UNIVERSITÀ DEGLI STUDI DI NAPOLI FEDERICO II

laboratorio nazionale per l'informatica e la telematica multimediali

cini item

High-speed Intrusion Detection in Support of Critical Infrastructure Protection

S. D'Antonio, F. Oliviero, and R. Setola

Rationale

- Nowadays communication networks play a fundamental role in the management of critical infrastructures.
- Command and control messages are exchanged between remote elements of the critical infrastructure by means of a communication network.
- The operation and management of networked infrastructures strongly depend on the correctness and security of the communication network.

Network security

- Control systems for critical infrastructures are rapidly moving from dedicated and proprietary solutions towards IP-based integrated frameworks.
- This trend brings to face security issues since networked systems prove to be exposed to cyber-related threats.
- The most realistic way of addressing network security is to successfully detect intrusions and trace the path leading to the attack source.

Intrusion detection

- To determine whether an intrusion has occurred two main approaches exist:
  - misuse detection
  - anomaly detection
- An Intrusion Detection System (IDS) is the instrument capable to identify inappropriate, incorrect or anomalous activities within a system, be it a single host or a whole network.
- Examples of open-source Intrusion Detection Systems: SNORT and BRO.

Intrusion detection based on user's behaviour (1/2)

- Intrusion Detection Systems represent the main networking application exploiting users' behavioural information in order to detect malicious activities ongoing in the network.
- To effectively face attacks and intrusions such information should also take into account relationships among users sharing common features, resources and purposes.
- Such "inter-user" information can contribute to gain a deeper knowledge of the network context needed to improve the detection process.

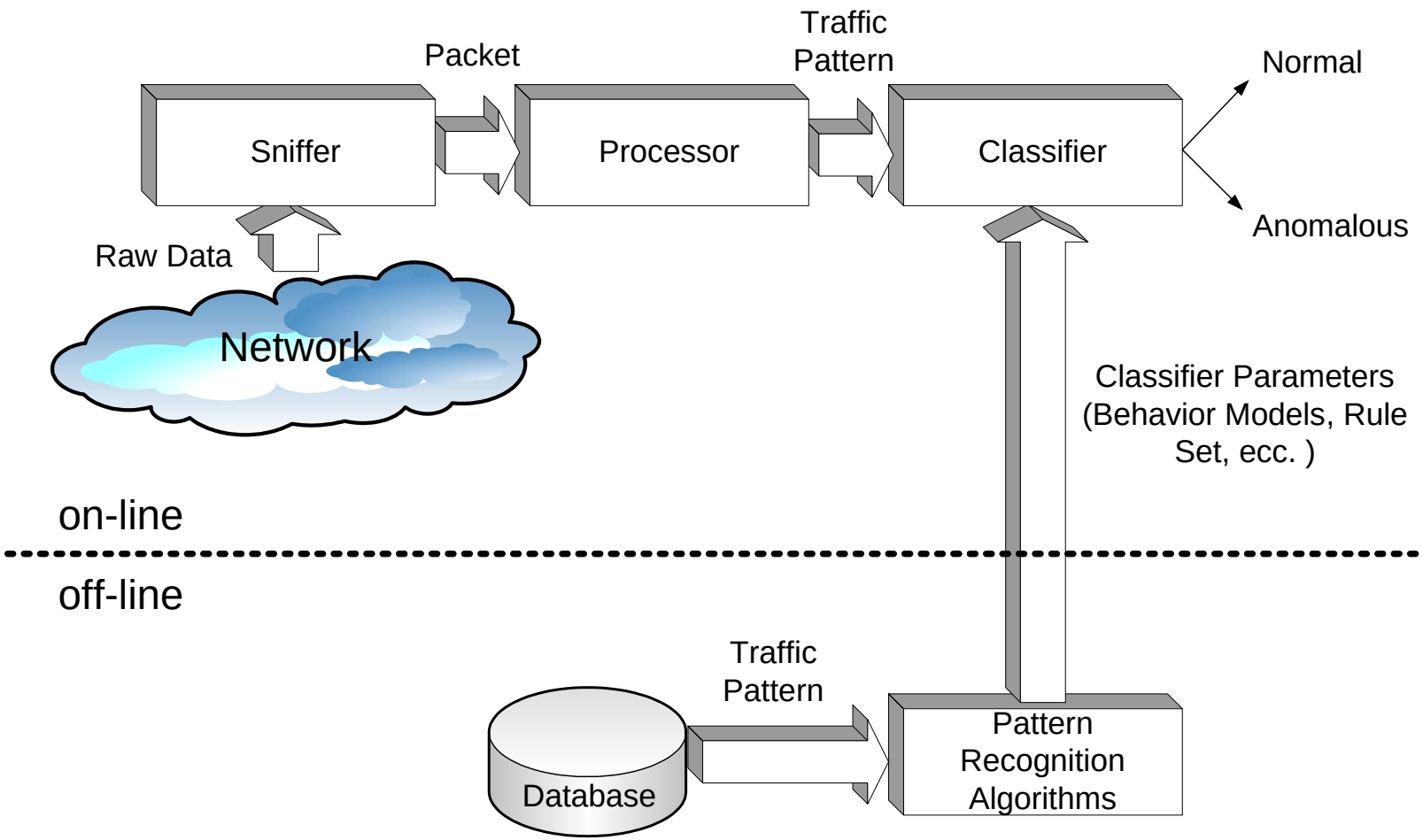
Intrusion detection based on user's behaviour (2/2)

