An Adaptive Mitigation Framework for Handling Suspicious Network Flows via MPLS Policies

Nabil Hachem, Joaquin Garcia-Alfaro and Herve Debar

Institut Mines-Telecom, Telecom SudParis
CNRS Samovar UMR 5157, Evry, France

Abstract. As network attacks become more complex, defence strategies must provide means to handle more flexible and dynamic requirements. The Multiprotocol Label Switching (MPLS) standard is a promising method to properly handle suspicious flows participating in such network attacks. Tasks such as alert data extraction, and MPLS routers configuration present an entailment to activate the defence process. This paper introduces a novel framework to define, generate and implement mitigation policies on MPLS routers. The activation of such policies is triggered by the alerts and expressed using a high level formalism. An implementation of the approach is presented.

Keywords: Network Security, Policy Management, MPLS, OrBAC.

1 Introduction

Nowadays, protecting data and network resources requires a whole new set of processes and technology challenges [30]. In [16] we presented HADEGA, a novel and efficient mitigation technique to counter network attacks. HADEGA relies on MPLS (Multiprotocol Label Switching [25]). The MPLS technology is widely used by service providers (i.e. to establish VPN, or to maintain service level guarantees, etc.) and presents a de-facto standard practice for Traffic Engineering and Differentiated Services. In HADEGA, MPLS is used for the sake of network security: through the settlement of various routing and QoS schemes on suspicious communications flowing across service providers' networks.

As it happens with many other mitigation technologies, HADEGA requires the enforcement of appropriate security rules triggered by adaptive defence processes (e.g., monitoring tools reporting incidents via alerts). However, in the original proposal of HADEGA, the management aspect of the solution was omitted. The goal of this paper is, therefore, to complement the HADEGA approach by addressing this crucial aspect. Our goal is to develop a policy-based management tool that post-processes the output of monitoring tools (e.g., incident alerts) and provide the appropriate mitigation scripts necessary to configure MPLS routers. For this purpose, we use a high level formalism based on the OrBAC model [21]. OrBAC is chosen for its expressiveness and transformation capabilities, which are rich enough to cover all the necessities of our approach. The OrBAC model is used in a top-down fashion, to properly generate MPLS router configuration rules from high-level (abstract) routing and QoS mitigation policies.

We validate our proposal by presenting an ongoing prototype developed under the open source MotOrBAC framework [2]. MotOrBAC already provides some of the necessary elements of our approach, such as the OrBAC policy editor and a powerful Application Programming Interface (API) to extend the capabilities of the editor. In our case, we extend such capabilities by adding (1) a new policy instantiation engine to provide the mapping between OrBAC policies and alerts; and (2) a policy transformation engine to translate the inferred rules into MPLS configuration scripts.

Paper organization — Section 2 elaborates further on our motivation problem and provides some background and state of the art literature. Sections 3 and 4 address the modeling of MPLS reaction policies using the OrBAC formalism. Section 5 overviews the ongoing development of a practical implementation of our approach. Section 6 presents a discussion and some related work. Section 7 concludes the paper.

2 Background

2.1 HADEGA

In the normal context, an MPLS domain is responsible to direct packet flows along a predetermined path in a per-route scheme. It also defines packets behaviour in a per-hop scheme. These dual schemes are achieved in MPLS through Traffic Engineering [5] and Differentiated Services [14] strengths. We presented in [16] HADEGA, a novel mitigation technique that benefits from these strengths to mitigate and reduce the impact of suspicious flows. In HADEGA, each MPLS domain is seen as a single packet forwarding component that first aggregates the suspicious flows, and second controls them (e.g., *de-prioritizes* their treatment or points them to a *blackhole*). The network suspicious flows are associated to suspicious traffic classes. The definition of these classes relies on network and assessment information. Mapping the suspicious flows to these classes is achieved via the data extracted from the alerts raised by monitoring tools. Then, MPLS labels are associated to those suspicious flows. These labels bounded to suspicious packets are used to make the treatment and forwarding decision all over the MPLS domain. From a life-cycle perspective, HADEGA consists of the following processes:

Planning Process: it consists of the definition of a pool of class of suspicious services, paths and forwarding behaviour treatments. The suspicious class of services are fixed based on security assessment attributes. The paths are distinguished by their distinct per-route attributes (i.e., number of hops, minimum/maximum bandwidth, link colors, etc.). The forwarding behaviour treatments have different per-hop attributes (i.e., scheduling, dropping policy, etc.). These paths and forwarding behaviour treatments handle the suspicious flows. We call them suspicious paths and forwarding behaviour treatments. The planning process is based on the predicted state of traffic load and existing traffic views. It consists of *long-term* strategies. It is done off-line taking into account global network conditions and traffic load.

Reaction Process: it consists of responding to alerts on both network (i.e., performance) and security (i.e., threat) levels. The reaction process is divided into two aspects: (1) network adaptation and (2) flow admission control.

– *Network adaptation control* is a *short-term* aspect, limited to minutes or hours. It is triggered by network performance alerts reporting significant changes in the traffic load or the network topology, or the inability of the *long-term* strategies defined in the planning process to adapt properly. It consists of employing certain dynamic resources and route management procedures for the previously established suspicious paths and forwarding behaviour treatments.
– *Flow admission control* extends throughout the reaction process. It is based on security alerts. The network attributes of security alerts, such as IP addresses and port numbers, are used to define and control suspicious flows through Forward Equivalence Class (FEC) definition. Assessment attributes, such as impact and confidence, are used to map these flows to their corresponding path and forwarding behaviour. Mapping these FECs to a single or a set of Next Hop Label Forward Entry (NHLFE) —via FEC-to-NHLFE tables— permits the assignment of these suspicious packets to the previously established suspicious paths and forwarding behaviour treatments.

