A Privacy-Aware Access Control Model for Distributed Network Monitoring

Eugenia I. Papagiannakopoulou, Maria N. Koukovini, Georgios V. Lioudakis, Joaquin Garcia Alfaro, Dimitra I. Kaklamani, Iakovos S. Venieris, Frédéric Cuppens, Nora Cuppens-Boulahia

A Privacy-Aware Access Control Model for Distributed Network Monitoring[⋆1]

Eugenia I. Papagiannakopoulou[a,*], Maria N. Koukovini[a], Georgios V. Lioudakis[a], Joaquin Garcia-Alfaro[b], Dimitra I. Kaklamani[a], Iakovos S. Venieris[a], Frédéric Cuppens[b], Nora Cuppens-Boulahia[b]

[a] *School of Electrical and Computer Engineering, National Technical University of Athens, Athens, Greece*
[b] *Institut TELECOM, TELECOM Bretagne, CS 17607, 35576 Cesson-Sévigné, Rennes, France*

Abstract

In this paper, we introduce a new access control model that aims at addressing the privacy implications surrounding network monitoring. In fact, despite its importance, network monitoring is natively *leakage-prone* and, moreover, this is exacerbated due to the complexity of the highly dynamic monitoring procedures and infrastructures, that may include multiple traffic observation points, distributed mitigation mechanisms and even inter-operator cooperation. Conceived on the basis of data protection legislation, the proposed approach is grounded on a rich in expressiveness Information Model, that captures all the underlying monitoring concepts along with their associations. The model enables the specification of contextual authorisation policies and expressive separation and binding of duty constraints. Finally, two key innovations of our work consist in the ability to define access control rules at any level of abstraction and in enabling a verification procedure, which results in inherently privacy-aware *workflows*, thus fostering the realisation of the *Privacy by Design* vision.

Keywords:
Distributed network monitoring, access control, privacy, Privacy by Design, workflows

1. Introduction

Network monitoring is very useful and important for purposes such as network operation, management, planning and maintenance, scientific research based on real traffic traces, law enforcement, as well as the protection of the networks and their users from both accidental failures and malicious activities. Nevertheless, the flip side of monitoring activities is that they are natively surrounded by serious privacy implications, holding an outstanding position among the most "leakage-prone" areas of Information and Communication Technologies, as far as the protection of privacy is concerned. Therefore, as electronic communications proliferate in everyday life, privacy with all its facets is considered as a quality attribute of paramount importance and a salient issue not only for their users–customers but also for the operators that are facing the danger of business losses due to privacy violations; this includes the risk of reputation damage, as well as the potential of sanctions and fines.

Intuitively, since network monitoring depends by default on the collection and processing of information, it raises issues related to the protection of personal data. In fact, the core identification token in data communications is the IP address and, in that respect, the Article 29 Data Protection Working Party has concluded that "... *unless the Internet Service Provider is in a position to distinguish with absolute certainty that the data correspond to users that cannot be identified, it will have to treat all IP information as personal data to be on the safe side*" [1]. Therefore, data gathered through network monitoring are considered as personal data, subject to the data protection legislation [2][3]; IP addresses are not "just numbers" [4].

Even more, network monitoring activities are in particular interesting compared to other domains (e.g., e-commerce), as far as privacy protection is concerned, for a number of reasons:

[1] ⋆This is a revised and expanded version of a paper that appeared in the proceedings of the 4th MITACS Workshop on Foundations & Practice of Security, Paris, France, May 2011, pp. 208–217 [44].

*Corresponding author.
Email address: epapag@icbnet.ntua.gr (Eugenia I. Papagiannakopoulou)

- Privacy-sensitive information is not limited to the payload of the network packets, i.e., the content of the monitored communications. In fact, this case could be even considered as trivial from a privacy protection point of view, since the confidentiality of the content can be adequately achieved by using strong end-to-end encryption. The focus of network monitoring is on the collection of so-called context data [5]. In [6] such data have been characterised as "semi-active", in the sense that data collection occurs transparently for the user; this type of implicit collection tends to raise greater privacy concerns [7].

- While the various protocols' headers already reveal much information (e.g., a visited web-site or the peers of a VoIP call), a huge amount of personal information can be further extracted from their processing, even if they have been anonymised. Once a flow can be examined in isolation, fingerprinting techniques allow to derive personal information from as little as the basic statistics (e.g., packet sizes and inter-arrival times correlation) of the delivered packets [8][9][10][11]. Danezis [12] and Sun et al. [13] have demonstrated that SSL/TLS does not resist statistical traffic analysis, while meaningful data can be extracted even from "unsuspicious" header fields, such as the IP ID field alone [14].

- The network monitoring activities, as well as the underlying categories of data, have been the subject of specific regulations, while, in many countries, independent protection authorities regulate and audit privacy protection in communications. Thus, the legal framework should be taken into account when designing data protection systems and specifying the policies governing their operation [2][3].

- Network monitoring deals with very high data rates that exceed the order of Gbps; thus, any mechanisms deployed for privacy preservation should be able to cope with such stringent requirements.

- Finally, there is an emerging trend of *collaborative monitoring*, reflecting the cooperation between different stakeholders, especially network operators, in order to effectively cope with current attacks threatening networks, such as botnets and large scale DDoS attacks. Beyond the necessity of applying trust and reputation systems [15], additional challenges and privacy implications must be taken into account, since collaboration implies the danger of data circulation across different administrative domains, as well as the potential of data linking and combination.

In light of these issues, this paper describes an innovative access control model, developed in the frame of the FP7 ICT project DEMONS[2], aiming at introducing privacy awareness into distributed network monitoring. Conceived on the basis of distributed network monitoring operational scenarios and the data protection legislation, the proposed model fosters the realisation of the *Privacy by Design* vision [16], by driving the automatic verification of network monitoring workflows' compliance with the privacy principles and their enhancement with privacy features already at design-time. The proposed access control approach relies on a rich in semantics Information Model that captures all concepts related with network monitoring and upon which the access rules are specified. The rules, along with the detailed interrelations of the monitoring concepts, enable the evaluation of the workflows and their appropriate modification at design-time, taking into consideration a variety of parameters, including access rights, purpose, Separation of Duty (SoD) and Binding of Duty (BoD) constraints, dependencies between actions and contextual variations. This way, it provides for the specification of monitoring workflows that are inherently privacy-aware.

1.1. Related Work

Privacy protection in network monitoring is typically thought of as the anonymisation of traffic traces, an area where several works have been proposed [17][18][19][20][21]. Although such frameworks are aimed to be quite generic, a significant drawback is that they base on quite "static" anonymisation policies specification; in all cases, the policies that will regulate the execution of the underlying anonymisation APIs must be defined in an explicit manner. Additionally, although they work well for applications using previously collected traffic data, they are not applicable to applications' domains that demand real-time data, such as intrusion detection systems, while being vulnerable to attacks able to infer sensitive information [22][23].

[2]DEcentralized, cooperative, and privacy-preserving MONitoring for trustworthinesS (DEMONS), homepage: http://fp7-demons.eu/.

Privacy-aware access control has recently evolved to a prominent research area [24]; this trend typically concerns the enhancement of traditional Role-Based Access Control (RBAC) models [25] in order to incorporate different criteria in access control decisions, rather than just *which user*, having *which role*, is performing *which action* on *which data object*. A common characteristic of all models falling into this category is that a central role is held by the concept of *purpose* for which personal information is collected and/or being processed. In this context, various works apply purpose-aware transformations of queries requesting data from relational databases [26][27][28][29]; however, since they target stored data, such approaches cannot be suitable for real-time monitoring systems.

Research has also been driven by the idea of integrating access control with privacy policies, in the sense that the former should ensure the enforcement of the latter. However, approaches such as [30][31][32] have not been designed for meeting the particular requirements of network monitoring and for modelling the corresponding functionalities and infrastructures; additionally, they either do not support context-awareness or they only support some straightforward contexts. Furthermore, they are not suitable for highly dynamic and distributed environments and –especially– for automating privacy-awareness (both in software and hardware scenarios [33]).

In distributed network monitoring environments, access control has to be coordinated across multiple components interoperating for the fulfilment of complex goals that include the detection and mitigation of attacks; in such environments, the automation of privacy provisions enforcement becomes a critical issue for the effectiveness and consistency of the associated procedures. In this context, work in the area of access control enforcement in workflow management [34][35] and Model-Driven Security [36][37], though important, suffer from enforcing security policies only at run-time and not during the workflow formation.

Finally, it should be mentioned that the proposed approach draws inspiration from our previous works, notably OrBAC [38][39][40] and PRISM [41][42][2]. OrBAC provides a mature framework for managing contextual policies, and several extensions, e.g., for dynamically deploying security policies [40], whereas a privacy-enhancing version of OrBAC is presented in [43]. However, the contextual representation of privacy concepts decreases the flexibility of the resulting model, since it limits the integration of complex structures of actions (e.g., paths or subworkflows) that may benefit from automatically being evaluated at design time. On the other hand, PRISM is an access control and authorisation approach specifically devised for network monitoring. Its basic limitation is that it only applies to single-probe environments, being unsuitable for distributed monitoring workflows.

1.2. Contribution and Structure of the Paper

Identifying the afore-described limitations of the related work in relation with the particular characteristics of distributed network monitoring, this paper proposes an innovative policy-based access control model aiming at addressing the underlying issues and introducing privacy awareness to network monitoring. The proposed approach, initial ideas of which were presented in [44], provides a manifold of contributions to the associated research fields that are summarised in the following.

An important feature of the proposed approach is that it relies on a semantically rich information model that captures a variety of network monitoring concepts and the relations between them. This includes concepts that have not been met in other works (e.g., operation containers), as well as relational structures that have hardly been considered yet, such as AND- hierarchies for roles. Moreover, it is ultimately grounded on the requirements stemming from the elaboration of legal and regulatory provisions regarding data protection [2]. This is reflected by the concepts included in the model, as well as the access control rules, the design pattern of which fosters the *by design* realisation of the fundamental principles of necessity, proportionality, adequacy, minimisation and access limitation.

Something that in particular characterises the proposed access control model is that it considers both *concrete* and *abstract* levels for the participating entities' representation. While in traditional access control models the rules are structured over pure concrete entities and state-of-the-art approaches typically follow the RBAC abstraction of role or rules built on fully abstract entities, the proposed model provides flexibility to express concepts and rules at any level of abstraction: only abstract, only concrete, or mixed.

The proposed approach is workflow-oriented, fostering effective management of distributed activities execution. In this context, access control provisions are decided and enforced at the workflow level, addressing the constraints stemming from the interrelations between actions. That is, decisions are not taken considering the actions "in isolation", but taking into account the operational and data flows, resulting in a holistic view of access control across the processes. This way, it is made possible to support a *Verification Procedure* that analyses the workflows at design

time as far as their compliance with privacy requirements is concerned and proceeds with the necessary transformations; relying on knowledge derived from the access control rules, the procedure finally results in workflows that are inherently privacy-aware.

A contribution of our model that should not be neglected concerns the means it offers regarding the specification and enforcement of Separation of Duty (SoD) and Binding of Duty (BoD) constraints. Going beyond the typical guarantees that users are not given enough privileges to misuse the system on their own, the proposed model's structure of rules extends the concepts of exclusiveness and binding and applies them to all elements comprising an action, including the actor, the operation and the resource.

The proposed framework targets primarily privacy protection in network monitoring, on the basis of which it has been conceived. In that respect, it considers the environment of an operator that monitors its IP network for security, management and maintenance reasons. Nevertheless, it should be stressed that the approach aims at being quite generic and applicable in a variety of domains. The access control model itself incorporates all aspects present in the state-of-the-art, thus being able to operate in the majority of use cases. Going beyond, the focus on workflows makes the model promising also for other types of networks; for instance, cases under consideration due to their privacy issues include vehicular ad hoc networks (VANETs) [45] and Content Delivery Networks [46]. Furthermore, a challenge for future investigation concerns to leverage the mechanisms for workflows verification and transformation for being applied in other systems performing structured information processing that can be modelled as workflows, as in the case of multi-user detection in communication systems [47][48].

