth
- more users
- more devices (Internet everywhere)
- more on-line services
Context

- Growth of Internet / network sizes, heterogeneity, mobility
- Continuous arising new threats, high sophistication
- Cyber criminality = new motivations

Network security:
1. prevention / proaction
2. detection
3. reaction

Network security → observations → network monitoring
Multiple infection vectors: direct attack, email, pdf, instant messaging, social networks

Distributed attacks (botnet → DDoS, spam,...)
  - Multi-hops attacks
  - Enhancement of malware robustness: fastflux, double-flux
Challenges:

- local view inefficient against distributed attacks → collect global and multiple information (network traffic, DNS domains, used applications, etc)
  - detect attacks at the operator levels
  - collect global data about the network from individual location

- scalability: storage and analyze large volume of data (60,000 flows/second, millions of hosts, etc)
  - aggregate information
  - combine individual information = collaborative security
  - distributed computing

- privacy:
  - sensitive information to analyze (user tracking)
  - multiple sources / information sharing
1 Introduction
   Some facts
   Motivation

2 Traffic analysis
   Anomaly detection
   Botnet detection

3 Topology analysis
   Bad behaviors in Internet
   Detection
   Evaluation

4 Conclusion
1. Introduction
   - Some facts
   - Motivation

2. Traffic analysis
   - Anomaly detection
   - Botnet detection

3. Topology analysis
   - Bad behaviors in Internet
   - Detection
   - Evaluation

4. Conclusion
(Net)flow records

- **Condensed** information about a traffic “instance”
  - timestamp, ip src, ip dst, protocol, #bytes, #pkts, etc

- **Advantages:**
  - Widely available at ISP level
  - No payload → privacy preserving

- **Challenges:**
  - Few information
  - Huge volume of data (100 000 flows/second)

- → combine multiple flow records to highlight malicious activities
Aggregation

- **Scalable** way to represent information
  - Outline relevant correlated facts
  - reduce storage needs and post processing time
- **Temporal and Spatial aggregation**
  - temporal: time windows split ($\beta$)
  - spatial: keep nodes with activity $> \alpha$ e.g. *traffic volume*, aggregate the others into their parents $\rightarrow$ needs hierarchical relationships
- **Heterogeneous Data**
  - No specific order
    - 1st Source IP@, 2nd Destination IP@
  - Auto adjust to Information Granularity
    - $/18$, $/24$, $/27$ subnetworks...
### Traffic analysis

#### Anomaly detection

#### Botnet detection

### Topology analysis

### Conclusion

---

**Mutidimensional Aggregation Example**

<table>
<thead>
<tr>
<th>PORT</th>
<th>PROTO</th>
<th>KB</th>
<th>TIME</th>
<th>SOURCE</th>
<th>DEST</th>
</tr>
</thead>
<tbody>
<tr>
<td>80</td>
<td>TCP</td>
<td>1491</td>
<td>2010-02-24 02:20:15</td>
<td>192.168.6.2</td>
<td>92.250.221.82</td>
</tr>
<tr>
<td>110</td>
<td>TCP</td>
<td>988</td>
<td>2010-02-24 02:20:19</td>
<td>192.168.8.2</td>
<td>92.250.223.87</td>
</tr>
<tr>
<td>443</td>
<td>TCP</td>
<td>902</td>
<td>2010-02-24 02:20:27</td>
<td>192.168.11.2</td>
<td>92.250.220.82</td>
</tr>
<tr>
<td>110</td>
<td>TCP</td>
<td>1513</td>
<td>2010-02-24 02:20:29</td>
<td>192.168.112.1</td>
<td>92.250.222.81</td>
</tr>
<tr>
<td>80</td>
<td>TCP</td>
<td>1205</td>
<td>2010-02-24 02:20:29</td>
<td>192.168.11.1</td>
<td>92.250.220.82</td>
</tr>
<tr>
<td>80</td>
<td>TCP</td>
<td>1491</td>
<td>2010-02-24 02:20:31</td>
<td>192.168.1.2</td>
<td>92.250.220.83</td>
</tr>
<tr>
<td>110</td>
<td>TCP</td>
<td>1467</td>
<td>2010-02-24 02:20:39</td>
<td>192.168.12.2</td>
<td>92.250.221.81</td>
</tr>
<tr>
<td>80</td>
<td>TCP</td>
<td>927</td>
<td>2010-02-24 02:20:39</td>
<td>192.168.12.2</td>
<td>92.250.220.82</td>
</tr>
<tr>
<td>443</td>
<td>TCP</td>
<td>1294</td>
<td>2010-02-24 02:20:39</td>
<td>192.168.11.1</td>
<td>92.250.223.82</td>
</tr>
<tr>
<td>110</td>
<td>TCP</td>
<td>940</td>
<td>2010-02-24 02:20:49</td>
<td>192.168.21.2</td>
<td>92.250.221.81</td>
</tr>
<tr>
<td>80</td>
<td>TCP</td>
<td>917</td>
<td>2010-02-24 02:20:49</td>
<td>192.168.23.1</td>
<td>92.250.220.82</td>
</tr>
</tbody>
</table>
Mutidimensional Aggregation Example