The reaction process arises the essential need of an automated and adaptive management tool addressing both the network and security levels. A policy-based approach is the adequate solution for the management of such tool. It permits the adaptability to dynamic changes on both levels. It allows as well the application of the policy rules to the MPLS large-scale networks and heterogeneous routers.

2.2 Policy-based Management

Policy-based management improves flexibility within the management system. Policies can be considered as guidelines for the behaviour of a system [28]. The IT communities are performing research and implementation activities of policy-based techniques in several fields, such as: network, caching, security management and others. Two main frameworks are relevant in our work: network and security based frameworks.

Network policy-based management frameworks are extensively adopted for QoS matters. They aim at driving network devices and resources to meet system requirements, e.g., Service Level Agreement (SLA) assignments [17]. Several work has been performed in the literature to consistently adapt to these assignments in a Differentiated Service capable networks, such as the work of Snir et al. [29], Verma et al. [33] and Stone et al. [31]. Other presented frameworks for representing MPLS policies, including MPLS for traffic Engineering and QoS, such as the work of Isoyama et al. [19] and Brunner et al. [6].

Security policy-based management frameworks focus on the protection of system and network resources. They are commonly used to express access control or usage policies. These policies define the high-level rules specifying the conditions under which subjects are permitted to access targets [26]. For instance, RNBS [18] is a security policy-based management frameworks based on the Role-Based Access Control (RBAC) model [27] to manage access control rules on firewalls. Other examples such as [8] and [15] include the use of the Organization Role-Based Access Control (OrBAC) model [21] to refine and deploy global security policies into other network security components, such as intrusion detection systems and VPN routers.

Policy languages are classified into different groups, according to their application scenarios [17]. Network policy languages include Ponder [10], PDL [22] and others. Security policy languages include XACML [32], REI [20] and others. Our policy driven approach consists of expressing two reaction policies that handle the security and the network management levels. The high level language needed has to be expressive enough. It should be capable of expressing policies for both network and security management. We base our approach on OrBAC to specify these reaction policies.

2.3 OrBAC

Organization is the centric concept in the OrBAC model [21]. An *organization* is considered any entity in charge of managing a security policy. The goal of the OrBAC model is to specify security policies abstractly from implementation details. It proposes reasoning with the roles that subjects, actions or objects play at an organizational level. A *subject* is empowered into a *role*, an *action* is considered to implement an *activity*, and an *object* is used in a *view* (cf. Listing 1.1 in Appendix A).

By adopting this abstract conception, each *organization* can then set security rules which specify that some roles are permitted, prohibited or obliged to perform some other actions. The activation of these security rules may depend on contextual stipulations. To this end, the concept of *context* is explicitly introduced in OrBAC. By using a formalism based on first order logic, security rules are modelled using a 6-tuple predicate as per the following rule:

$$security\_rule(type, organization, role, activity, view, context)$$

The type belongs to *permission*, *prohibition*, or *obligation*. *Organization*, *role*, *activity*, *view* and *context* concepts can be structured hierarchically. *Permission*, *prohibition* and *obligation* rules are inherited through these hierarchies [9].

A *context* is used as a supplementary condition that must be satisfied to activate a given privilege (i.e. *permission*, *prohibition* or *obligation*). Using this notion, the OrBAC model provides the means to deal with flexible and dynamic requirements. In [7], they presented several types of *context* – temporal, spatial, prerequisite, user-declared and provisional contexts – and explained how to model them in the OrBAC model.

In [12, 13], the OrBAC model is used to express reaction policies. A threat context manages the intrusion detection alerts which are expressed in the Intrusion Detection Message Exchange Format (IDMEF) [11]. The threat context first specifies the alert classification, and second triggers the activation and the mapping between alert attributes and concrete entities of the OrBAC model. In [4], an extension to the previous approach is presented. The novelty is the use of dynamic organizations to ease the definition and enforcement of more elaborated reaction requirements. The dynamic organization concept is used to map the alerts and the policy using entities at the abstract level of the OrBAC model, through XPath expressions. In the sequel, we show how to use these previous efforts based on the OrBAC formalism to properly generate MPLS router rules to enforce the dual reaction policies of the HADEGA approach.

2.4 MPLS Reaction Policies using OrBAC

HADEGA relies on two reaction policies: (1) a network management policy and (2) an access control policy. Figure 1 depicts the work-flow associated to each policy.

- *Network management policy* permits the adaptation of network resources. It is triggered by performance alerts; these alerts are raised by network monitoring tools. A performance context is activated to manage the given performance alert. The activation of this context specifies a network adaptation rule expressed as an *obligation* security rule. This rule consists of establishing network management changes, such as changing the routing and QoS scheme of paths inside the MPLS domain.

- *Access control policy* provides the flow admission control. It is triggered by security alerts; when a security monitoring tool raises an alert, and the alert diagnosis data identify a suspicious flow as a part of an attack, a flow admission rule expressed as a *permission* security rule is activated. It affects the suspicious flow to the proper routing and QoS scheme inside the MPLS domain. The process is set off by the activation of a threat context that manages the given security alert.