- We developed a security framework relying on two complementary approaches to intrusion detection:
  - a "punctual" classification, i.e. every user's behaviour is classified by means of a specific model.
  - a "non-punctual classification" based on the concept of behaviour generalization, i.e. each classification model generalizes the fundamental properties of a set of user's behaviours. This generalization approach involves methodologies belonging to the data mining research area.

The proposed network security framework (1/2)

- The overall framework is composed of two parts:
  - the real-time Intrusion Detection System consisting of three components:
    - a sniffer,
    - a processor,
    - a classifier.
  - the data mining process, which extracts behavioural models from pre-elaborated network traffic.

The proposed network security framework (2/2)



Two challenging tasks

- Definition of an appropriate set of behavioral models to be used in the intrusion detection process (*off-line task*).
  - Use of data mining techniques,
  - availability of a database of labeled traffic patterns.
- Effective real-time extraction of “user behaviour” from network traffic (*on-line task*).
  - Development of a customizable traffic flow monitoring

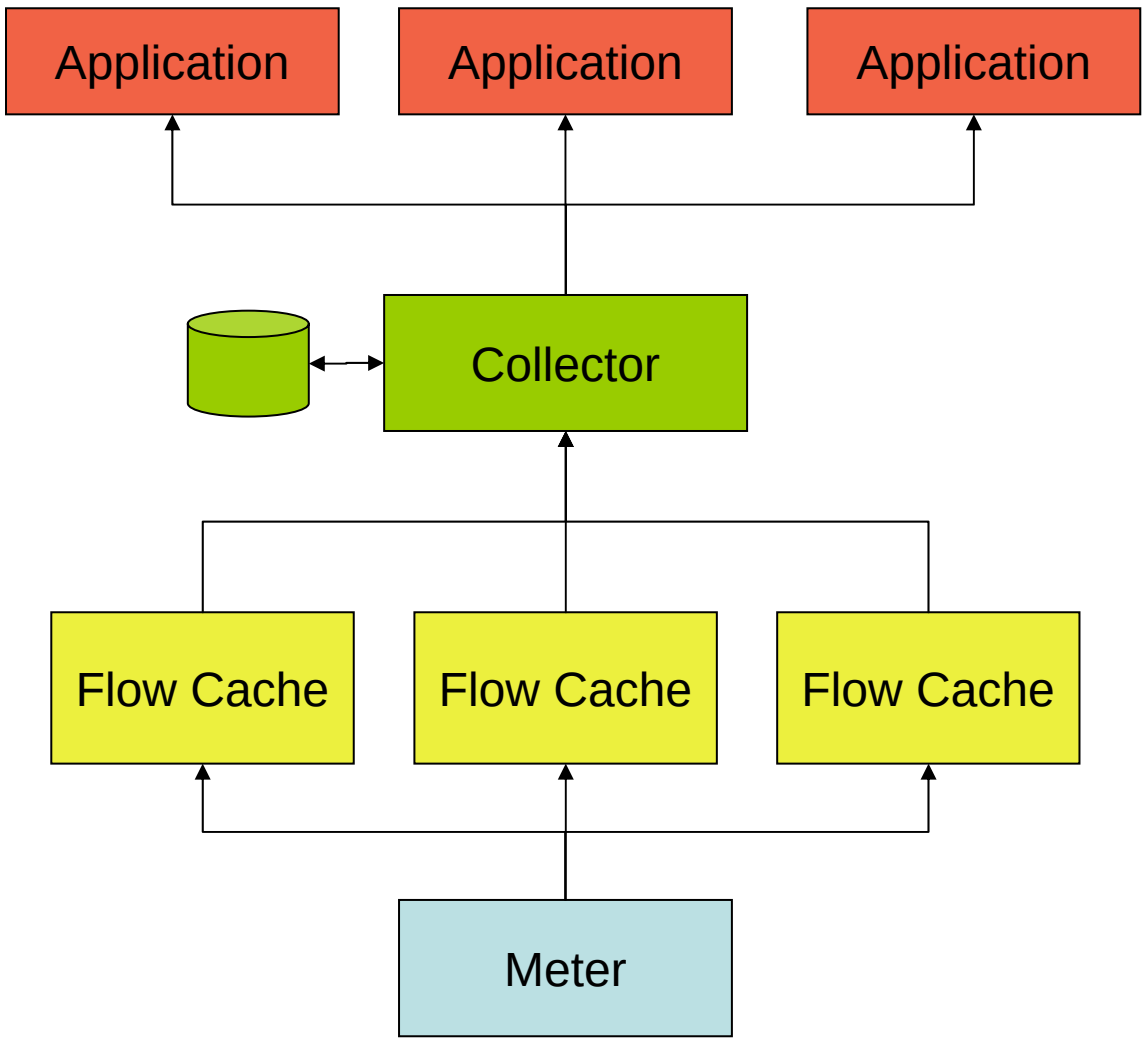
Pattern recognition

- Pattern recognition process operates on a set of data which has been organized in a suitable fashion (e.g. all the data are identified through a label which explicitly specifies the category they belong to).
- Traffic features may be referred to
  - a single packet
  - an entire session which the packet belongs to
  - statistical analysis of the relationships between the current session and the others
- Network features are based on the “connection”
  - UDP and ICMP traffic: single packet
  - TCP traffic: classical meaning

Real-time traffic analysis

- The detection engine runs on top of a customizable flow monitoring system which is responsible for packet capturing and traffic feature summarization.
- Monitoring network traffic entails:
  - capturing packets from an observation point, e.g. an end-system, an access network link, or a backbone link,
  - classifying observed packets in application-based meaningful sets,
  - evaluating appropriate parameters from which user's behaviour can be inferred.

Distributed Flow Monitor (DiFMon)