- Previous approach:

  0.0.0.0/0 4.91%
  96.0.0.0/3 5.09%
  101.0.0.0/8 5.00%
  144.115.176.0/20 5.02%
  144.0.0.0/4 7.01%
  144.213.132.0 / 22 5.36%
  101.138.64.0/20 6.86%
  101.176.128.0/19 5.18%
  101.138.74.115/32 5.13%
  An end-host
**Mutidimensional Aggregation Example**

- **app**: mail
  - **src_ip**: next_bit(17,32)
  - **dst_ip**: next_bit(17,32)

- **Destination port**
- **Source IP**
- **Destination IP**

- **Mutidimensional Aggregation Example**

- **app**: ROOT
  - **src_ip**: 192.168.0.0/17
  - **dst_ip**: 92.250.220.0/22
  - 6.91% 100.00%

- **app**: $.v3.Pop.Get.Mail.ROOT
  - **src_ip**: next_bit(17,32)
  - **dst_ip**: next_bit(17,32)
  - **app**: SAME

- **app**: HTTP.Web.ROOT
  - **src_ip**: 192.168.6.2/32
  - **dst_ip**: 92.250.221.82/32
  - 10.97% 10.97%

- **app**: $.Secure.HTTP.Web.ROOT
  - **src_ip**: 192.168.11.1/32
  - **dst_ip**: 92.250.223.82/32
  - 9.52% 9.52%

- **app**: $.HTTP.Web.ROOT
  - **src_ip**: 192.168.8.0/21
  - **dst_ip**: 92.250.220.82/32
  - 15.68% 15.68%
**Tree based structure**: Root node and multiple children

**Directions**

- How to find the right path to insert a node within a tree?
- Direction function
  - Most specific ancestor common ancestor between two nodes
  - Longest common prefix match
- IPv4: binary function (0,1) as next bit value
- DNS: every level name is a direction
- ports: service taxonomy
Aggregation

- From leafs to root node
- On a complete tree of a time window
- → Large data structures in memory before aggregation

**Online Strategies (before the end of the time window)**

- Tree size > MAX NODES → aggregation

<table>
<thead>
<tr>
<th></th>
<th>Root</th>
<th>LRU</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Aggregation is triggered from root node</td>
<td>Aggregation is triggered in the least recently used node</td>
</tr>
<tr>
<td>RAM</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Performance</td>
<td>- -</td>
<td>-</td>
</tr>
</tbody>
</table>
Datasets

- Real ISP data + attack injection

<table>
<thead>
<tr>
<th># Flows</th>
<th>3,907,859</th>
</tr>
</thead>
</table>
| # IP Addresses | source addresses: 250,314  
destination addresses: 235,120 |
| # bytes      | 24.1 GB   |
| Avg. bytes/Flow | 6,829 |
| # Packets    | 38,132,130 |
| Avg. Packets/Flow | 9.76 |
| # UDP Flows  | 2,756,321 |
| # TCP Flows  | 1,097,030 |
| # ICMP Flows | 50,914 |
| # Other Protocol Flows | 3,594 |
Introduction
Toposlogy analysis

Traffic analysis

Anomaly detection
Botnet detection

Conclusion

Results

- Source and destination IP address + distance → decision tree
- average tree size = 3288, 90 (after aggr.)

<table>
<thead>
<tr>
<th>Type of Attack</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>TPR</td>
</tr>
<tr>
<td>Nachi scan</td>
<td>0.912</td>
</tr>
<tr>
<td>Netbios scan</td>
<td>0.941</td>
</tr>
<tr>
<td>Popup Spam</td>
<td>0.882</td>
</tr>
<tr>
<td>SSh scan + TCP flood</td>
<td>0.882</td>
</tr>
<tr>
<td>DDoS UDP flood</td>
<td>0.923</td>
</tr>
<tr>
<td>DDoS TCP flood</td>
<td>0.887</td>
</tr>
<tr>
<td>DDoS UDP flood + traffic deletion</td>
<td>0.932</td>
</tr>
</tbody>
</table>

- False positive reduction → compare Netflow without aggregation (Networking’11)
- Aggregation → better to detect large scale attacks
Anomaly detection in ISP network