Each type of reaction policy requires a different modelling due to different inputs and entities involved in each aspect. Next, we develop the modelling of each policy.

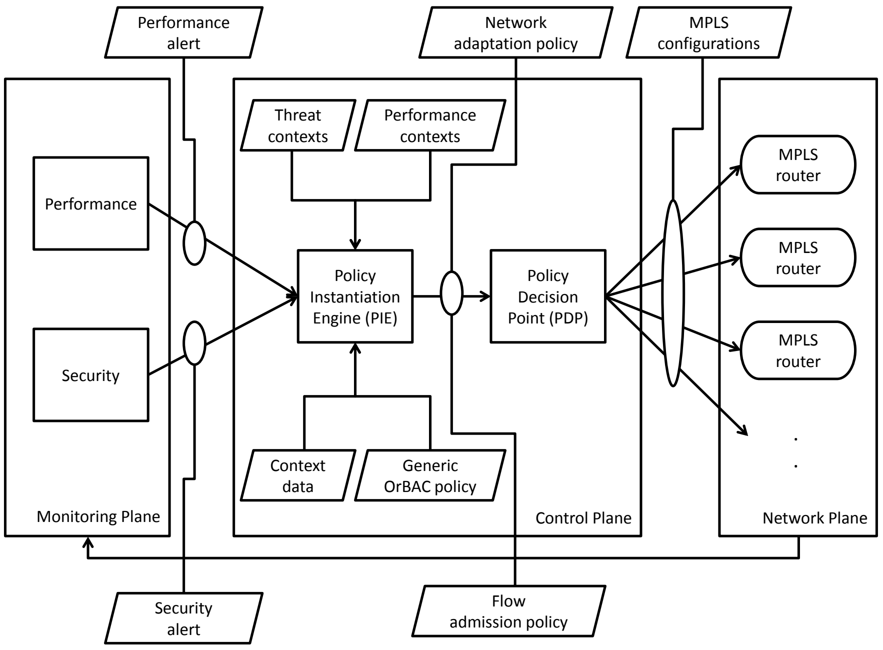


Fig. 1. Generation of MPLS configuration rules based on monitoring alerts

Concrete level	Definition	Attributes
Subject	MPLS path	ingress & egress router, identifier
Action	reroute, deactivate, etc.	explicit, implicit
Object	routing and QoS schemes	resource class, bandwidth set-up/hold priority, etc.

Table 1. Concrete entities

3 Network Adaptation Policy

The network adaptation is established via network management policies. These policies include modifying the path and/or the forwarding behaviour treatment for certain packets. Such policies are established on the network-level. They are performed by the ingress router and take effect on all the domain via MPLS paths. The network management policies include also changing queue length or scheduler weight. These policies are considered as device level; therefore the configurations apply for the specific device [6]. Although HADEGA proposes changes on these two levels, we address solely the policy that takes effect on the network-level.

Let us consider the following network management policy statement: *In the saturation network phase, paths holding high level suspicious flows must be pointed to a blackhole*. The ingress router maintains this policy by triggering the process of path and forwarding behaviour modification. Therefore, if a performance alert is received with a classification that maps to the saturation performance context, then this context is activated in the corresponding sub-organization and the network adaptation rule or a subset of rules are activated. These rules are turned into configuration rules on the ingress router of the MPLS domain suffering from saturation usage.

3.1 Concrete Entities

Table 1 summarizes our proposed set of concrete entities.

– *Subject*: we call it MPLS path. This path supports Diff-Serv e.g., the L-LSP or E-LSP (defined in [14] to map DiffServ treatment into MPLS paths). The MPLS path is distinguished by its start and end point which are the ingress and egress router and by certain identifier e.g., the NHLFE (the LSP Next Hop for a particular FEC is the next hop as selected by the NHLFE table entry [25]).
– *Action*: the basic network operation significant for Traffic Engineering and Diff-Serv. It can be a reroute, establish, deactivate, etc. Such action can be performed (1) explicitly by including all or some hops or (2) dynamically via certain path computation engine and signalling protocols.
– *Object*: represents the routing and QoS schemes that model the MPLS path. It is defined by different attributes like bandwidth, set-up priority, hold priority, link color affinity, scheduling/queuing priority, discarding policy, hops, etc.

3.2 Abstract Entities

Table 2 summarizes our proposed set of abstract entities. We assume the following entities in the network adaptation policy:

Abstract level	Definition	Examples
Role	path and forwarding behaviour	*gold_path*, *suspicious_path*
Activity	operation	*modify*, *remove*
View	network resources	*nodes*, *links*, *bandwidth*

Table 2. Abstract entities associated to the *DomainAdapt* organization

- *Organization*: *Domain adapt (DomainAdapt)* can be inherited from higher organization associated to the service provider. This organization is in charge of adapting the MPLS domain.
- *Role*: abstraction of the path and forwarding behaviour. They reflect different level of QoS provided inside a single MPLS domain. For instance, suspicious paths provide degradable quality inside the MPLS domain.
- *Activity*: abstraction of the operations that can be performed on paths and forwarding behaviours. Such abstraction can be seen as a modification or removal of specific paths.
- View: abstraction of resources presented in the network. We call it network resources. Such abstract resources include nodes, links, bandwidth, scheduling, etc.