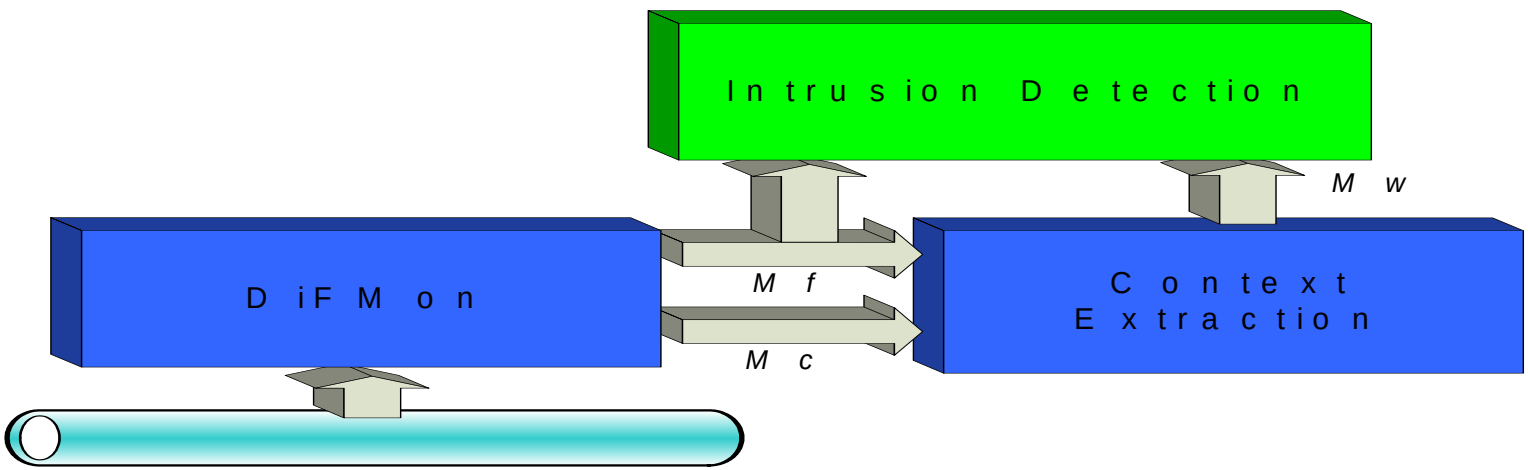
- The meter captures packets from the network and associates them to a flow by enabling a customizable flow definition.
- The flow cache stores and updates flow records.
- The collector exports flow data to the applications in the right format.

The Flow Cache component

- It is the critical module, it must look up and update a flow record as a packet arrives (for this reason it is distributed).
- Packet multiplexing is done by means of a hash function computed on the flow id.
- Metrics can be implemented in a flexible way through an API.
- Flow records are ordered according to the Least Recently Used (LRU) algorithm (on the basis of the last access time).

Context summarization

- M_f = Vector of metrics related to fine-grain flows
- M_c = Vector of metrics related to course-grain flows
- M_w = Vector of metrics describing the overall network context



Context summarization: an example

- The goal is to detect denial of service attacks against a certain server.
- Fine-grain flows are identified through the 4-tuple (source IP address, destination IP address, source port, and destination port).
- Course-grain flow is composed of packets having as destination address the server's IP address.
- M_f (fine-grain) contains the byte count and the byte rate, M_c (course-grain) reports the byte count and the number of packets.
- The context extraction algorithm computes M_w in which the percentages of byte count as well as of number of flows per server port number are reported.
- The detection engine analyses information contained in M_w. If both byte count and the number of flows are high for a specific port number, it can decide to further check data reported in M_f. If a small value of byte rate is found for multiple fine grain-flows having the server as destination, then such situation might be hiding an attack.

Performance evaluation

- Emulated network scenario
  - CPU: Intel Pentium® III 1GHz
  - RAM: 256 MB
  - OS: Mandrake Linux 9.1
  - Traffic rate: 20 pps
- Real network scenario
  - CPU: Intel Pentium® IV 2 GHz
  - RAM: 512 MB
  - OS: RedHat Linux 8.0
  - Traffic rate: 500 pps

	Emulated Traffic	Real Network Traffic
Average CPU Overhead		
Snort™ 2.1.0	0.12%	1.16%
RT-IDS 1.2.0	0.22%	2.42%
Average Memory Overhead		
Snort™ 2.1.0	1.69%	4.99%
RT-IDS 1.2.0	1.70%	9.46%
Packet Loss		
Snort™ 2.1.0	0.39%	0.14%
RT-IDS 1.2.0	0.42%	0.16%

Conclusions and future work

- The IDS represents a significant way to demonstrate how the knowledge of the user's behaviour can prove to be fundamental in order to effectively secure a networking environment.
- The underlying monitoring framework witnesses the importance of making available a large and heterogeneous set of flow information.
- As a future work, significant metrics related to coarse-grain flows have to be defined in order to achieve a more accurate understanding of the network status.
- Furthermore, the scalability of the integrated framework we designed and implemented has to be assessed.
- Finally, a performance comparison in terms of detection accuracy between the proposed framework and other IDS solutions has to be conducted.

UNIVERSITÀ DEGLI STUDI DI NAPOLI FEDERICO II

laboratorio nazionale per l'informatica e la telematica multimediali

cini item

Questions ?

Thank you.

saldanto@unina.it