- privacy preserving → Netflow data
- low complexity:
  - LRU algorithm (Least Recently Used) → maximal size fixed to 128
  - usually lower in practice
- Dynamic granularity over the IP address space
  - granularity is guided by the events to monitor...
  - ...not by the size of space to monitor

- tool: https://github.com/jfrancois/mam
- Publications: Networking’11, LISA’12
1. Introduction
   - Some facts
   - Motivation

2. Traffic analysis
   - Anomaly detection
   - Botnet detection

3. Topology analysis
   - Bad behaviors in Internet
     - Detection
     - Evaluation

4. Conclusion
Botnet architecture: Command & Control (C&C) to propagate orders
  - centralized approach (IRC, HTTP)
  - structured P2P botnet: high performance

Detection (state of the art)
  - detect large volumes of related attacks
  - centralized botnets: detect central component
  - P2P botnets: active participation

Objective: passive detection of P2P botnets which do not generate high volume of traffic (data stealing / espionage, stealthy infection)
Contribution

- Discover the C&C channel at the ISP level:
  - NetFlow monitoring $\rightarrow$ who talks to whom? (dependency graph)
  - linkage analysis + clustering techniques $\rightarrow$ identify groups of hosts sharing similar behaviors
  - MapReduce implementation
  - experiments using real NetFlow data
Who talks to whom?
- bots have a distinguishable communication patterns
- bots are well interconnected together

Trivial example: bots = 1, 2, 3, 4

Automatic analysis:
- Local view: node adjacency, benign hosts well interconnected (server)
- Global view: a bot may be connected to few others which are connected to few others and so one + loops → they are globally well interconnected together
Global link analysis
- Google web page ranking algorithm
- A page/host is highly scored if it is well pointed by others especially if these latter have high scores

Iterative computation
- Equal score at the begin
- Stop when stable
- Score propagation
- Weighted nodes (bot knowledge)

\[ P_t(i) = (1-d) \sum_{k=1}^{n} W(k) + d \sum_{(j,i) \in E} \frac{P_{t-1}(j)}{O_j} \]

Both communication directions are important → invert arrows → two values per node: hub, authority
Inefficiency of pure link analysis
- benign hosts may be highly scored (popular services)
- bots → similar communication patterns
- botnet might be partitioned (randomness of connection, disruption)
- simple thresholds not well fitted

Clustering
- find similarly scored hosts
- unsupervised algorithm + few parameters
- DBSCAN: density based
Cluster distinction

- A cluster can be composed of benign hosts → necessary prior knowledge about the botnet:
  - one bot per cluster → all the hosts of the clusters are bots
  - additional tool: honeypot, blacklists, IDS, etc.
Map-Reduce:

- data-intensive processing
- shift the network transfer from the data to the code
- approach based on \( \langle \text{key}, \text{value} \rangle \) pairs:
  - map input: \( \langle k_1, v_1 \rangle \) (\( k_1 \): line number, filename... but rarely used for further usage)
  - intermediate between mappers and reducers: \( \langle k_2, v_2 \rangle \)
  - reduce output: \( \langle k_3, v_3 \rangle \)
- partitioner: \( k_2 \rightarrow \text{Reducers} \)

![Diagram of Map-Reduce with examples](image_url)
Node = ID [key] + (score + adjacent nodes) [value]

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Current Score</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
</tr>
</tbody>
</table>

Map Tasks:
- Mapper
- Mapper
- Mapper
- Mapper

Shuffle and Sort: aggregate values by keys
- Reducer
- Reducer
- Reducer

Reduce Tasks:
- 2 0.3
- 3 0.3, 1
- 4 0.3, 1
Real data issue

- Netflow ISP Data containing labeled botnet C&C traffic → impossible
- Compromise:
  - real data (considered as to being free of botnets)
  - synthetic botnet traffic injected
- P2P botnet traffic → define host relationships:
  - id space: $N = 2^{160}$
  - Chord (DHT) → theoretical but generic: routing in $\log(N)$
  - Kademlia (XOR metric): routing in $O(\log(N))$ but with a high redundancy → high robustness
  - Koorde (sub-partitioning): routing in $O(\log(N)/\log(\log(N)))$ with a low redundancy → less robustness
- **Stealthy** botnets: 1% of IP addresses
- **Bot IP addresses** randomly and uniformly selected