3.3 Performance Contexts

We consider the scenario adopted in [16]. We assume three different performance contexts. The default context corresponds to a stable core network. The critical context reflects network critical phase. The saturation context corresponds to network saturation phase. The default context consists of the *long-term* strategies established in the planning process. The critical and saturation context are the triggers of short-term strategies of the reaction process. These contexts are initiated by performance alerts sent by network monitoring tools.

For instance, when a performance alert $Alert_i$ is generated signalling a saturation phase, a new sub-organization under the *DomainAdapt* — called saturation domain adaptation and denoted as $SatDomainAdapt_i$ — is created to manage it (cf. Listing 1.2 in Appendix A). The saturation assessment context $SatAssContext$ is activated in $SatDomainAdapt_i$ to manage the performance alert triggering a network saturation phase. This context is activated for every triple *{subject, action, object}* with the reception of a performance alert (i.e. $Alert_i$) with a *network.status* attribute equal or equivalent to saturation (cf. Listing 1.3 in Appendix A).

3.4 Generation of Network Adaptation Rules

In the saturation phase, we consider that the service provider strategy consists on pointing the third level and second level suspicious paths to a *blackhole*. The following two network adaptation rules, based on OrBAC obligations, reflect this strategy:

$$security\_rule(obligation, DomainAdapt, TLSusPath, Modify, Blackhole,\\ SatAssessContext)$$

Concrete level	Definition	Attributes
Subject	Source	AS number, user ID, country, etc.
Action	MPLS path	ingress and egress router, identifier
Object	Flow	IP source + IP destination + [Protocol | SPort | DPort | . . .]

Table 3. Concrete entities

$$security\_rule(obligation, DomainAdapt, SLSusPath, Modify, Blackhole, SatAssessContext)$$

These security rule means that in the saturation performance context, third level and second level suspicious paths (i.e., set of MPLS paths) are rerouted to a *blackhole* capable node. The activated rules for alert $Alert_i$ are deleted when the performance context is deactivated (i.e., when the network load is stable) by destroying the organization $SatDomainAdmit_i$. As a result, the organization *DomainAdapt* is *rolled-back* to the *DefaultContext* (i.e., long-term strategies implemented during the planning process). A similar modelling approach is applied to the network critical phase.

4 Flow Admission Policy

The definition of suspicious flows and their mapping to the corresponding path and forwarding behaviour is done at the entry point of each domain, the ingress router. The configuration of an ingress router contains the policy enforcement that regulates the access of suspicious flows to a given MPLS domain resources.

Let us assume the following high-level policy statement: *any suspicious flow must be given a de-prioritized path and forwarding behaviour*. The ingress router is assumed to maintain this policy requirement by being a single point through which all communication between the networks and the MPLS domain must pass and get controlled. When a security alert is received with an assessment classification that maps to a threat context, this latter is activated. Moreover, a mapping between network alert attributes and concrete entities is established to define the newly discovered suspicious flows. The activation of the context and the definition of these concrete entities are performed into dynamic sub-organizations. A flow admission rule is activated in order to affect the suspicious flow to its routing and QoS scheme inside the MPLS domain. This security rule is turned into configuration rules on the MPLS ingress router.

4.1 Concrete Entities

Table 3 summarizes our proposed set of concrete entities. We assume the following entities:

– *Subject*: source identifier of a flow of packets. It can be the Autonomous System (AS) number of an Internet Service Provider (ISP), a country, a user ID, etc.
– *Action*: the MPLS path characterized by its identifier and the ingress and egress routers.
– *Object*: any suspicious flow of packets. We characterize such flows by their IP destination, IP source, port source, and port destination, etc.

Abstract level	Definition	Examples
Role	origin	*customer*, *outsider*
Activity	path and forwarding behaviour	*gold_path*, *suspicious_path*
View	session	*VoIP_session*, *BestEffort_session*

Table 4. Abstract entities associated to the *DomainAdmit* organization

4.2 Abstract Entities

Table 4 summarizes our proposed set of abstract entities. We assume the following entities in the flow admission policy:

– *Organization*: *Domain admit (DomainAdmit)*, which in turn can be inherited from higher level organizational structures, such as the *organization* associated to an ISP network in charge of the affected MPLS domains.
– *Role*: abstraction of the origin of traffic flows. For instance, customers of the ISP subscribed to certain QoS services, or outsider customers sharing the resources of the ISP.
– *Activity*: abstraction of the path and forwarding behaviour. They reflect different level of QoS provided inside a single MPLS domain.
– *View*: abstraction of traffic flow. Such abstraction can be seen as session, characterized by destination port numbers, such as VoIP sessions, best effort sessions, or by certain predefined IP addresses such as critical sessions.

4.3 Threat Contexts

We model the management of threat contexts based on the construction of the original HADEGA proposal presented in [16]. This way, the contexts are based on the alert attributes: Impact Level (IL), and Confidence Level (CL). Table 5 shows an example based on such construction.

We assume the reception of security alerts. Each alert transports diagnosis data: assessment and network attributes. A new sub-organization under the *DomainAdmit* is created to manage each alert. For instance, the sub-organization $FLDomainAdmit_j$ manages the alert $Alert_j$ (cf. Listing 1.4 in Appendix A).