The rest of this paper is organised as follows. Section 2 describes the basics of the considered framework for network monitoring, including the architecture of the monitoring infrastructure and the procedure for workflow verification. Moreover, it sketches a reference example that will be leveraged throughout the paper for illustrating key concepts. Section 3 delves into the details of the Information Model that constitutes the basis for the definition of actions, tasks and workflows –documented in Section 4– and the access control rules, that are described in Section 5. Sections 6 and 7 present, respectively, the primitives for the inheritance of rules across the graphs of the Information Model and the specification of SoD and BoD constraints by the proposed model. Before concluding the paper, Section 8 deals with discussing how the approach fosters privacy awareness and achieves its goals.

2. Reference Framework for Network Monitoring

The proposed framework relies on a service-oriented, modular architecture providing the abstraction of the underlying infrastructure. In this context, network monitoring processes are specified (and executed) as *Workflows*, consisting of *Tasks* that implement *Actions*. The latter reflect a structure similar to the *subject–verb–object* linguistic pattern and refer to situations where an *operation* is performed by an *actor* on a *resource*. Different types of entities may play the role of an actor, including human users and functional components; similarly, resources may refer to several entities, such as data and system components, while operations comprise by themselves a complex universe of possible functions, characterised by interrelations between its members (cf. Section 3.3).

As illustrated in Fig. 1, a workflow's lifecycle is divided into two phases, notably *Planning* and *Execution*, referring, respectively, to the specification and execution of the workflow. During the Planning Phase, the workflow is specified, including all steps for its graphical definition, decomposition to elementary tasks, compliance checking and necessary transformations. On the other hand, the Execution Phase relies on the Planning Phase's outcome and refers to the deployment of the workflow and its consequent execution by the corresponding components. In other words, workflows specified at an abstract level are finally translated into concrete ones, a conceptualisation appearing also in other recent works, like, e.g., in [49] and [50].

An essential part of the Planning Phase is the so-called *Verification Procedure* (cf. Section 2.3), introduced in [51]. During this procedure, the workflow is analysed and decomposed to elementary tasks that correspond to executable actions. This transformation, grounded on the access control model, is ultimately driven by the goal of privacy compliance and, in this context, the necessary checks and, when needed, the appropriate modifications take place, in order for the final executable workflow to be compliant.

Before presenting the details of the access control model in the forthcoming Sections, in the following, an overview of the architecture is provided (Section 2.2), along with a summary of the Verification Procedure (Section 2.3). In order to better illustrate the utility of the proposed model, a reference example scenario is described in Section 2.1.

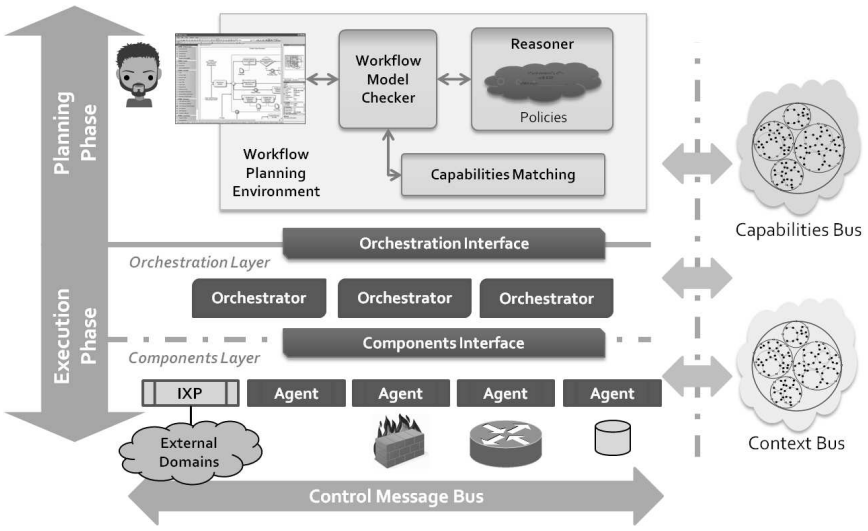


Figure 1: Overall Architecture

2.1. Example Scenario

As a reference example, we consider the case of fast-flux botnet detection and mitigation, which has emerged as malicious flux networks have recently started to thrive [52]. Fig. 2 illustrates the corresponding network monitoring workflow, simplified for enhanced visibility.

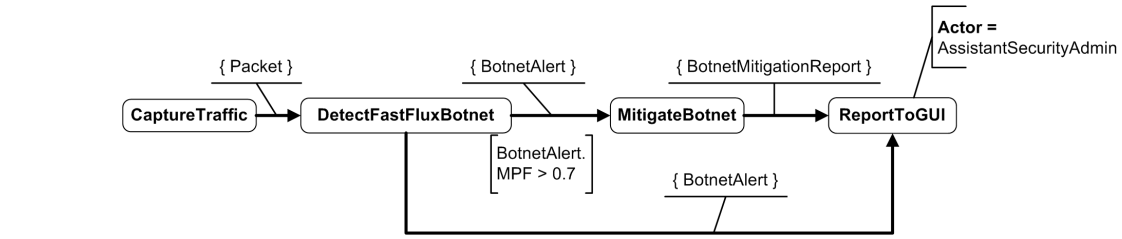
Fig. 2(a) depicts the initial workflow, as specified by its designer. It consists of four tasks, namely `CaptureTraffic`, `DetectFastFluxBotnet`, `MitigateBotnet` and `ReportToGUI`. Its description is rather straightforward: traffic is collected (`CaptureTraffic`) and analysed in order for fast-flux botnets to be detected (`DetectFastFluxBotnet`); in all cases, some results are presented to a GUI (`ReportToGUI`), while in the case of an alert denoting a botnet with considerable Malware Probability Factor (MPF), a mitigation procedure is invoked (`MitigateBotnet`). In the example, the designer has also specified the role that the recipients of the results should hold (`AssistantSecurityAdmin`). The data types foreseen to be exchanged between tasks are enclosed in braces (e.g., `Packet`), whereas the MPF condition is surrounded by brackets: mitigation should take place if the MPF field of the `BotnetAlert` is greater than 0.7.

This workflow is apparently not directly executable; not only it is defined at a high abstraction level and should be transformed so that it contains only concrete elements, but it also needs to be checked regarding compliance with regulatory provisions related with personal data protection and appropriately adapted. In that respect, Fig. 2(b) depicts the workflow after some transformations following the verification and transformation procedure. Further details on the example are provided throughout the following sections.

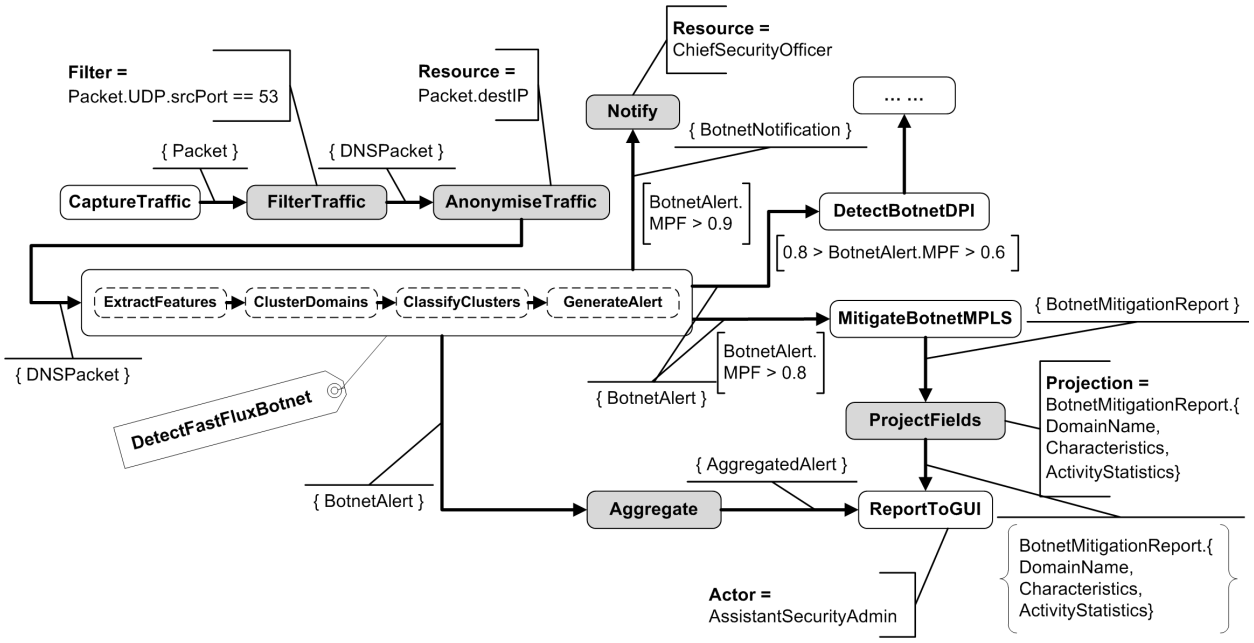
2.2. Overview of the Architecture

As afore-mentioned, a workflow's lifecycle consists of two broad phases: *Planning* and *Execution*. From an architectural point of view, this is translated to two domains, the *Workflow Planning Environment* and the *Execution Environment* (Fig. 1).

The Execution Environment constitutes a conceptual domain that incorporates all entities participating in the workflow execution; it can be seen as an infrastructure of two layers, notably the *Orchestration Layer* and the *Components Layer*. The former provides a pool of *Orchestrators*, each being a stateful component, playing the role of the workflow coordinator throughout its execution. The Components Layer consists of *Agents*, that provide the service abstractions of the underlying actual components; that is, each component (e.g., a detection or mitigation one, a traffic probe, etc.) is associated with an Agent that takes care of control communications, the transformation of the specified behaviour to a Platform Specific Model (PSM), as well as tasks related with context generation, publication, retrieval and evaluation.



(a) Initial Workflow (as specified by its designer)



(b) (Partially) Processed Workflow

Figure 2: Workflow Example

The *Inter-domain Exchange Point* (IXP) constitutes a special type of Agent, being the functional gateway towards external domains, for the realisation of cooperative monitoring. On the one hand, the local domain is enabled to incorporate into local workflows and invoke services that are provided by remote domains; on the other hand, it provides external domains with a single point of access to local services. In both cases, the exposed service is associated with a security policy, reflecting the conditions and constraints under which the service can be accessed.

Apart from the graphical editor for the workflow specification, the Workflow Planning Environment consists of the following high-level components:

- *Workflow Model Checker*: the component responsible for performing all the necessary verifications and modifications of the original workflow, as specified by its designer, towards a sound, consistent and compliant workflow; in the end, it provides the final executable workflow to the Orchestration Layer.
- *Reasoner*: the entity that provides the intelligence to the system, incorporating the access control rules and exploiting the knowledge that is contained therein.
- *Capabilities Matching*: it verifies that the capabilities implied by the workflow can indeed be provided by the underlying infrastructure.

Two essential components of the architecture are the *Context Bus* and the *Capabilities Bus*. The former keeps track of the real-time contextual parameters and events, enabling their efficient circulation. The latter is responsible for

providing information regarding the availability of capabilities (such as functionalities and attributes) of the underlying system. Both components constitute Peer-to-Peer Publish/Subscribe infrastructures enabling the specification of the rendez-vous points for information publication and retrieval, leveraging the semantics of the information itself [53].

A walkthrough of the operational aspects is summarised by the following steps:

(i) The workflow is authored at a high abstraction level. Using the Workflow Planning Environment, the designer visually specifies the workflow by putting "boxes" corresponding to tasks and specifying the control and data flows between them, resulting in a view similar to Fig. 1(a).

(ii) The workflow is analysed by the Workflow Planning Environment regarding its operational consistency, as well as its compliance with the underlying privacy provisions. The analysis concerns the semantic interpretation of the specified operations and their interactions, the matching with capabilities offered by the system, the decomposition to elementary computation tasks and the appropriate modification for achieving consistency and compliance.