<table>
<thead>
<tr>
<th></th>
<th>chord</th>
<th>Kademlia</th>
<th>Koorde</th>
</tr>
</thead>
<tbody>
<tr>
<td>Flow#</td>
<td>2133887</td>
<td>2399032</td>
<td>1997049</td>
</tr>
<tr>
<td>Host#</td>
<td>323610</td>
<td>323610</td>
<td>323610</td>
</tr>
<tr>
<td>Bytes#</td>
<td>13.7G</td>
<td>13.7G</td>
<td>13.7G</td>
</tr>
<tr>
<td>Duration</td>
<td>18min23sec</td>
<td>18min23sec</td>
<td>18min23sec</td>
</tr>
</tbody>
</table>
Without clustering:
- threshold based method
- threshold varies to compute both true positive and false positive rates

High redundancy → easy detection (Kademlia)

Hub values are more discriminative

FPR = 2% = 6400 FPs → still needed to improve the accuracy
Clustering → better accuracy
- Kademlia: TPR = 99%, FPR = 0.2%
- Koorde: less redundancy → more noise points with DBSCAN → clustering is better before a certain threshold
- Bot knowledge: significant impact only with Chord

- **Kademlia**
- **Koorde**
With Clustering

- Clustering $\rightarrow$ better accuracy
  - Kademlia: TPR = 99%, FPR = 0.2%
  - Koorde: less redundancy $\rightarrow$ more noise points with DBSCAN $\rightarrow$ clustering is better before a certain threshold

- Bot knowledge: significant impact only with Chord

![Graph showing the comparison of clustering with different methods and varying levels of bot knowledge.](image)
- **Unrealistic extrema cases** for detecting all botnets
  - one single cluster $\rightarrow$ huge number of false positives (ROC curves)
  - one cluster per bot $\rightarrow$ all the botnet monitored by the honeypot

- High TPR without one bot per cluster
- Best tradeoff obtaining with few clusters: worst case (Chord): $TPR = 0.96$, $FPR = 0.04$, 21 clusters
Knowledge: 20 bots
Importance of each cluster?
- cluster with few bots
- only needed to monitor huge clusters \(\rightarrow\) limits the knowledge requirements

Kademlia + discard smallest clusters
- low impact on true positives: 92\% with only 2 clusters
- significant reduction of false positives
Efficiency analysis

- score forwarded through the links $\rightarrow$ number of nodes has no impact $+$ number of intermediate (key,value) pairs depends on the number of links
- test different size of dataset $\rightarrow$ subset between 100k and 300M links
- different Hadoop cluster configurations (number of machines)
Efficiency analysis

- **Result**
  - **linear increase** (execution time divided by 7 for a huge dataset)
  - **#links x 10 → execution time x 6** (8 slaves)
  - few links → no improvement due to overhead of Map-Reduce (data split, reduce phase)
  - **< 1M #links → Hadoop useless**
  - **> 10M #links → 4 slaves are useful**
Detection of botnets:
- structured P2P networks
- ISP level / IP flow monitoring (passive approach)
- 2 levels approach: link analysis + clustering
- Some prior knowledge (additional source of information like honeypot)
- Scalability: 18min of monitoring handled in 160 seconds

Future work: how to alleviate the need of a honeypot / relying only on traffic observation → service dependency
1 Introduction
   Some facts
   Motivation

2 Traffic analysis
   Anomaly detection
   Botnet detection

3 Topology analysis
   Bad behaviors in Internet
   Detection
   Evaluation

4 Conclusion
Outline

1. Introduction
   - Some facts
   - Motivation

2. Traffic analysis
   - Anomaly detection
   - Botnet detection

3. Topology analysis
   - Bad behaviors in Internet
   - Detection
   - Evaluation

4. Conclusion
Autonomous Systems

- BGP routing $\rightarrow$ routing table $= AS$ paths (sequence of $AS$ to reach an IP subnet)
- Malware providers needs also hosting (malware, C&C servers, phishing website...)
  - detection: monitoring, complains, reports,...
  - Operators can disconnect/blacklist malware hosters
- $\rightarrow$ some $AS$ are not blocking their malicious users
  - some $AS$ are more tolerant for hosting services (money-driven, political-driven...)
  - malicious entities are their own operators
▶ How to detect AS hosting malware → BGP ranking (http://bgpranking.circl.lu/)
  ▶ ~ ASs administrated by cyber-criminal organization = malicious AS
  ▶ blacklists of IP addresses involved in malicious activities
  ▶ map IP addresses to ASs → compute a score for each AS = detection
  ▶ → neighbor ASs can react (de-peering, complains)

\[
AS_{\text{rank}}(AS_x) = 1 + \sum_{b \in BL} \frac{occ(b, AS_x) \cdot b_{\text{impact}}}{AS_{x\text{size}}}
\]
Avoid detection → hide malicious AS behind ASs looking normal (malware transit AS)
- Complex cyber-criminal organization networks ~ long manual investigation
  - Russian Business Network: 3 years before being disrupted
Contribution