The first level assessment context, denoted as *FLAssessContext*, is activated in the $FLDomainAdmit_j$ context to manage a given alert $Alert_j$ if the definition matches the classification of the alert. The context is active for every triple {*subject, action, object*}. The classification of the alert is inferred from its assessment attributes (i.e. IL, and CL). For instance and in case of IDMEF alerts [16], the first level assessment context is reported by alerts with (1) an *impact.severity* low or medium and (2) a *confidence.rating* low (cf. Listing 1.5 in Appendix A).

We introduce an additional abstract entity, *view*, in this sub-organization. We call it first level suspicious flow and denoted as *FLSusFlow*. The mapping between the alert $Alert_j$ and the *FLSuSFlow* is done through the view definition. The definition of the flow, is inferred from the network attributes of the alert (i.e., IP source, IP destination, port source, etc.). For example and in case of an IDMEF alert, the target.address field provides the IP destination of the flow (cf. Listing 1.6 in Appendix A).

Context \ Assessment Attributes	IL	CL
First level assessment	low	low
	med	low
Second level assessment	low	med
	low	high
	med	med
	high	low
Third level assessment	med	high
	high	med
	high	high

Table 5. Context definition, based on the Impact Level (IL) and Confidence Level (CL) alert attributes.

4.4 Generation of Flow Admission Rules

The generation of flow admission rules consists on defining security rules for each context. The *FLAssessContext* is now active, security rule associated with this context is triggered. The following security rule matches this context:

$$security\_rule(permission, DomainAdmit, Any, FLSusPath,\\ FLSusFlow, FLAssesscontext)$$

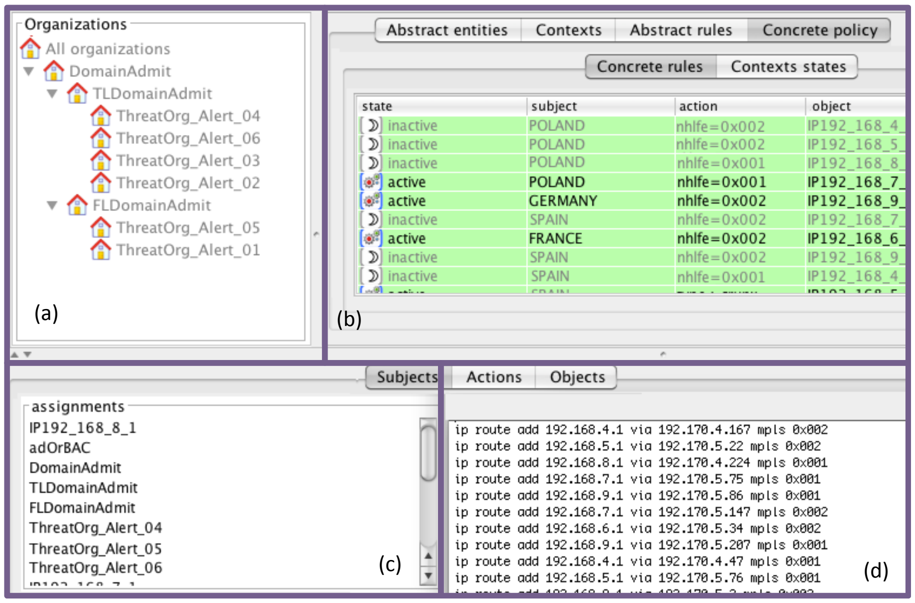
This permission rule means that in the threat context first level assessment, any first level suspicious flow is affected to the previously established first level suspicious path *FLSusPath*. When the flow is not suspicious any more, the threat context is deactivated by simply deleting the organization $FLDomainAdmit_j$. By destroying this organization, all related entities disappear and, therefore, the flow receives back a normal treatment. Similar modelling is applied to the second and third level assessment contexts.

5 Implementation

We present in this section a practical implementation of our approach. It is based on the open source MotOrBAC framework [2]. The ongoing prototype already allows the specification of our modelling approach, and its transformation into security rules for MPLS-linux routers [3]. From an implementation point of view, the reaction policies (both flow admission and network adaptation policies) can be executed in the same way but with different entity and organization definition. The flow admission control is more complicated because it involves the creation of dynamic entities in the sub-organizations and invokes mapping a long list of attributes from alerts to policies. We overview in this section the implementation of the flow admission reaction policy. A sample screen-shot of such an implementation is shown in Figure 2. We show in the screen-shot how a concrete OrBAC policy, instantiated via a series of IDMEF alerts, is processed and transformed into MPLS-linux configuration rules. In the sequel, we detail the specific steps associated to the enforcement depicted in Figure 2.

5.1 Policy Instantiation via Mapping of Alerts and Policies

The process for mapping alerts and policies relies on the use of the OrBAC API, available at the MotOrBAC website [2]. All the necessary functions for the definition of threat organizations and dynamic abstract entities are directly obtained via such an API. The combination of threat organizations, dynamic abstract entities and remainder OrBAC elements (as defined in Section 4) enable the complete specification of our policy modelling and corresponding predicates (e.g., permissions). The mapping between alerts, threat organizations and dynamic entities is done via XML and XPath methods already available in OrBAC API. Once established, the policy instantiation engine obtains the complete list of security attributes, activates the list of threat contexts, and instantiates the abstract entities. Finally, and based on the inference engine provided by the OrBAC API, the complete series of concrete rules are generated and provided as input to the transformation component of the prototype.