(iii) As soon as the workflow is submitted for execution, a free Orchestrator is employed in order to coordinate the execution. The Orchestrator performs a second round of capabilities matching, this time identifying the specific components that are able to fulfil the desired functionalities. The workflow is split into individual components' behaviours and each component is made aware of its mission for the execution of the workflow.

(iv) At the components level, the entities representing the underlying actual components are the Agents. Upon receiving the description of its duties, each Agent performs the transformation to the platform-specific language (e.g., a configuration) and the component starts operating accordingly, serving the assigned tasks and interacting with the peer components. This consists in both control- and data-level interaction, with the former performed through the Agents and the latter through the appropriate data plane IPFIX [54] interfaces.

2.3. Workflow Verification Procedure

Of great importance is the procedure for verifying and appropriately adjusting the workflow at design-time, so that it becomes inherently privacy-compliant before its execution; it consists in three steps ([51]):

- *Purpose verification:* Checks regarding purpose compliance are performed; specifically, in order for a workflow to be privacy-aware, its design must be relevant and consistent with a purpose, while the purpose itself should not contradict with the access rights of the person initiating the workflow.
- *Skin task verification*[3]: Each skin task is evaluated both individually and in relation to the rest of the skin tasks. During these checks, the system may introduce modifications, such as task additions, removals or substitutions.
- *Decomposition*: Each composite skin task is decomposed in more elementary subtasks, until it is finally analysed in a combination of atomic tasks that will eventually be included in the final executable form of the workflow.

The means for performing the checks and making decisions regarding necessary modifications are provided by the access control rules; the Workflow Model Checker queries the Reasoner, which is assigned with the task of extracting knowledge from the rules. Many of the associated aspects are described in the following sections.

[3] We call *skin tasks* the ones defined by the workflow designer, as opposed to the tasks that their inclusion in the workflow is a result of workflow check and modification; all tasks in Fig. 2(a) are considered to be skin tasks. Their separate examination without considering their decomposition targets the early identification of conflicts.

3. Information Model

The proposed access control model considers both *concrete* and *abstract* levels for the participating entities' representation. The concrete level refers to well-specified entities, e.g., named humans, while the abstract level uses abstractions of the underlying concepts, mostly their semantic type or other quality attributes, enabling reference to entities that are characterised by features, such as role or type. This is not a new idea, considering, e.g., the introduction of the *role* abstraction for users by Role-Based Access Control (RBAC) [25]; moreover, the concept shares several goals with other approaches placed in the PBNM (Policy Based Network Management) area [55]. Nevertheless, this approach goes several steps beyond RBAC and other works, mainly by capturing new concepts and by enabling the two levels to coexist in rules and other structures. The main concepts considered by the model are presented in Table 1.

In a typical case and at a concrete level, a set of *Users* (*U*), participating in *Organisations* (*Org*), are –directly or indirectly– using *Operation Containers* (*OpC*), deployed over *Machines* (*M*) and offering *Operation Instances* (*OpI*)[4], in order to act upon *Objects* (*Obj*); the latter refer to everything that is affected by or required for the execution of an action, such as *Data* (*D*) and *Alerts* (*Al*) being collected and/or processed during the system's operation.

At an abstract level, the users are assigned with *Roles* (*R*), their actions instantiate some *Operations* (*Op*) and are performed for fulfilling some *Purposes* (*Pu*). Moreover, data, alerts, organisations, machines and operation containers are characterised by *types*, reflecting the semantic class they fall under; thus, sets of *Data Types* (*DT*), *Alert Types* (*AlT*), *Organisation Types* (*OrgT*), *Machine Types* (*MT*) and *Operation Container Types* (*OpCT*) are defined, respectively. It is noted that in the general case concrete concepts are associated to the corresponding abstract ones through an *isOfType* predicate; for instance, the association of a data item with its data type is achieved by means of the predicate *isOfDataType*(d, dt), where $d \in D$, $dt \in DT$.

Additional elements of the model include *Context* (*Con*), that enables the definition of contextual parameters, as well as *Attributes* (*Att*), that are leveraged for describing properties and characteristics of other elements.

Table 1: Concepts of the Information Model

Abstract Level	**Concrete Level**	**Definition**
Data Types (*DT*)	Data (*D*)	Data being collected and/or processed, organised according to their semantic types
Roles (*R*)	Users (*U*)	Human users assigned with roles reflecting their responsibilities inside an organisation
Operations (*Op*)	Operation Instances (*OpI*)	Operations reflect all actions that can take place in the context of the system's operation
Operation Container Types (*OpCT*)	Operation Containers (*OpC*)	Components or other functional structures that typically offer a set of operations together
Machine Types (*MT*)	Machines (*M*)	Hardware components hosting operation containers
Organisation Types (*OrgT*)	Organisations (*Org*)	The various domains within which actions are performed
Alert Types (*AlT*)	Alerts (*Al*)	A special category of data for communicating events
Context (*Con*)	Context values	Real-time parameters and events
Purposes (*Pu*)	(No concrete representation)	Purposes for which access to resources is requested
Attributes (*Att*)	Attribute values	Characteristics further describing members of the other sets

While most of these notions are either typically present in state-of-the-art models or intuitively self-explained, a few remarks are deemed necessary for some of them. Specifically, the *OpC* and *OpCT* are introduced in order to model components or other functional structures that typically offer a set of operations together. For instance, an `IntrusionDetectionSystem` clusters several operations related with intrusion detection. Apart from the convenience it introduces regarding several modelling aspects (such as the inheritance of attributes), these structures are also helpful for describing a variety of concepts related with "horizontal" dependencies and transfer of characteristics. Moreover, the machines play a fundamental role in network monitoring and, therefore, our model cannot be limited to

[4]In Web Services' terms, Operation Containers correspond to a service `interface`, whereas Operation Instances represent the associated `operations`.

a level of abstraction exclusively centred around functionalities; in any case, functionalities are provided by machines which, on the one hand, are characterised by attributes (e.g., topological ones) that may be inherited to the hosted functionalities and, on the other hand, create inherent dependencies between the hosted functionalities. Finally, organisations are explicitly modelled at both the abstract and concrete levels, because we have considered the case that, within a unified model, similar rules may be differently defined for heterogeneous organisation types. For instance, within an organisation of type `BillingServiceProvider`, authorisations regarding the access to customers' data will not be the same with the ones defined for the same data within an organisation of type `TelecomOperator` being the Data Controller entity. Even more, an organisation that participates in cross-domain collaborations will likely define different authorisations for different types of third-party organisations based on their type.

All concepts shown in Table 1 comprise graphs of elements that are characterised by relations; the latter are implemented by predicates defining AND- and OR- hierarchies and enabling the inheritance of attributes and rules, as well as the specification of dependencies. For instance, and with respect to the *DT* graph, three partial order relations are defined: $isA(dt_i, dt_j)$, $lessDetailedThan(dt_i, dt_j)$ and $isPartOf(dt_i, dt_j)$, where $dt_i, dt_j \in DT$, reflecting the particularisation of a concept, the detail level and the inclusion of some data types to another, respectively. Some indicative relations forming hierarchies among the members of a set are presented in Table 2. It should be noted that the $inheritsFrom(m_i, m_j)$ predicate reflecting the inheritance of characteristics, such as attributes and rules, is implied by the other predicates and it enables the definition of rules at high abstraction levels, thus resulting in reduction of the number of policies; for instance, policies regulating the access to data can be defined over *generic* data types and then propagated to the more specific ones; or composite data types may inherit policies explicitly defined for the included ones (cf. Section 6).

Moreover, the model specifies the necessary predicates in order to link concepts from different graphs; for example, the predicate $mayActForPurposes(r, \langle pu \rangle^k)$, where $r \in R$, $\langle pu \rangle^k \subseteq \mathcal{P}(Pu)$, indicates the legitimate purposes $\langle pu \rangle^k$ for which the users assigned with the role r may act.

Table 2: Basic relations between m_i, m_j members of a set

Predicate	**Description**	**Appears in sets**	**Example**
$isA(m_i, m_j)$	Expresses the particularisation relation	*DT*, *AIT*, *R*, *Op*, *OpCT*, *MT*, *Pu*	isA(`ChiefSecurityOfficer`, `SecurityOfficer`), $m_i, m_j \in R$
$isPartOf(m_i, m_j)$	Expresses the inclusion relation	*DT*, *AIT*, *R*, *Op*	$isPartOf$(`ClassifyClusters`, `DetectFastFlux-Botnet`), $m_i, m_j \in Op$
$lessDetailedThan(m_i, m_j)$	Expresses the detail level of a concept	*DT*, *AIT*	$lessDetailedThan$(`IPv4AddressNetworkID`, `IPv4-Address`), $m_i, m_j \in DT$
$inheritsFrom(m_i, m_j)$	Implicit relation expressing the inheritance of characteristics and authorisations	*DT*, *AIT*, *R*, *Op*, *OpCT*, *MT*, *Pu*	$inheritsFrom$(`MitigateBotnetMPLS`, `Mitigate-Botnet`), $m_i, m_j \in Op$, for positive authorisations (cf. Section 6)

In the following, some of the presented concepts are described in more detail, in order to highlight the core aspects of the proposed approach related to privacy-awareness.

3.1. Data and Data Types

Network monitoring depends by default on the collection and processing of *data* that are communicated over the underlying network links. Thus, data comprise a core concept within network monitoring infrastructures. Data are characterised by a *data type*, that is, the semantic class they fall to. In fact, such semantics of the data items play a critical role on how the data should be processed, as well as on the enforcement of provisions related with privacy. In that respect and as afore mentioned, two sets were defined; *Data* (*D*), representing the data at a concrete level, and *DataTypes* (*DT*), reflecting the semantic types of data at an abstract level.

The different data types are organised by means of the $isA(dt_i, dt_j)$, $lessDetailedThan(dt_i, dt_j)$ and $isPartOf(dt_i, dt_j)$ predicates of Table 2 that define transitive and anti-symmetric partial orders of the data types and reflect, respectively, the particularisation of a concept, the detail level and the inclusion of a data type to another. Regarding the $lessDetailedThan(dt_i, dt_j)$ predicate, it allows for effectively tuning the accuracy of disclosed data, in order to meet the so named "proportionality principle", which requires that the personal and business data may be gathered and processed

only to the extent that they are adequate, relevant and not excessive if compared with the monitoring function for which are collected by the system. For instance and with respect to Fig. 2(b), the following apply:

- *isA*(`DNSPacket`, `Packet`)
- *isPartOf*(`DomainName`, `BotnetMitigationReport`)
- *lessDetailedThan*(`AggregatedAlert`, `BotnetAlert`)

Regarding inheritance of attributes, each particularised data type has all the characteristics of the generic data type, plus additional characteristics that make it special. For instance, a data type may be complemented by an attribute indicating its owner or its storage period; this attribute is inherited to all the particularised data types. Inheritance of attributes can be inferred also for data types interrelated through *isPartOf* and *lessDetailedThan* relations, depending though on the nature of the corresponding attribute.

A special category of data concerns alerts, which are of particular importance in the domain of network monitoring; they often serve as the means for communicating events, while being also subject to controlled access. First, not everybody has access to all the alerts; this is highlighted especially in the inter-domain scenario where alerts containing sensitive business information may travel from one organisation towards others. Second, the alerts themselves may directly contain personal information. In that respect, *AlertTypes* (*AlT*) are modelled as a sub-category of *DT*. The structure of an alert is illustrated through the *isPartOf*(*dt*, *alT*) predicate; based on the IDMEF format [56] for object-oriented representation of alerts, an alert type contains various information, such as `AlertCreationTime`, `Alert-DetectionTime`, `AlertTargetUser`, `AlertTargetOperation`, `AlertSourceNode`, being modelled as data types.