- Detection of malware transit AS not filtering their bad neighbors
  - Accurate AS graph based analysis
  - Global
    - investigation not focused on a single AS
    - not only at the first hop
  - Efficiency = real-time (route stability $\sim 1$ day)
1. Introduction
   - Some facts
   - Motivation

2. Traffic analysis
   - Anomaly detection
   - Botnet detection

3. Topology analysis
   - Bad behaviors in Internet
     - Detection
     - Evaluation

4. Conclusion
BGP routes = who provides transit to whom

Theoretical measure
- extract pair of ASs which are connected through the evaluated AS
- evaluate the potential impact of an AS to another one

\[ MT(AS_x) = \sum_{(AS_y, AS_z)} \frac{(AS_{rank}(AS_y) - AS_{rank}(AS_z))^+}{\text{card}(\{AS_u \in V, AS_y \xrightarrow{AS_u} AS_z\})} \]

Issues
- voluminous number of routes \(\rightarrow\) high complexity
- instability of routes \(\rightarrow\) needs to collect data over long time period to avoid a bias
- \(\rightarrow\) compress routes into an AS graph
- **AS graph → lost of exact transit information**

- **Approximation**: *a malicious AS A can provide malware to AS B through AS C if all paths from A to B goes by C*

- → limit analysis to \( k \) hops around AS B

- **Normalization** regarding the number of neighbors

- **Issue**: single AS analysis

\[
MT'_k(ASx) = \frac{\sum_{(c1,c2) \in pairs(C_k)} \left| \sum_{a \in c1} \text{Rank}_a - \sum_{b \in c2} \text{Rank}_b \right|}{\# \text{neighs}(ASx)}
\]
Global link analysis

- Google: a page/host is highly scored if it is well pointed by others especially if these latter have high scores
- unweighted vs. weighted (BGP ranking)

\[
P_t(i) = (1 - d) \sum_{k=1}^{n} W(k) + d \sum_{(j,i) \in E} \frac{P_{t-1}(j)}{O_j}
\]

- Normalization → average score of ASs having the same number of neighbors

\[
P_t'(i) = P_t(i) - \frac{\sum_{j \in V, \#\text{neighs}(j) = \#\text{neighs}(i)} P_t(j)}{\text{card} \left( \{ j \in V, \#\text{neighs}(j) = \#\text{neighs}(i) \} \right)}
\]
Introduction

Traffic analysis

Topology analysis

Conclusion

Bad behaviors in Internet Detection Evaluation

System overview

- **Input:** BGP announces
  - BGP ranking (additional input/knowledge)
  - AS graph representation $\rightarrow$ graph analysis
Dataset and Methodology

- **Dataset**
  - BGP route announces collected at rrc00.ripe.net (Amsterdam)
  - April 2012, 41k ASs
  - AS paths: 7243k / 1028k (unique)
  - As graph edges: 95k

- **Methodology**
  - no groundtruth
  - → use theoretical estimation = natural definition of malware transit AS
    - cannot be applied to all ASs
    - → check coherency of the output of PageRank-based approach with the theoretical estimation
- A minority of malicious AS…
- … but not blocked

- PageRank-based analysis
  - Damping factor impacts a lot the results
  - variation coefficient \((\sigma/\mu) = 0.41\)
  - Criteria
    - Always in top 30 → Malware Transit AS → 23 AS
    - Always out of top 30 → Normal AS
  - Worst case analysis: normal AS in top 30-100 → 30 AS
Theoretical estimation (single AS) of selected AS
  - Malware transit AS are clearly distinguishable → global analysis is coherent with the natural definition
  - First value (index 0) = BGP ranking
    - no correlation between BGP ranking and the malware transit measure
    - → the malware hoster are not the malware forwarder
  - Malware transit AS
  - Normal AS
Sample topology extraction

- 2 malware transit ASs: T14, T27
- High BGP ranking $\rightarrow$ light color, higher size
Malware transit AS detection

- domain not well covered until now
- graph analysis approach → global analysis + low complexity
- practical validation → famous countries
- publication: IM’13

Future work

- enhanced metric / graph analysis
- time series evaluation
Graph analysis = accurate way to assess security in Internet

- data selection? → what should a graph represent and highlight?
- analysis → more sophisticated method?

Some issues

- algorithm tuning → learning
- datasets
  - real data including various users, services, etc.
  - labeled traffic (attacks)
  - recent
  - → www.caida.org/data/
Scalable Analysis for Network Monitoring and Forensics Purposes

Jérôme François