All organizations
DomainAdmit
TLDomainAdmit
ThreatOrg_Alert_04
ThreatOrg_Alert_06
ThreatOrg_Alert_03
ThreatOrg_Alert_02
FLDomainAdmit
ThreatOrg_Alert_05
ThreatOrg_Alert_01

state	subject	action	object
inactive	POLAND	nhlfe=0x002	IP192_168_4_
inactive	POLAND	nhlfe=0x002	IP192_168_5_
inactive	POLAND	nhlfe=0x001	IP192_168_8_
active	POLAND	nhlfe=0x001	IP192_168_7_
active	GERMANY	nhlfe=0x002	IP192_168_9_
inactive	SPAIN	nhlfe=0x002	IP192_168_7_
active	FRANCE	nhlfe=0x002	IP192_168_6_
inactive	SPAIN	nhlfe=0x002	IP192_168_9_
inactive	SPAIN	nhlfe=0x001	IP192_168_4_

assignments
IP192_168_8_1
adOrBAC
DomainAdmit
TLDomainAdmit
FLDomainAdmit
ThreatOrg_Alert_04
ThreatOrg_Alert_05
ThreatOrg_Alert_06

```
ip route add 192.168.4.1 via 192.170.4.167 mpls 0x002
ip route add 192.168.5.1 via 192.170.5.22 mpls 0x002
ip route add 192.168.8.1 via 192.170.4.224 mpls 0x001
ip route add 192.168.7.1 via 192.170.5.75 mpls 0x001
ip route add 192.168.9.1 via 192.170.5.86 mpls 0x001
ip route add 192.168.7.1 via 192.170.5.147 mpls 0x002
ip route add 192.168.6.1 via 192.170.5.34 mpls 0x002
ip route add 192.168.9.1 via 192.170.5.207 mpls 0x001
ip route add 192.168.4.1 via 192.170.4.47 mpls 0x001
ip route add 192.168.5.1 via 192.170.5.76 mpls 0x001
```

Fig. 2. Prototype system developed under the MotOrBAC framework. (a) Dynamic organizations created upon reception of IDMEF alerts. (b) Concrete permissions inferred from active contexts. (c) Concrete entities and alert attribute containers. (d) Transformation results, displaying the final MPLS-linux configuration rules.

5.2 From Inferred Rules to MPLS Configurations

The translation of concrete rules into MPLS-linux routers' configurations has been implemented as an OSGi bundle plug-in [1] for MotOrBAC. The plug-in receives as input the instantiated policy and collaborates with the OrBAC API to generate concrete MPLS routers instructions. The transformation engine relies on the concept of classes and attributes already provided by the OrBAC API. We described and encapsulated into generic OrBAC definitions all the complete network semantic required by the reaction policy. We also encoded a translator into a Java Class. This translator is responsible of generating MPLS-linux routers' configurations. The transformation engine parses the concrete rules and generates the configurations adapting to the mitigation strategy.

6 Discussion and Related Work

The adaptive policy-based framework proposed in this paper has a dual management aspects: the network management through implementation of network adaptation rules, and the security management through the flow admission rules.

Most of existing work on network QoS-based policy management [33, 31, 19, 6] does not support policy rules that can be dynamically triggered by events. Moreover, the work of IETF policy specification [19, 6] is based on directories to store policies but not for grouping the entities involved in the policies. In another word, it does not have the concepts of subjects and targets to specify to which components the policy applies. The work of [29, 33, 31] aimed more specifically on the management of DiffServ network solely. The work whose motivation is close to ours was proposed in [23, 24] to specify the network QoS policy. While this work provided an adaptive framework to answer events on the network level, the abstraction of different entities invoked in the policies was absent due to the usage of Ponder language [10]. In some policies' definition the action and its target were concrete and clear, in some other their definitions remained very ambiguous. Moreover, there was a mixing between the Policy Enforcement Point and the subject entity of the policy. Through the *obligation* security rule, we used the OrBAC to model network management policies. We defined a well-structured two-level grouping using abstract and concrete entities; thanks to OrBAC model [21]. It completely distinguishes between the Policy Enforcement Point on which we implement the configurations and the subject/target on which we are supposed to apply the policy. The model provides answer to adaptive changes on network level. It supports as well the *roll-back* and the update of normal context i.e., *long-term* strategies modification.

Concerning the security management scheme, most of existing work addressed the management of firewalls for the simple reason that they form the principal network security component [18, 8, 15]. In this paper, we proposed a management framework for controlling the admission of flows to the MPLS domain through the *permission* security rule. Therefore, the ingress MPLS router of the domain is seen as a security component. While this work is considered the first assuming the MPLS routers as a security components, there were some works that addressed mapping the traffic specification e.g., Service Level Specification (SLS) assignments into certain established QoS scheme inside the MPLS domain such as [6, 33]. Differently from this work, we provided an adaptive framework for handling alerts and map its diagnosis data into certain flow classification and QoS scheme. The model took in consideration the SLSs by providing two entities that abstract source of flows e.g., *gold customer*, and the session type e.g., *voice session*. Moreover, the use of the dynamic sub-organization concept provided the possibility to create views for the suspicious flows. Therefore, the *roll-back* of suspicious flows to the normal treatment was simply performed by deleting the given sub-organization.