3.2. Users and Roles

Evidently, humans constitute a pervasive link in the operational chain of network monitoring systems; regardless the degree of automation of a procedure, there is always some human that designs, initiates, executes a workflow and/or is in charge of the underlying software and/or hardware components. On the other hand, users are assigned with roles, reflecting, e.g., their responsibilities inside an organisation, through the following predicate, meaning that the user u is assigned with the roles $\langle r \rangle^k$:

- *assignedWithRoles*(u, $\langle r \rangle^k$), where $u \in U$ and $\langle r \rangle^k \subseteq \mathcal{P}(R)$.

As illustrated in Table 2, roles are also organised by means of the *isA* and *isPartOf* relations, reflecting particularisation and membership to a complex role, respectively; the second one models cases such as the participation of a role `NetworkAdministrator` in the Computer Security Incident Response Team (CSIRT) [57], semantically defined as `CSIRT`.

Inheritance of authorisations through the *inheritsFrom* relation is examined in Section 6. Moreover, roles, apart from explicitly defined attributes, acquire all the attributes of their ancestors in the role hierarchies; the `Employee` role may bear an `att_Schedule` attribute, which is consequently inherited to the `NetworkAdministrator` role as the latter relates to the former role through the *isA* predicate. In the case of *isPartOf* relation, some attributes of the composite role are inferred from the member roles' corresponding attributes, with e.g., the `CSIRT` role acquiring some characteristics of its member `NetworkAdministrator` role.

3.3. Operations

Operations reflect all actions that can take place in the context of the system's operation, constituting the "heart" of the *Action* structure as it will be described in Section 4. There exist also cases where operations are treated as subjects of actions, especially within highly automated systems like the one of the reference framework, where operations trigger the execution of others. For instance, a DDoS detection operation passes the execution token to a DDoS mitigation one, when a DDoS attack is observed (e.g., the enforcement of a null-routing policy reconfiguration [58] in order to automatically react against the detected attack).

Operations comprise a set of semantically defined elements (*Operations* (*Op*) set) and are characterised by different granularities; starting from very fundamental processing units that can be characterised as *atomic*, functional components can be organised in different structures and create complex compositions that can go up to a very high

level and represent generic tasks such as *intrusion detection* or *DDoS mitigation*, thus providing convenient abstractions of the underlying complex compositions. In other words, operations form hierarchies that reflect the different levels of granularity, based on the semantic definitions of the operations themselves. The *isA* predicate reflects an OR relation indicating different alternative operations that implement the same operation, while the *isPartOf* one comprises an AND relation, requiring all the low-level operations to take place in order for the high-level operation to be fulfilled.

For example, the meaning of the following two statements is, respectively, that `DetectFastFluxBotnet` is a way of doing `DetectBotnet` and that `ClusterDomains` is one of the computational steps of `DetectFastFluxBotnet`.

- *isA*(`DetectFastFluxBotnet`, `DetectBotnet`)
- *isPartOf*(`ClusterDomains`, `DetectFastFluxBotnet`)

However, in order to draft a workflow, having as a starting point the participating operations, these OR and AND relations do not suffice. The first step for this purpose is to define a sequence of operations offering the required functionality as a complex operation. Although this resembles the previously described AND hierarchy, in fact it is a combination of OR and AND relations introducing sequence constraints among the participating operations. In other words, a structure of this kind is characterised by a set of participating operations and a set of *Legs* (edges) connecting adjacent operations and modelling the control and data interactions between them (exactly like in the case of workflows, cf. Section 4). Such sequences of operations with well specified control and data flows between them are called *worklets* and describe in detail the workflows that implement some operations. Therefore, the following set is defined, along with the corresponding predicate:

- *Worklets* (*Wl*), the set of predefined self-contained sub-processes that describe the implementation of an operation by lower-level operations.
- *implementsOperation*(*wl*, *op*), where $wl \in Wl$, $op \in Op$.

In other words, a worklet constitutes itself an operation −serving thus the principle of reusability− and may result in the specification of several executable workflows in accordance with the possible authorisations regarding the included operations.

For instance, the `DetectFastFluxBotnet` operation of the corresponding task in Fig. 2(a) is decomposed by means of a worklet consisting of the `ExtractFeatures`, `ClusterDomains`, `ClassifyClusters` and `GenerateAlert` operations; as described in Section 5, they constitute a *path*, due to their sequential execution.

Operations may take as input data to process, alerts, and input parameters. The outcome of the operations may be some output data and/or alerts, or they may result to the execution of some other operation(s). Regarding the input and output data of an operation −covering also the case of alerts, as they constitute a sub-category of data− we can specify two relevant predicates corresponding to the data types that this operation can consume and produce respectively:

- *hasInputData*(op, $\langle dt \rangle^k$), where $op \in Op$, $\langle dt \rangle^k \subseteq \mathcal{P}(DT)$.
- *hasOutputData*(op, $\langle dt \rangle^k$), where $op \in Op$, $\langle dt \rangle^k \subseteq \mathcal{P}(DT)$.

The required parameters for the execution of an operation may be thought of as attributes. For example, one can define an attribute `att_Filter` for further describing the operation `FilterTraffic`.

Moreover, operations are grouped following the mechanism reflected by the concept of *operations' containers*. As afore-mentioned, this refers to components or other functional structures that typically offer a set of operations together. In that respect, the predicate that glues operations with a container type is the following:

- *providesOperations*($opct$, $\langle op \rangle^k$), where $opct \in OpCT$, $\langle op \rangle^k \subseteq \mathcal{P}(Op)$.

So far, operations have been thought of as abstract entities. Nevertheless, operations have their representation at the concrete level, notably by means of *Operation Instances*. Operation instances correspond to *implementations* or *instantiations* of operations by specific functional components. In Web Services' terms, an operation instance is

equivalent to the abstract description of an `operation`, along with its concrete binding information, as derived by the associated operation container and the machine that provides the operation. Assuming, for instance, the operation `AppendFirewallRule` that reflects the functionality of appending one or more rules to a firewall's ruleset chain, the provision of this operation by two different firewalls, Firewall#1, Firewall#2 $\in M$, implies two different elements in the *Operation Instances* (*OpI*) set.

For dealing with operation instances, several predicates are defined, including the ones for associating an operation instance with the operation it instantiates and assigning operation instances to an operation container:

- *instantiatesOperation*(*opi*, *op*), where $opi \in OpI$, $op \in Op$.
- *containsOperationInstances*(*opc*, $\langle opi \rangle^k$), where $opc \in OpC$, $\langle opi \rangle^k \subseteq \mathcal{P}(OpI)$.

Nevertheless, not all operations can have corresponding instances; for instance, operations such as `execute`, `read` and `invoke` are considered only at the abstract level.

It is noted that as operations' containers are hosted in machines, the corresponding predicate is defined, namely *hostsContainers*(*mt*, $\langle opct \rangle^k$), where $mt \in MT$, $\langle opct \rangle^k \subseteq \mathcal{P}(OpCT)$, allowing for the reflection of any effects of an operation performed on a machine to the hosted operations' containers and possibly to the operations they offer.

3.4. Context

In practice and especially within workflows, access control rules remain inactive until a set of conditions are fulfilled, i.e., until the conditions are evaluated and mapped to a truth value. Therefore, we denote as *contextual* the authorisation policies containing dynamic authorisation provisions. The proposed approach exploits the rich expressiveness of the OrBAC Context Taxonomy [39]. In this regard, authorisation rules may depend on temporal contexts (e.g., authorisations granted only during working hours), geographical contexts (e.g., permission inside the physical boundaries of a company), *a priori* contexts (in which a permission to execute a set of actions can only be carried out as a result of the completion of previous actions). Thus, it is essential that not only the contextual conditions are captured by the model, but also that they are taken into consideration during the verification and transformation procedure, providing for the specification of context-based differentiated workflow behaviours, already during workflow formation.

Therefore, the model considers a *Context* (*Con*) definition set; a definition refers, on the one hand, to real-time parameters, such as location or time, while on the other to events, such as system alarms. Context definitions leverage the evaluable conditions required to get a security rule to become applicable. Some examples of contexts are: *Temporal* contexts, which depend on the time at which the subject is requesting access to the system; *Spatial* contexts, which depend on the subject/object location; and *Threat* contexts, which are activated on the reception of a specific alert. Furthermore, structures of contexts are defined using two different relations, creating AND- and OR-trees, respectively. This way, a context $cont \in Con$ can be expressed as the logical OR / logical AND of a set of sub-contexts, respectively, while negative contexts are also possible. Contexts that constitute combinations of sub-contexts are called *composite contexts*, as opposed to the *atomic contexts*.

3.5. Purposes

A very fundamental concept for privacy that must anywise be taken into consideration is that of the *purpose* for which access to resources is requested. Although this could be modelled as a special context (like in [43]), we have chosen to treat purpose as a stand-alone concept, in order to highlight its significance regarding privacy-awareness, as well as distinguish it from real-time parameters and events. Thus, the *Purposes* (*Pu*) set is defined, with its members forming hierarchies by means of OR (*isA*) relations modelling particularisation of a high level purpose to more specific ones; for instance, a purpose `PerimeterSecurity` specialises the more general purpose `NetworkSecurity`.

Intuitively, not all operations can be used for fulfilling some purpose, in the sense that they are not compliant with and consistent to this purpose; this is expressed by means of the following predicate:

- *mayServePurposes*(*op*, $\langle pu \rangle^k$), where $op \in Op$, $\langle pu \rangle^k \subseteq \mathcal{P}(Pu)$.

Through this predicate and in combination with the afore-mentioned OR relation, it is implied that, apart from the explicitly defined purposes, an operation may serve more specific or more general ones. Additionally, all operations related to this by means of particularisation (OR) or inclusion (AND) relations are implied to also serve these purposes; however, as the distance between a child operation and the one which is explicitly bound to a purpose grows, the degree of affinity between the purpose and the child operation becomes weaker. This is particularly visible in cases of trivial operations that may serve almost every purpose. Finally, in order for a series of operations to be included in a worklet, they must be proven to serve adequately at least one common purpose.

Likewise, the observation that not all roles may act for all purposes leads to the definition of the predicate:

- *mayActForPurposes*(r, $\langle pu \rangle^k$), where $r \in R$, $\langle pu \rangle^k \subseteq \mathcal{P}(Pu)$.

It is noted that these two predicates are explicitly specified −while they could just be inferred by reasoning over the access control rules (cf. Section 5)− in order to support the purpose verification step of the Workflow Verification Procedure, allowing for a "quick" check of purpose compliance. In that respect, the *mayServePurposes* predicate allows for checking whether the operations contained in a workflow are in line with the purpose that the workflow is supposed to serve; all the defined functions `CaptureTraffic`, `DetectFastFluxBotnet`, `MitigateBotnet` and `ReportToGUI` of Fig. 2 are relevant to the purpose of `NetworkSecurity`, while a task `RecordTraffic` is out of the scope of the same purpose and would have been rejected. On the other hand, the *mayActForPurposes* predicate serves for answering whether the roles held by the initiator of a workflow justify triggering the execution of a workflow, in order for some specific purpose to be served; for instance, an `AssistantSecurityAdmin` should be able to execute a workflow serving the purpose of `NetworkSecurity`, while an `Accountant` should normally not.

Regarding the compliance check between a role's acting purpose (e.g., a user-declared one) and the one that an operation serves, the following predicate is leveraged:

- *compliantWithPurpose*(pu_r, pu_{op}), where pu_r, $pu_{op} \in Pu$.

3.6. Attributes

As it is already made clear, attributes constitute an essential part of the information model by complementing and further describing the members of the other sets at both abstract and concrete levels. Therefore, the *Attributes* (*Att*) set is defined and its members are formally described through the ordered pair ⟨AttributeName, AttributeType⟩. It is noted that for the name of the attributes, the `att_` prefix is used by convention, in order to avoid naming confusions with the elements of other sets. Regarding AttributeType, this can be some regular type, e.g., boolean, integer, etc., or a member of another set. For example, consider the following attributes:

- ⟨`att_Raw`, `boolean`⟩, denoting whether a data resource is raw, that is, explicitly contained in the monitoring flows (e.g., protocol header fields or any other types of data that can be provided directly), or constitutes the product of some processing function.
  Here the AttributeType is set to `boolean`, implying that the attribute takes a boolean value.