7 Conclusion

We have introduced an adaptive policy-based framework for handling suspicious flows via MPLS policies. The framework builds upon the OrBAC formalism. The result is a top-down enforcement of mitigation policies and its automatic transformation into MPLS router configuration rules. The framework is divided into two aspects: flow admission and network adaptation control. In each aspect, different modelling was established. We have also presented the implementation of our approach for the generation of configuration rules for MPLS-linux routers triggered by IDMEF alerts and OrBAC policies. Future work will aim on managing the conflicts that can be originated from the selected reaction policies, as well as reducing the complexity in suspicious flows definition; by including topology-related attributes in dynamic organizations definitions. We will also study more complex policies by introducing a list of SLSs.

Acknowledgements: Research partially supported by the European Commission, in the framework of the FP7 DEMONS project (Grant agreement no. FP7-257315); and the Spanish Ministry of Science and Innovation (projects TSI2007-65406-C03-03 E-AEGIS, TIN2011-27076-C03-02 CO-PRIVACY, CONSOLIDER INGENIO 2010 CSD2007-0004 ARES, and TIN2010-15764 N-KHRONOUS). We would also like to thank F. Autrel, F. Cuppens, N. Cuppens-Boulahia, X. Rimasson and several other members of the DEMONS project, for their help and for fruitful discussions on the topic of this paper.

References

1. Eclipse. The Eclipse Foundation open source community website. `http://www.eclipse.org/`.
2. MotOrBAC: an Open-Source OrBAC Policy Editor. `http://motorbac.sourceforge.net/`.
3. MPLS for Linux. `http://mpls-linux.sourceforge.net/`.
4. F. Autrel, N. Cuppens-Boulahia, and F. Cuppens. Reaction Policy Model Based on Dynamic Organizations and Threat Context. In *Proceedings of the 23rd Annual IFIP WG 11.3 Working Conference on Data and Applications Security XXIII*, pages 49–64, Berlin, Heidelberg, 2009. Springer-Verlag.
5. D. Awduche, J. Malcolm, J. Agogbua, M. O'Dell, and J. McManus. Requirements for Traffic Engineering Over MPLS. RFC 2702 (Informational), September 1999.
6. M. Brunner and J. Quittek. MPLS Management using Policies. In *International Symposium on Integrated Network Management Proceedings, 2001 IEEE/IFIP*, pages 515 –528, 2001.
7. F. Cuppens and M. Alexandre. Modelling Contexts in the Or-BAC Model. In *Proceedings of the 19th Annual Computer Security Applications Conference*, ACSAC '03, pages 416–, Washington, DC, USA, 2003. IEEE Computer Society.

8. F. Cuppens, N. Boulahia-Cuppens, T. Sans, and A. Miege. A Formal Approach to Specify and Deploy a Network Security Policy. *IFIP International Federation for Information Processing*, 173:203–218, 2005.
9. F. Cuppens, N. Cuppens-Boulahia, and A. Miege. Inheritance Hierarchies in the OrBAC Model and Application in a Network Security Environment. In *Second Foundations of Computer Security Workshop (FCS'04)*, 2004.
10. N. Damianou, N. Dulay, E. Lupu, and M. Sloman. The Ponder Policy Specification Language. In *Proceedings of the International Workshop on Policies for Distributed Systems and Networks*, POLICY '01, pages 18–38, London, UK, UK, 2001. Springer-Verlag.
11. H. Debar, D. Curry, and B. Feinstein. The Intrusion Detection Message Exchange Format (IDMEF). RFC 4765 (Experimental), March 2007.
12. H. Debar, Y. Thomas, F. Cuppens, and N. Boulahia-Cuppens. Using Contextual Security Policies for Threat Response. In *Detection of Intrusions and Malware and Vulnerability Assessment (DIMVA2006)*, volume 4046, pages 109–128. Springer, 2006.
13. H. Debar, Y. Thomas, F. Cuppens, and N. Boulahia-Cuppens. Enabling Automated Threat Response through the Use of a Dynamic Security Policy. *Journal in Computer Virology*, 3(4):195–2010, 2007.
14. F. Le Faucheur, L. Wu, B. Davie, S. Davari, P. Vaananen, R. Krishnan, P. Cheval, and J. Heinanen. Multi-Protocol Label Switching (MPLS) Support of Differentiated Services. RFC 3270 (Proposed Standard), May 2002. Updated by RFC 5462.
15. J. Garcia-Alfaro, F. Cuppens, and N. Cuppens-Boulahia. Aggregating and Deploying Network Access Control Policies. In *Proceedings of the The Second International Conference on Availability, Reliability and Security*, ARES '07, pages 532–542, Washington, DC, USA, 2007. IEEE Computer Society.
16. N. Hachem, H. Debar, and J. Garcia-Alfaro. HADEGA: a Novel MPLS-based Mitigation Solution to Handle Network Attacks. In *Performance Computing and Communications Conference (IPCCC), 2012 IEEE 31st International*, pages 171 –180. IEEE, dec. 2012.
17. W. Han and C. Lei. Survey Paper: a Survey on Policy Languages in Network and Security Management. *Computer Networks*, 56(1):477–489, January 2012.
18. A. Hassan and L. Hudec. Role Based Network Security Model: A Forward Step towards Firewall Management. In *Workshop On Security of Information Technologies*, 2003.
19. K. Isoyama, M. Brunner, M. Yoshida, J. Quittek, R. Chadha, G. Mykoniatis, A. Poylisher, R. Vaidyanathan, A. Kind, and F. Reichmeyer. Policy Framework MPLS Information Model for QoS and TE. IETF Internet Draft – expired 01, December 2000.
20. L. Kagal. Rei: a Policy Language for the Me-Centric Project. Technical report, HP labs, 2002.
21. A. Abou El Kalam, R. El Baida, P. Balbiani, S. Benferhat, F. Cuppens, Y. Deswarteand, A. Miege, C. Saurel, and G. Trouessin. Organization Based Access Control. In *4th International Workshop on Policies for Distributed Systems and Networks (Policy 2003)*, pages 120–131. IEEE, 2003.
22. J. Lobo, R. Bhatia, and S. Naqvi. A Policy Description Language. In *Proceedings of the sixteenth national conference on Artificial intelligence and the eleventh Innovative applications of artificial intelligence conference innovative applications of artificial intelligence*, AAAI '99/IAAI '99, pages 291–298, Menlo Park, CA, USA, 1999. American Association for Artificial Intelligence.
23. L. Lymberopoulos, E. Lupu, and M. Sloman. An Adaptive Policy based Management Framework for Differentiated Services Networks. In *Proceedings of the 3rd International Workshop on Policies for Distributed Systems and Networks (POLICY'02)*, POLICY '02, pages 147–158, Washington, DC, USA, 2002. IEEE Computer Society.
24. L. Lymberopoulos, E. Lupu, and M. Sloman. An Adaptive Policy-based Framework for Network Services Management. *J. Netw. Syst. Manage.*, 11(3):277–303, September 2003.