- ⟨`att_NetworkAddress`, `IPv4Address`⟩, where `IPv4Address` ∈ *DT*, characterising some entity, e.g., a network interface, by means of its network address.
  In this case, the `IPv4Address` element of the *DT* set is used as the AttributeType.

The mappings between entities and attributes are achieved through the following predicates:

- *hasAttribute*(*entity*, $\langle at \rangle^k$), where *entity* ∈ $DT \cup Op \cup OpCT \cup R \cup MT \cup OrgT \cup D \cup OpC \cup U \cup M \cup Org$ and $\langle at \rangle^k \subseteq \mathcal{P}(Att)$, or

- *hasAttributeValue*(*entity*, *at*, *value*), including also the value of the associated attribute, where the value must be consistent with the attribute's type.

For example:

- *hasAttribute*(`IPv4Address`, `att_Raw`), where `IPv4Address` ∈ *DT*, `att_Raw` ∈ *Att*.

- *hasAttributeValue*(*pc*, `att_NetworkAddress`, 192.168.1.1), where $pc \in M$, `att_NetworkAddress` $\in Att$ and 192.168.1.1 $\in D$.

It is noted that the attributes' value can be defined at both abstract and concrete levels. Moreover, attributes may be inherited from some entity to others through the various types of relations. However, there may exist attributes that should not be inherited, a case anticipated by defining a boolean propagation-related attribute on the attribute itself.

4. Actions, Tasks and Workflows

All the afore-described entities of the model participate, on the one hand, in the specification of workflows and, on the other, in the definition of access control rules and other provisions that regulate the workflows' execution and transformation. Indeed, two central concepts within our approach are the *Workflow* and the *Action*; network monitoring processes are typically invoked in the context of a workflow, while actions constitute not only the fundamental elements of workflows, but also the core of access control, being the main component of access control rules and the "seed" for knowledge extraction.

As afore mentioned, an *action* refers to the situation where an *operation* is performed by an *actor* on a *resource*. Different types of entities may play the role of an actor, including human users and functional components; similarly, resources may refer to several entities, such as data and system components. Considering both abstract and concrete entities, actors and resources comprise the corresponding sets:

- *Actors* (A) = $R \cup Op \cup OpCT \cup U \cup OpI \cup OpC$
- *Resources* (Res) = $DT \cup Op \cup OpCT \cup R \cup MT \cup AlT \cup U \cup OpI \cup OpC \cup M \cup D \cup Al$

Especially for concrete entities when playing the role of actor and resource within an action, we refer to them us *subjects* and *objects*, defining the corresponding sets *Subjects* (*Subj*) and *Objects* (*Obj*).

Each action act_i is defined as a tuple $\langle a_i, op_i, res_i \rangle$, where a_i is an actor, op_i is an operation and res_i is a resource. Considering in addition the organisation *org* within which an action takes place, an action is represented by the tuple $act_i = \langle a_i, op_i, res_i, org \rangle$. Indeed, the concept of organisation is of great importance in the context of the reference framework, as workflows may span across several domains and therefore, every single action must be associated with the domain within which it is performed. Following the hierarchical relations of operations *Op*, an action can be either *atomic* or *composite*, depending on whether the associated operation can be decomposed to more elementary operations or not.

It is stressed here that an action may contain exclusively concrete or abstract entities, or may contain both concrete and abstract ones. Especially regarding the definition of access control rules on actions, this constitutes an innovative feature introduced by our approach, since the current state-of-the-art approaches either make use of abstractions only for the subject, following the RBAC paradigm [25], or they focus clearly on policies specification in totally abstract terms. This hybrid perspective provides an advantage against the other models; while abstraction provides for a higher degree of generality and allows for cohesively treating entities falling into the same conceptualisation, it should be possible for access constraints to be defined also over concrete entities. In the reference framework, the operation of which is centred around workflows, actions and rules must be able to respond to access requests concerning both concrete and abstract levels, since the specification of a workflow may refer to both.

Depending on the level, concrete or abstract, that each of the four elements of an action is defined, there are sixteen different possibilities regarding the actions. An action that all the elements are defined at an abstract level is called an *Abstract Action*; for example, the following tuple implies that a security officer within a network operator executes an IDS application:

$\langle$`SecurityOfficer`, `execute`, `IDSApplication`, `NetworkOperator`$\rangle$,
where `SecurityOfficer` $\in R$, `execute` $\in Op$, `IDSApplication` $\in OpCT$ and `NetworkOperator` $\in OrgT$.

On the other hand, the action ⟨Ingrid, PC123-FTPClient-Connect, FTPServer100, StarryNightSA⟩ means that Ingrid $\in U$ is using PC123-FTPClient-Connect $\in OpI$ in order to connect to FTPServer100 $\in M$ within StarryNightSA $\in Org$. Such actions that consist only of concrete elements are characterised as *Concrete Actions*. It is noted that

PC123-FTPClient-Connect is assumed to be the instantiation of the operation `FTPConnect` $\in Op$, as offered by an operation container of type `FTPClient` $\in OpCT$ that is deployed on a machine of type `PersonalComputer` $\in MT$.

Going one step further, actions containing both abstract and concrete elements are referred to as *Semi-Abstract Actions*. For instance, an action may consist in a specific user u performing an operation op on some data resource of type dt within an organisation org, i.e., $act_i = \langle u, op, dt, org \rangle$, where $u \in U$, $op \in Op$, $dt \in DT$, $org \in Org$. Another case pointing the usefulness of combining concrete and abstract entities concerns actions explicitly mapped to a concrete organisation; for instance, authorisations defined for a role performing an operation on some resource may differ according to the organisation within which this action takes place.

Moreover, not all the the fields in the action tuple have to be populated. This means that an action may include the declaration of the operation only, regardless the possible actors and resources; or an action may consist of an actor and an operation. Actually, it is only the operation field that is mandatory; by itself, it can be enough for the specification of an action, leaving to the system the duty of filling the other fields with the appropriate values, e.g., the combinations of actors and resources that are meaningful for this operation. This is represented as the tuple $\langle *, op, *, * \rangle$, with the $*$ character indicating the fields that may be inferred regarding the defined authorisations for the corresponding op. Nevertheless, totally concrete actions derive from the abstract and semi-abstract actions as the result of a refinement process. Thus, a concrete action finally consists of a subject, an operation and an object.

When an action is part of a workflow, it is called a *Task*. In other words, a workflow is a structure consisting of a finite number of tasks, along with their interaction patterns (both data- and control-flow), that is, the tasks are the *cells* of a workflow. The interaction patterns are represented by means of edges connecting two adjacent tasks and are referred to as *Legs*. Thus, a leg is characterised by an initial task, a terminal task and a type denoting whether the leg constitutes a control or a data flow interaction. Finally, a workflow is represented as $w = (\langle t_1, t_2, \ldots, t_n \rangle, \langle l_1, l_2, \ldots, l_m \rangle)$, where $t_i = act_{i,w}$, or $t_i = \langle a_i, op_i, res_i \rangle_w$ and $l_k = \langle t_i, t_j, type \rangle$.

While the term *worklet* was introduced above, it is noted that actions and tasks can be parts of additional structures; their description is postponed for the following section, where their importance for putting constraints in access control rules is made clear.

5. Access Control Rules

Access control rules are used for defining *permissions*, *prohibitions* and *obligations*[5] over actions. Since actions can be abstract, concrete or semi-abstract, the rules are specified at these three different levels, as well. Evidently, the definition of rules at the highest possible level of abstraction enables a significant reduction of the number of policies; however, rules' representation at a concrete level not only enables the definition of exceptions, but also facilitates reasoning when tasks within a workflow are defined at the concrete level or contain both abstract and concrete elements.

The proposed model supports three types of rules, namely permissions, prohibitions and obligations, which are specified using the corresponding predicates:

$$\left.\begin{array}{c} Permission \\ Prohibition \\ Obligation \end{array}\right\} (pu, act, preAct, cont, postAct)$$

In these expressions, $act \in Act$ is the action that the rule applies to, $pu \in Pu$ is the purpose for which act is permitted/prohibited/obliged to be executed and $cont \in \mathcal{P}(Con)$ is a structure of contextual parameters. $preAct \in Act$ is a structure of actions that should have been preceded in order for an access control rule to be enforced, while $postAct \in Act$ refers to the action(s) that must be executed following the enforcement of the rule.

An example of such a rule is presented next, where an action referring to the execution of an operation op on an object obj by a role r_i excludes the execution of the same operation on the same object by a second role r_j for some purpose pu and within the same organisation org, regardless the possible context and post-actions:

- $Prohibition(pu, \langle r_j, op, obj, org \rangle, \langle r_i, op, obj, org \rangle, *, *)$

[5] We assume the use of immediate obligations, as defined in [59]. The use of obligations with delays are not considered in this work.

An important observation here is that the concept of organisation is not involved in the rules' body, but instead it is −directly or indirectly− specified for each action. Thus, although a rule concerns the execution of an action within an organisation, pre- and post- actions may take place within other organisations. This approach overcomes the need for defining a conceptual dynamic organisation, every time a rule refers to actions that span across different organisations.

As mentioned before, apart from single actions, pre- and post- actions may also refer to structures of actions. Thus, pre- and post-actions may consist of actions interrelated by means of logical operators $\wedge$ and $\vee$, including negation, i.e., $\neg$*preAct*, $\neg$*postAct*. Such logical structures of actions for the specification of pre- and post-actions do not imply any sequence constraint, i.e., the actions can be executed at any order. Nevertheless, there can be cases in a workflow where the individual actions comprising a pre-/post-action are executed in a specific order, or that are executed following specific patterns. Therefore, the definition of additional types of structural patterns is necessary.

In that respect, we characterise as *Skeleton* any structure of actions that must be executed according to a specific order, regardless whether other actions mediate between the actions of the skeleton. A skeleton can incorporate any control-flow pattern [60], such as AND-split/AND-join or XOR-split/XOR-join, where again the interactions between actions are modelled by means of legs. In the case when no other control-flow pattern but the *Sequence Pattern* [60] is used, the skeleton is referred to as a *Path*. Moreover, when the actions comprising a skeleton should be executed with no other action mediating, then the skeleton (or path) is characterised as *critical*, as opposed to *non-critical* skeleton. In order to denote the *criticality* of such a structure, the leg tuple is complemented by a fourth boolean attribute `att_Critical`. On the other hand, there may exist constraints regarding *when* a pre-/post- action is executed with respect to the action that the rule applies to. In this context, a pre-/post-action is characterised as *tight* when it must be executed immediately before or after the action, i.e., with no other actions mediating, or otherwise *loose*.

Issues related with administration and delegation are considered out of the scope of this paper; nevertheless, it should be noted that the described structure of the rules serves the principle of uniformity, allowing for a single style of specifying authorisations –either "basic" or administrative ones– while enabling the expression of rules of any order ([61]). In that respect, all aspects of policy are indeed controlled by the policy ([62]).

Regarding the reference scenario, the following rules result in the addition of the `AnonymiseTraffic` task in the workflow, which anonymises the destination IP address of the incoming packets, serving thus the principles of necessity, proportionality and minimal disclosure; `DetectFastFluxBotnet` does not need the destination IP addresses in order to accomplish its internal operation. Specifically, the second rule particularises the first regarding the `DestIP`:

- *Permission*(`NetworkSecurity`, ⟨`DetectFastFluxBotnet`, `read`, `DNSPacket`, StarryNightSA⟩, $*$, $*$, $*$)
- *Prohibition*(`NetworkSecurity`, ⟨`DetectFastFluxBotnet`, `read`, `DestIP`, StarryNightSA⟩, $\neg$⟨$*$, `AnonymiseTraffic`, `DestIP`, StarryNightSA⟩, $*$, $*$)

Similarly, the `ProjectFields` and `Aggregate` tasks have been added to the workflow. Regarding the former, we consider a prohibition for the `AssistantSecurityOfficer` to read a full `BotnetMitigationReport`; however, there exist explicit permissions concerning the access to `DomainName`, `Characteristics` and `ActivityStatistics`, contained in a `BotnetMitigationReport`. Thus, the projection task `ProjectFields` is incorporated, with the value of its attribute `att_Projection` set to the corresponding data types. Furthermore, although the `AssistantSecurityOfficer` has no access rights for data of type `BotnetAlert`, she is authorised to read `AggregatedAlert` data being less detailed than `BotnetAlert` ones; therefore, the `Aggregate` task performing such a transformation is added.

Additionally, the following rule prescribes that whenever a botnet alert with MPF greater that 0.9 occurs, the prompt notification of the `ChiefSecurityOfficer` should take place, and thus the task `Notify` has been added:

- *Obligation*(`NetworkSecurity`, ⟨$*$, `Notify`, `ChiefSecurityOfficer`, StarryNightSA⟩, ⟨$*$, `DetectBotnet`, $*$, StarryNightSA⟩, `BotnetAlert.MPF` $>$ `0.9`, $*$)

Finally, the next rules indicate the possible workflow differentiations according to the MPF of the produced alerts, resulting in the incorporation of conditional branches within the workflow. In that respect, a conditional branch containing the task `DetectBotnetDPI` has been added, so that a more thorough detection procedure is carried out for `0.6` $<$ `MPF` $<$ `0.8`, leading to a hybrid detection mechanism (like e.g. the one presented in [63]). In the second branch the initial task `MitigateBotnet` has been replaced by `MitigateBotnetMPLS` (being a way of performing `MitigateBotnet`), while also the condition has changed following the corresponding rule.

- *Obligation*(NetworkSecurity, ⟨∗, DetectBotnetDPI, ∗, StarryNightSA⟩, ⟨∗, DetectBotnet, ∗, StarryNightSA⟩, BotnetAlert.MPF > 0.6 && BotnetAlert.MPF < 0.8, ∗)
- *Permission*(NetworkSecurity, ⟨∗, MitigateBotnetMPLS, ∗, StarryNightSA⟩, ⟨∗, DetectBotnet, ∗, StarryNightSA⟩, BotnetAlert.MPF > 0.8, ∗)

It is noted that the rules defined for the operation DetectBotnet apply also for the DetectFastFluxBotnet of the reference workflow as the predicate *isA*(DetectFastFluxBotnet, DetectBotnet) holds (cf. Section 6).

Concrete authorisations derive from abstract or semi-abstract ones, while they can also be explicitly defined at the concrete level, e.g., as exceptions; they are modelled by means of the following predicates, where $subj \in Subj$, $op \in Op \cup OpI$ and $obj \in Obj$:

$$\left.\begin{array}{c} isPermitted \\ isProhibited \\ isObliged \end{array}\right\} (subj, op, obj)$$

Thus, concrete authorisations lead to the possible workflow instantiations, after the assessment of the access rights, along with the available capabilities. As an example, let's assume that Ingrid, the engineer on duty at the time of a severe Botnet system alarm (i.e., BotnetAlert.MPF > 0.9), has the role of JuniorNetworkAdministrator and, therefore, in order to implement the Notify task, she is authorised to only use the MakeVoIPCall operation offered by operation containers of type VoIPSoftwareClient. In addition, Ingrid is authorised to use only a limited number of the PersonalComputer machines that have VoIPSoftwareClient software deployed. Therefore, for the instantiation of the workflow during Ingrid's duty hours, the access control rules should enable the identification of a VoIPSoftwareClient enabled machine of type PersonalComputer that Ingrid is authorised to use, in order for the concrete task to be included in the workflow.

6. Inheritance of Authorisations

As it has been described in the corresponding sections, the information model considers hierarchies of data types, roles, operations and purposes, among others, which are mostly (cf. Table 2) formed by means of the *isA* and *isPartOf* relations. Because of the exactly same semantics of these relations for every considered graph, in the following we present general inheritance patterns for permissions, obligations and prohibitions associated with these different hierarchies for the case of data types, in order to also show how authorisations are inherited through the *lessDetailedThan* relation, associating only members of this particular graph. It is noted that as both permissions and prohibitions are supported, authorisations inherited along the hierarchies may conflict with explicitly defined ones and consequently be overridden by such authorisations; in any case, explicitly defined authorisations have higher priority than implicit ones, while also prohibitions prevail over permissions.

The *inheritsFrom* relation is inferred as a result of the others and is leveraged in order to reflect the inheritance of rules. In more detail, the *isA*, *isPartOf* and *lessDetailedThan* relations imply the *inheritsFrom* one as follows, depending also on the nature of an authorisation (positive vs. negative):

- $isA(dt_i, dt_j) \longrightarrow inheritsFrom(dt_i, dt_j)$,
  for both positive and negative authorisations, i.e., a permission (resp. prohibition) for a more general data type is inheritted to a more specific one.
- $isPartOf(dt_i, dt_j) \longrightarrow inheritsFrom(dt_i, dt_j)$,
  for positive authorisations, i.e., if access to a complex data type is permitted, the permission is propagated to all the included data types.
- $isPartOf(dt_i, dt_j) \longrightarrow inheritsFrom(dt_j, dt_i)$,
  for negative authorisations, i.e., if access to even one part of a data type is denied, the prohibition holds also for the complex one.

- *lessDetailedThan*(dt_i, dt_j) $\longrightarrow$ *inheritsFrom*(dt_i, dt_j),
  for positive authorisations, i.e., a permission for a more detailed data type implies a permission also for the less detailed ones.

- *lessDetailedThan*(dt_i, dt_j) $\longrightarrow$ *inheritsFrom*(dt_j, dt_i),
  for negative authorisations, i.e., if access to a data type being less detailed that another one is prohibited, then access to the latter is also prohibited.

It is noted that obligations follow the same inheritance pattern as the permissions.

Inheritance of authorisations is also considered between different graphs of the information model. For instance, authorisations concerning types of operations' containers affect also the offered operations, while this applies as well to the case of authorisations specified for a machine type which are consequently transferred to operations through the hosted operations' containers.

7. Separation and Binding of Duty

Separation of Duty (SoD) [64] constitutes another concept to be taken under consideration for the specification of access control rules. In its simplest form it is referred to as *Static Separation of Duty* (SSoD) prescribing that a user cannot be in any case assigned to mutually exclusive roles; this is indeed a static constraint decided during the policy specification phase, focusing only on the exclusion between roles without taking into account any other parameters. On the other hand, *Dynamic Separation of Duty* (DSoD) puts constraints on the simultaneous −typically during a *session*− activation of roles that the user is in principle enabled to hold. That is, mutual exclusiveness concerns activating roles, not holding them.

Another innovation of our approach concerns the specification of SoD constraints. Instead of relying on role-/user-centric constraints, we reconsider the exclusiveness concept by applying it to all elements comprising an action, i.e., the actor, the operation, the resource and the organisation. That is, SoD in our model concerns essentially actions that are mutually exclusive.

Similarly, this approach supports the specification and enforcement of constraints stemming by the *Binding of Duty* (BoD) principle. Typically, BoD refers to the requirement that the subject that performed something must be the same that will perform something else. Again, we extend this concept by generalising it to any element that can be part of an action. For instance, a BoD constraint may specify that if something is executed within an organisation, something else should also be executed within the same organisation.

In other words, SoD and BoD constraints can be specified on the basis of any combination of the $\langle a_i, op_i, res_i, org\rangle$ elements of actions. The means for their specification are provided by the access control rules themselves (cf. Section 5), i.e., in terms of permissions, prohibitions and obligations. More specifically, a DSoD constraint is specified by a prohibition rule:

- *Prohibition*($*$, *act*, *preAct*, $*$, $*$)

Essentially, the actions *act* and *preAct* contain the conflicting elements, thus preventing them from being executed according to the SoD constraint.

On the other hand, a BoD is specified as a permission that defines a positive exception to a default prohibition, by constraining the permission by means of a pre-action. That is, for binding elements of two actions *act* and *preAct*, the rule that is used is:

- *Permission*($*$, *act*, *preAct*, $*$, $*$)

It is noted that the rules above are not the only way for specifying a DSoD/ BoD; in fact, there are several alternative ways to express the same constraints.

Leveraging the concept of action for SoD and BoD specification results in a combination of *task-based* [65] and *history-based* [64] approach. In this context, every completed action is logged in the history of the system and on this basis, dynamic −specifically, history-based− SoD and BoD constraints are enforced. In fact, similarly to the *Session* concept of RBAC, there are cases when SoD and BoD constraints' scope is within a workflow, or other cases when

actions are conflicting/bound regardless the workflow to which they belong. For instance, a user must be restricted from performing an operation on a resource, if she has already performed some other operation on that same resource in the system's history.

Whether two actions are included within the same workflow is modelled by means of the `withinSameWorkflow` context, which can be evaluated during both workflow specification and invocation. For instance, a SoD constraint handled during workflow specification and concerning the execution of two operations on some data resource by the same role is represented by a prohibitive rule as the following:

- *Prohibition*(pu, $\langle r, op_i, d, org\rangle$, $\langle r, op_j, d, org\rangle$, `withinSameWorkflow`, $*$)

Coming back to the example scenario, a BoD restriction prescribing that the user which receives the workflow outputs must be the one having initiated this workflow may be imposed by the following rule:

- *Permission*($*$, $\langle$*initiator*, `ReportToGUI`, $*$, StarryNightSA$\rangle$, $\langle$*initiator*, `invoke`, *this*, StarryNightSA$\rangle$, `withinSameWorkflow`, $*$),
  where *initiator* $\in U$, StarryNightSA $\in Org$, and `ReportToGUI`, `invoke`, *this* $\in Op$, with *this* modelling the composite operation implemented by the considered workflow.

On the other hand, the following rule restricts a user from performing the operation op_i on a resource *res*, if the same user has already performed any operation (even the same operation) on that same resource in the system's history:

- *Prohibition*($*$, $\langle u, op_i, res\rangle$, $\langle u, *, res\rangle$, $*$, $*$)

Apart from the `withinSameWorkflow` contextual constraint, a SoD or BoD provision is subject to all possible contextual parameters. For example, a SoD or BoD may apply "for 24 hours" or "until an alert of type $alt \in AlT$ is received". Moreover, it should be noted that the actions participating in a SoD or BoD constraint definition can be structures of actions, such as *skeletons* (cf. Section 5).

Finally, the case of SSoD is specified by means of the *disjointWith* predicate, which, for the case of roles, is expressed as follows:

- *disjointWith*(r_i, r_j), that can be translated as *assignedWithRoles*(u, r_i) $\longrightarrow \neg$*assignedWithRoles*(u, r_j),
  where r_i, $r_j \in R$ and $u \in R$.

Nevertheless, apart from this typical case of static separation of roles, the same predicate can be used for all concepts of the abstract level of the information model. For example, the following predicate excludes a machine from having two types mt_i and mt_j at the same time:

- *disjointWith*(mt_i, mt_j), that can be translated as *isOfMachineType*(m, mt_i) $\longrightarrow \neg$*isOfMachineType*(m, mt_j),
  where mt_i, $mt_j \in MT$ and $m \in M$.

8. Discussion — Privacy Awareness

"*The machine is the problem: the solution is in the machine*" [66]; along this line, the approach presented in the previous Sections has been motivated by the idea of proactively embedding all the privacy aspects into the technologies themselves, as *Privacy by Design* anticipates. Therefore, in order to address the underlying issues, network monitoring architectures have been rethought so as to incorporate privacy preservation *in* their design and make associated requirements an integral part of their operational and functional procedures.

The starting point for this work has been the data protection legislation of the European Union. This is because it enforces a high standard of data protection, capturing all the important privacy provisions, and it constitutes the most representative, influential and mature approach worldwide; thus, it has been characterised as an "engine of a global regime" [67]. The legislation provides the important requirements that have been considered by this work, such as the principles of *lawfulness*, *purpose specification* and *binding*, data *necessity*, *adequacy*, *proportionality* and *minimisation*. For a detailed analysis, which is out of scope of this paper, the reader is referred to [2][68].