25. E. Rosen, A. Viswanathan, and R. Callon. Multiprotocol Label Switching Architecture. RFC 3031 (Proposed Standard), January 2001.
26. P. Samarati and S. Vimercati. Access Control: Policies, Models, and Mechanisms. In *Revised versions of lectures given during the IFIP WG 1.7 International School on Foundations of Security Analysis and Design on Foundations of Security Analysis and Design: Tutorial Lectures*, FOSAD '00, pages 137–196, London, UK, UK, 2001. Springer-Verlag.
27. R. Sandhu, E. Coyne, H. Feinstein, and C. Youman. Role-Based Access Control Models. *Computer*, 29(2):38–47, February 1996.
28. M. Sloman. Policy Driven Management For Distributed Systems. *Journal of Network and Systems Management*, 2:333–360, 1994.
29. Y. Snir, Y. Ramberg, J. Strassner, R. Cohen, and B. Moore. Policy Quality of Service (QoS) Information Model. RFC 3644 (Proposed Standard), November 2003.
30. Sophos. *Security Threat Report 2012* , 2012.
31. G.N. Stone, B. Lundy, and G.G. Xie. Network Policy Languages: a Survey and a New Approach. *Network, IEEE*, 15(1):10 –21, jan/feb 2001.
32. The OASIS technical commitee. XACML: eXtensible Access Control Markup Language, 2005.
33. D. Verma, M. Beigi, I. Beigi, and R. Jennings. Policy based SLA Management in Enterprise Networks. In *Proc. Policy 2001: International Workshop on Policies for Distributed Systems and Networks*, pages 137–152. Springer-Verlag, 2001.

A Sample OrBAC Security Rules used in HADEGA

Listing 1.1. Roles assignment

```
empower(org, subject, role): means that in organization
org, subject is empowered in role.

consider(org, action, activity): means that in
organization org, action is considered an implementation
of activity.

use(org, object, view): means that in organization org,
object is used in view.
```

Listing 1.2. Performance context management

```
performance_context_management($Alert_i$,$SatDomainAdapt_i$)
```

Listing 1.3. Activation of saturation performance context

```
hold($SatDomainAdapt_i$,-,-,-,SatAssessContext)
  ∧ threat_context_management($Alert_i$,$SatDomainAdapt_i$)
  ∧ $Alert_i$(network.status)
  ∧ network.status=saturation
```

Listing 1.4. Threat context management

```
threat_context_management($Alert_j$,$FLDomainAdmit_j$)
```

Listing 1.5. Activation of first level assessment context

```
hold($FLDomainAdmit_j$,-,-,-,FLAssessContext)
  ∧ threat_context_management($Alert_j$,$FLDomainAdmit_j$)
  ∧ $Alert_j$(Assessment)
  ∧ (Impact(Assessment, 'IL') ∧ (IL=low ∨ IL=medium))
  ∧ (Confidence(Assessment, 'CL') ∧ CL=low)
```

Listing 1.6. Mapping between the IDMEF alert and View entity

```
use($FLDomainAdmit_j$, flow, FLSusFlow)
  ∧ threat_context_management($Alert_j$, $FLDomainAdmit_j$)
  ∧ $Alert_j$(Source, Target)
  ∧ (Address(Source, 'IP_Src')
  ∧  flow.IP_Source = 'IP_Src')
  ∧ (Address(Source, 'Port_Src')
  ∧  flow.Port_Source = 'Port_Src')
  ∧ (Address(Target, 'IP_tgt')
  ∧  flow.IP_Destination = 'IP_tgt')
  ∧ (Address(Target, 'Port_tgt')
  ∧  flow.Port_Destination = 'Port_tgt')
```