A very fundamental concept for privacy is the *purpose* for which data are processed, being a core part of the *lawfulness principle*. In the proposed system, purpose plays an important role (cf. Section 3.5); each workflow is associated with a number of purposes and all underlying actions are evaluated in accordance with the served purpose(s). In that respect, the information model includes the *Purposes* (*Pu*) set, whereas purpose is an integral part of the rules and a parameter that affects separation and binding of duty. In addition, the *purpose principle* prescribes mechanisms for specifying the compatibility between processing purposes, while also providing for checking whether the processing purposes are consistent with these for which data have been collected. Therefore, the proposed system provides the means for defining prevention rules regarding incompatible purposes. In order to achieve the above, the formal conceptualisation of purposes, by means of the *Pu* set, has been considered, while for their direct association with data processing activities, the *mayServePurposes* predicate implements the association with operations. Moreover, the proposed model provides the means for the specification of the purposes that a role may hold (*mayActForPurposes* predicate), as well as for checking compliance between a role's acting purpose and the ones served by an operation (*compliantWithPurpose* predicate). All these are complemented by well-defined norms for being inherited across the corresponding graphs of the concepts involved.

The principles of *necessity*, *adequacy* and *proportionality* are also tightly connected to purpose. In fact, the proposed system provides the means for necessity specification, as well as for examining whether the collection or processing of specific data is necessary for the provision of the service in question. In that respect, the adopted pattern for rules' specification implements a relation between data, processing activities, roles and purposes, thus enabling the definition of necessity and proportionality constraints. Due to the rich descriptiveness of the rules, these constraints are extended to past and future actions, contextual conditions and the organisation within which the actions are performed.

Furthermore, the proposed model enables the definition of different levels of *data granularity*, so that the accuracy of disclosed data can be adjusted depending on the purpose and the subject requesting access to the said data, among others. In that respect, the adopted approach puts in place the means for the definition of different AND- and OR- hierarchies, describing inclusion and particularisation, as well as a relation explicitly denoting the detail level (*lessDetailedThan* predicate). This way, the presented approach enables the description of concepts in a manner that is significantly more fine-grained than the other works in the field.

All the above foster the realisation of the *minimal use* principle, as also made clear by the example scenario (Fig. 2). Starting from raw network packets, several processing operations have been automatically added, as a result of reasoning on and consequent enforcement of access control rules, for minimising the amount of data that reach each operation. On their way, input data have been filtered (`FilterTraffic`), anonymised (`AnonymiseTraffic`) and aggregated (`Aggregate`), while the data to be presented to the human user have been subject to the appropriate projection of their fields (`ProjectFields`).

Considering the case of inter-domain network monitoring scenarios, where multiple organisations are involved, the proposed model provides for privacy compliance even when the data need to be circulated among different organisations and administrative domains. In that respect, several technical choices have been made. First, the adoption of the concept of *operations*, enabling the semantic characterisation of a functionality offered to external domains, along with the specification of appropriate security and privacy policies governing its provision. Second the inclusion of activities implemented by external domains as tasks within workflows, so that they are subject to verification at design-time as if they were executed internally. Finally, the incorporation of the *organisation* concept into the actions, providing a flexible way to consider complex organisational relations within the policies.

9. Conclusions

This paper has presented the specification of a new policy-based access control model that aims at enhancing distributed network monitoring architectures with privacy-awareness. The model is conceived on the basis of data protection legislation and network monitoring particular features and needs, and fosters the realisation of the *Privacy by Design* vision, by means of proactively embedding to the associated processes, already from the specification phase, all the privacy aspects that we expect from a network monitoring architecture.

In this context, the proposed access control framework relies on policies that are built on top of a rich information model, while the associated rules are specified over *actions* that reflect the operational activities carried out by and distributed across network monitoring systems. For the effective management of the distributed activities execution, said actions are organised into *workflows*; access control provisions are decided and enforced at the workflow

level, addressing the constraints stemming from the interrelations between actions, thus enabling the realisation of workflows verification and automatic modification at design-time.

The model also takes full advantage of the integration of contextual properties. This allows covering the definition of both simple and complex processes, while enabling to capture context-based differentiated workflow behaviours at design-time by means of conditional branching. On the other hand, our approach provides for the specification of access control rules at any level of abstraction, as they are defined over actions that may be concrete, abstract or semi-abstract, thus serving the need for flexibility during workflow specification.

Perspectives for future work, apart from the application of the approach in other domains as implied in Section 1.2, include the study of a more sophisticated approach for purpose verification based on the incorporation into the model of fuzzy relations between the elements of the triple *purpose–role–operation*, as well as the addition of structures to allow delegation of policies and the specification of dynamic workflow adjustment based on real-time constraints.

10. Acknowledgements

This research was partially supported by the European Commission, in the framework of the FP7 DEMONS project (Grant agreement no. FP7-257315). We also acknowledge partial support from the Spanish MICINN projects: ARES-CONSOLIDER INGENIO 2010 CSD2007-00004 and eAEGIS TSI2007-65406-C03-03.

The research of M. N. Koukovini is co-financed by the European Union (European Social Fund — ESF) and Greek national funds through the Operational Program "Education and Lifelong Learning" of the National Strategic Reference Framework (NSRF) — Research Funding Program: *Heracleitus II. Investing in knowledge society through the European Social Fund*.

The authors would like to express their gratitude to the anonymous reviewers for their insightful comments that helped to improve the paper.

[1] Article 29 Data Protection Working Party, Opinion 4/2007 on the Concept of Personal Data, WP 136 (2007).

[2] G. V. Lioudakis, F. Gaudino, E. Boschi, G. Bianchi, D. I. Kaklamani, I. S. Venieris, Legislation-aware privacy protection in passive network monitoring, in: I. M. Portela, M. M. Cruz-Cunha (Eds.), Information Communication Technology Law, Protection and Access Rights: Global Approaches and Issues, IGI Global, 2010, Ch. 22, pp. 363–383.

[3] D. C. Sicker, P. Ohm, D. Grunwald, Legal issues surrounding monitoring during network research, in: IMC '07: Proceedings of the 7th ACM SIGCOMM conference on Internet measurement, ACM, New York, NY, USA, 2007, pp. 141–148.

[4] P. Lundevall-Unger, T. Tranvik, IP addresses – just a number?, International Journal of Law and Information Technology 19 (1) (2011) 53–73.

[5] A. Zugenmaier, J. Claessens, Privacy in electronic communications, in: C. Douligeris, D. Serpanos (Eds.), Network Security: Current Status and Future Directions, Wiley-Interscience & IEEE Press, 2007, pp. 419–440.

[6] G. V. Lioudakis, E. A. Koutsoloukas, N. L. Dellas, N. Tselikas, S. Kapellaki, G. N. Prezerakos, D. I. Kaklamani, I. S. Venieris, A middleware architecture for privacy protection, Computer Networks 51 (16) (2007) 4679–4696.

[7] L. F. Cranor, 'I didn't buy it for myself': privacy and eCommerce personalization, in: C.-M. Karat, J. O. Blom, J. Karat (Eds.), Designing personalized user experiences in eCommerce, Kluwer Academic Publishers, Norwell, MA, USA, 2004, pp. 57–73.

[8] G. Bissias, M. Liberatore, D. Jensen, B. Levine, Privacy vulnerabilities in encrypted HTTP streams, in: G. Danezis, D. Martin (Eds.), Privacy Enhancing Technologies, Vol. 3856 of Lecture Notes in Computer Science, Springer, 2005, pp. 1–11.

[9] M. Crotti, F. Gringoli, P. Pelosato, L. Salgarelli, A statistical approach to IP-level classification of network traffic, in: ICC'06: Proceedings of the IEEE International Conference on Communications, IEEE, 2006, pp. 170–176.

[10] A. Hintz, Fingerprinting websites using traffic analysis, in: R. Dingledine, P. Syverson (Eds.), Privacy Enhancing Technologies, Vol. 2482 of Lecture Notes in Computer Science, Springer, 2002, pp. 229–233.

[11] X. Gong, N. Kiyavash, N. Borisov, Fingerprinting websites using remote traffic analysis, in: CCS '10: Proceedings of the 17th ACM conference on Computer and communications security, ACM, New York, NY, USA, 2010, pp. 684–686.

[12] G. Danezis, Traffic analysis of the TLS protocol and its suitability for providing privacy properties, Tech. rep., University of Cambridge (2002).

[13] Q. Sun, D. R. Simon, Y.-M. Wang, W. Russell, V. N. Padmanabhan, L. Qiu, Statistical identification of encrypted web browsing traffic, in: Proceedings of the 2002 IEEE Symposium on Security and Privacy, IEEE Computer Society, Washington, DC, USA, 2002, pp. 19–30.

[14] S. M. Bellovin, A technique for counting NATted hosts, in: IMW '02: Proceedings of the 2nd ACM SIGCOMM Workshop on Internet measurment, ACM, New York, NY, USA, 2002, pp. 267–272.

[15] M. Gil-Perez, F. Gomez-Marmol, G. Martinez-Perez, A. F. Gomez-Skarmeta, Mobility in collaborative alert systems: Building trust through reputation, in: International IFIP TC 6 Workshops (NETWORKING 2011), 2011, pp. 251–262.

[16] A. Cavoukian, Privacy by design (2009).
URL `http://privacybydesign.ca/publications/pbd-the-book/`

[17] J. Fan, J. Xu, M. H. Ammar, S. B. Moon, Prefix-preserving IP address anonymization, Computer Networks 46 (2) (2004) 253–272.

[18] M. Foukarakis, D. Antoniades, S. Antonatos, E. Markatos, Flexible and high-performance anonymization of NetFlow records using anontool, in: SECURECOMM Conference, 2007.

[19] D. Koukis, S. Antonatos, D. Antoniades, E. Markatos, P. Trimintzios, A generic anonymization framework for network traffic, in: ICC '06: IEEE International Conference on Communications, Vol. 5, 2006, pp. 2302 –2309.

[20] Y. Li, A. Slagell, K. Luo, W. Yurcik, Canine: A combined conversion and anonymization tool for processing netflows for security., in: International Conference on Telecommunication Systems Modeling and Analysis, 2005.
[21] G. Minshall, Tcpdpriv.
URL http://ita.ee.lbl.gov/html/contrib/tcpdpriv.html
[22] M. Burkhart, D. Schatzmann, B. Trammell, E. Boschi, B. Plattner, The role of network trace anonymization under attack, Comput. Commun. Rev. 40 (1) (2010) 5–11.
[23] R. Pang, M. Allman, V. Paxson, J. Lee, The devil and packet trace anonymization, Comput. Commun. Rev. 36 (1) (2006) 29–38.
[24] A. Antonakopoulou, G. V. Lioudakis, F. Gogoulos, D. I. Kaklamani, I. S. Venieris, Leveraging access control for privacy protection: A survey, in: G. Yee (Ed.), Privacy Protection Measures and Technologies in Business Organizations: Aspects and Standards, IGI Global, 2012, pp. 65–94.
[25] D. F. Ferraiolo, R. Sandhu, S. Gavrila, D. R. Kuhn, R. Chandramouli, Proposed NIST standard for role-based access control, ACM Trans. Inf. Syst. Secur. 4 (3) (2001) 224–274.
[26] R. Agrawal, J. Kiernan, R. Srikant, Y. Xu, Hippocratic databases, in: VLDB '02: Proceedings of the 28th international conference on Very Large Data Bases, VLDB Endowment, 2002, pp. 143–154.
[27] K. LeFevre, R. Agrawal, V. Ercegovac, R. Ramakrishnan, Y. Xu, D. DeWitt, Limiting disclosure in hippocratic databases, in: VLDB '04: Proceedings of the 30th international conference on Very Large Data Bases, VLDB Endowment, 2004, pp. 108–119.
[28] F. Massacci, J. Mylopoulos, N. Zannone, Hierarchical hippocratic databases with minimal disclosure for virtual organizations, The VLDB Journal 15 (2006) 370–387.
[29] J.-W. Byun, N. Li, Purpose based access control for privacy protection in relational database systems, The VLDB Journal 17 (4) (2008) 603–619.
[30] C. A. Ardagna, J. Camenisch, M. Kohlweiss, R. Leenes, G. Neven, B. Priem, P. Samarati, D. Sommer, M. Verdicchio, Exploiting cryptography for privacy-enhanced access control: A result of the prime project, Journal of Computer Security 18 (1) (2010) 123–160.
[31] A. Masoumzadeh, J. Joshi, PuRBAC: Purpose-aware role-based access control, in: R. Meersman, Z. Tari (Eds.), On the Move to Meaningful Internet Systems: OTM 2008, Vol. 5332 of Lecture Notes in Computer Science, Springer, 2008, pp. 1104–1121.
[32] Q. Ni, E. Bertino, J. Lobo, C. Brodie, C.-M. Karat, J. Karat, A. Trombeta, Privacy-aware role-based access control, ACM Trans. Inf. Syst. Secur. 13 (3) (2010) 24:1–24:31.
[33] Y. Zhang, J. Yang, L. Gao, Supporting flexible streaming media protection through privacy-aware secure processors, Computers and Electrical Engineering 35 (1) (2008) 286–299.
[34] S. Ayed, N. Cuppens-Boulahia, F. Cuppens, Deploying security policy in intra and inter workflow management systems, in: ARES '09: International Conference on Availability, Reliability and Security, 2009, pp. 58–65.
[35] G. Russello, C. Dong, N. Dulay, A workflow-based access control framework for e-health applications, in: WAINA 2008: Proceedings of the 22nd International Conference on Advanced Information Networking and Applications Workshops, IEEE Computer Society, Washington, DC, USA, 2008, pp. 111–120.
[36] M. Alam, M. Hafner, R. Breu, Constraint based role based access control in the sectet-framework a model-driven approach, Journal of Computer Security 16 (2) (2008) 223–260.
[37] M. Menzel, C. Meinel, SecureSOA, IEEE International Conference on Services Computing (2010) 146–153.
[38] A. Abou-El-Kalam, R. E. Baida, P. Balbiani, S. Benferhat, F. Cuppens, Y. Deswarte, A. Miège, C. Saurel, G. Trouessin, Organization Based Access Control, in: 4th IEEE International Workshop on Policies for Distributed Systems and Networks (Policy'03), 2003, pp. 120–131, lake Come, Italy.
[39] F. Cuppens, N. Cuppens-Boulahia, Modeling Contextual Security Policies, International Journal of Information Security 7 (4) (2008) 285–305.
[40] S. Preda, F. Cuppens, N. Cuppens-Boulahia, J. Garcia-Alfaro, L. Toutain, Dynamic deployment of context-aware access control policies for constrained security devices, Journal of Systems and Software 84 (2011) 1144–1159.
[41] F. Gogoulos, A. Antonakopoulou, G. V. Lioudakis, A. S. Mousas, D. I. Kaklamani, I. S. Venieris, Privacy-aware access control and authorization in passive network monitoring infrastructures, in: CIT 2010: Proceedings of the 10th IEEE International Conference on Computer and Information Technology, 2010.
[42] G. V. Lioudakis, F. Gogoulos, A. Antonakopoulou, A. S. Mousas, I. S. Venieris, D. I. Kaklamani, An access control approach for privacy-preserving passive network monitoring, in: ICITST 2009: Proceedings of the 4th International Conference for Internet Technology and Secured Transactions, 2009.
[43] N. Ajam, N. Cuppens-Boulahia, F. Cuppens, Contextual privacy management in extended role based access control model, in: J. Garcia-Alfaro, G. Navarro-Arribas, N. Cuppens-Boulahia, Y. Roudier (Eds.), Data Privacy Management and Autonomous Spontaneous Security, Vol. 5939 of Lecture Notes in Computer Science, Springer, 2010, pp. 121–135.
[44] E. I. Papagiannakopoulou, M. N. Koukovini, G. V. Lioudakis, J. Garcia-Alfaro, D. I. Kaklamani, I. S. Venieris, A contextual privacy-aware access control model for network monitoring workflows: Work in progress, in: J. Garcia-Alfaro, P. Lafourcade (Eds.), FPS 2011: Proceedings of the 4th MITACS Workshop on Foundations & Practice of Security, Vol. 6888 of Lecture Notes in Computer Science, Springer, 2011, pp. 208–217.
[45] H. Xiong, Z. Chen, F. Li, Efficient and multi-level privacy-preserving communication protocol for VANET, Computers and Electrical Engineering 38 (3) (2012) 573–581.
[46] P. Martinez-Julia, A. F. Gomez-Skarmeta, Using identities to achieve enhanced privacy in future content delivery networks, Computers and Electrical Engineering 38 (2) (2012) 346–355.
[47] S. V. Halunga, N. Vizireanu, Performance evaluation for conventional and MMSE multiuser detection algorithms in imperfect reception conditions, Digital Signal Processing 20 (1) (2010) 166–178.
[48] S. V. Halunga, N. Vizireanu, O. Fratu, Imperfect cross-correlation and amplitude balance effects on conventional multiuser decoder with turbo encoding, Digital Signal Processing 20 (1) (2010) 191–200.
[49] L. Wang, D. Chen, F. Huang, Virtual workflow system for distributed collaborative scientific applications on grids, Computers and Electrical

Engineering 37 (3) (2011) 300–310.
[50] S. H. Yeganeh, J. Habibi, H. Rostami, H. Abolhassani, Semantic web service composition testbed, Computers and Electrical Engineering 36 (5) (2010) 805–817.
[51] M. N. Koukovini, E. I. Papagiannakopoulou, G. V. Lioudakis, D. I. Kaklamani, I. S. Venieris, A workflow checking approach for inherent privacy awareness in network monitoring, in: J. Garcia-Alfaro, N. Cuppens-Boulahia, G. Navarro-Arribas (Eds.), DPM 2011: Proceedings of the 6th International Workshop on Data Privacy Management, Vol. 7122 of Lecture Notes in Computer Science, Springer, 2011.
[52] T. Holz, C. Gorecki, K. Rieck, F. C. Freiling, Measuring and detecting fast-flux service networks, in: NDSS '08: Proceedings of the Network and Distributed System Security Symposium, The Internet Society, 2008.
[53] G. V. Lioudakis, A.-C. G. Anadiotis, A. S. Mousas, C. Z. Patrikakis, D. I. Kaklamani, I. S. Venieris, Routing in content-centric networks: From names to concepts, in: NTMS 2012: Proceedings of the 5th International Conference on New Technologies, Mobility and Security, 2012.
[54] B. Claise, Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of IP Traffic Flow Information, RFC 5101 (Proposed Standard) (Jan. 2008).
URL http://www.ietf.org/rfc/rfc5101.txt
[55] C. Basile, A. Lioy, G. Martinez-Perez, F. J. Garcia-Clemente, A. F. Gomez-Skarmeta, POSITIF: A Policy-Based Security Management System, in: 8th IEEE International Workshop on Policies for Distributed Systems and Networks (POLICY 2007), 2007, p. 280.
[56] H. Debar, D. Curry, B. Feinstein, The Intrusion Detection Message Exchange Format (IDMEF), RFC 4765 (Experimental) (March 2007).
URL http://www.ietf.org/rfc/rfc4765.txt
[57] M. J. West-Brown, D. Stikvoort, K.-P. Kossakowski, G. Killcrece, R. Ruefle, M. Zajicek, Handbook for computer security incident response teams (CSIRTs), Tech. Rep. CMU/SEI-2003-HB-002, Carnegie Mellon University, Software Engineering Institute (2003).
[58] F. J. Garcia-Clemente, G. Martinez-Perez, A. F. Gomez-Skarmeta, Multi-layer framework for analysing and managing routing configurations, Computers and Electrical Engineering 35 (5) (2009) 634–643.
[59] Y. Elrakaiby, F. Cuppens, N. Cuppens-Boulahia, Formal enforcement and management of obligation policies, Data Knowl. Eng. 71 (1) (2012) 127–147.
[60] N. Russell, A. H. M. Ter Hofstede, W. M. van der Aalst, N. Mulyar, Workflow control-flow patterns: A revised view, Tech. Rep. BPM-06-22, BPM Center (2006).
[61] G. Bruns, M. Huth, K. Avijit, Program synthesis in administration of higher-order permissions, in: SACMAT '11: Proceedings of the 16th ACM symposium on Access control models and technologies, ACM, New York, NY, USA, 2011, pp. 41–50.
[62] N. Li, Z. Mao, Administration in role-based access control, in: ASIACCS '07: Proceedings of the 2nd ACM symposium on Information, computer and communications security, ACM, 2007, pp. 127–138.
[63] M. A. Aydin, A. H. Zaim, K. Ceylan, A hybrid intrusion detection system design for computer network security, Computers and Electrical Engineering 35 (3) (2009) 517–526.
[64] R. Simon, M. E. Zurko, Separation of duty in role-based environments, in: CSFW '97: Proceedings of the 10th IEEE workshop on Computer Security Foundations, IEEE Computer Society, 1997.
[65] R. A. Botha, J. H. P. Eloff, Separation of duties for access control enforcement in workflow environments, IBM Systems Journal 40 (3) (2001) 666–682.
[66] Y. Poullet, EU data protection policy. The Directive 95/46/EC: Ten years after, Computer Law & Security Report 22 (3) (2006) 206–217.
[67] M. D. Birnhack, The EU Data Protection Directive: An engine of a global regime, Computer Law & Security Report 24 (6) (2008) 508–520.
[68] F. Gaudino, G. V. Lioudakis, Interjurisdictional privacy compliance, DEMONS Deliverable D2.2 (2011).
URL http://fp7-demons.eu/wp-content/FP7-DEMONS-D.2.2-FINAL.pdf

Eugenia I. Papagiannakopoulou received her Diploma in Electrical and Computer Engineering from the School of Electrical and Computer Engineering, National Technical University of Athens, and she is currently working towards the PhD degree. Her research interests include security and privacy protection in dynamic and distributed systems, semantic web technologies and software engineering.

Maria N. Koukovini received her Diploma in Electrical and Computer Engineering from the School of Electrical and Computer Engineering, National Technical University of Athens, and she is currently working towards the PhD degree. Her research interests include workflow management systems, semantic web technologies, software engineering, security and privacy protection.

Georgios V. Lioudakis is a senior research associate of the National Technical University of Athens. He holds a Dipl.-Ing. degree and a Dr.-Ing. degree in Electrical and Computer Engineering, both from the National Technical University of Athens. His research interests include security and privacy protection, software engineering, middleware and distributed technologies.

Joaquin Garcia-Alfaro is a research associate at the TELECOM Bretagne, LUSSI department. He holds an Engineering degree in Computer Science and a PhD. His research includes management of security policies, analysis of vulnerabilities and enforcement of countermeasures.

Dimitra I. Kaklamani is a professor at the School of Electrical and Computer Engineering, National Technical University of Athens. Her research interests span across different fields and include security and privacy, middleware technologies and distributed systems, and the development of visualisation and real-time simulation techniques for solving complex, large scale mod-

elling problems of microwave engineering and information transmission systems.

Iakovos S. Venieris is a professor in the School of Electrical and Computer Engineering, National Technical University of Athens. His research interests include distributed systems, privacy-enhancing technologies, software and service engineering, agent technology, multimedia, mobile communications and Intelligent Networks, resource scheduling and allocation for network management, modelling, performance evaluation and queuing theory.

Frédéric Cuppens is a full professor at the TELECOM Bretagne, LUSSI department. He holds an Engineering degree in Computer Science, a PhD and an HDR. His research includes formal models, security policies, access control and intrusion detection.

Nora Cuppens-Boulahia is an associate researcher at the TELECOM Bretagne, LUSSI department. She holds an Engineering degree in Computer Science, a PhD and an HDR. Her research includes formalization of security properties, cryptographic protocol analysis, formal validation and thread assessment.