Towards Policy Refinement for Resilience Management in Cloud

Syed Noorulhassan Shirazi[1], Steven Simpson[1], Kanza Noor Syeda[2], Andreas Mauthe[1] and David Hutchison[1]
[1]InfoLab21, School of Computing and Communications, Lancaster University, LA1 4WA UK
Email: {n.shirazi, s.simpson, a.mauthe, d.hutchison}@lancaster.ac.uk
[2]Department of Entrepreneurship, Strategy & Innovation, Lancaster University Management School, LA1 4YX, UK
Email:k.syeda@lancaster.ac.uk

***Abstract*—Cloud computing is becoming increasingly important for provision of critical services because of potential cost saving, scalability and elasticity. Therefore, it is particularly important for clouds and cloud-based services to be resilient, i.e., they are able to operate correctly and continuously even in the presence of challenges. To do this, a number of resilience-supporting mechanisms are needed at various levels in cloud infrastructure. It is non-trivial to manage these mechanisms and there is a need for flexible instruments which assist cloud providers in this complex task. Policy based management is an established instrument to manage resilience supporting mechanisms and they are useful if it allows not only high level description of abstract policy (e.g high level security and resilience requirements), but also enables such policy to be refined and eventually mapped into an appropriate low levels in cloud settings. This paper sheds light on basic concepts behind policy based management in cloud, more specifically it emphasizes the use of policy refinement which is the process of translating higher level requirements (such as security and resilience requirements) into the sequence of actions at lower levels that can implement them, in order to generate more refined policies that govern the behaviour of an overall cloud system when under challenge. We finally present two example scenarios on how policy refinement can work for the cloud to establish its relevance for the overall resilience management.**

***Index Terms*—Resilience, policy refinement, cloud computing.**

I. Introduction

The Cloud Computing is a major trend, which will likely reach the IT services that support critical infrastructures , because of the potential cost savings, scalability and elasticity [1], [2]. As many critical services are moving to cloud, it is of paramount importance that cloud administrators are able to effectively manage the underlying technologies which enable configurations, elasticity, dynamic invocation of services, and monitoring activities such that security and resilience requirements of the critical service users and providers are met. Moreover, making sure that the management activities do not disturb critical service delivery, the providers must be able to easily change the behaviour of the system they are managing.

This work is sponsored by UK-EPSRC funded TI3 project, grant agreement no. EP/L026015/1: A Situation-aware Information Infrastructure; and the European Union FP7 Project SECCRIT (Secure Cloud Computing for Critical Infrastructure IT), grant agreement no. 312758.

Management and resilience of the cloud environments are closely linked. The extra layer of resource virtualization makes it difficult to plan effective management due to varying user demands, co-hosted VMs and the arbitrary deployment of multiple applications. Generally, management policies are used to govern the behaviour of a system. These management policies can be mostly looked upon as: *"the constraints and preferences on the state or the state transition, of a system and is a guide on the way to achieving the overall objective which itself is also represented by a desire system state"* [3]. When using policy based management it is critical that rules being specified actually stem out of the higher level requirements and they are implementable. This process of going from high level requirements to low level policy implementation is hereafter refer as *"policy refinement"* and main subject of this paper.

The refinement process involves stages of decomposition, operationalization, deployment and re-refinement, and operates on policies expressed in a logical language flexible enough to be translated into many different enforceable configurations [4] as depicted in Fig.1 below.

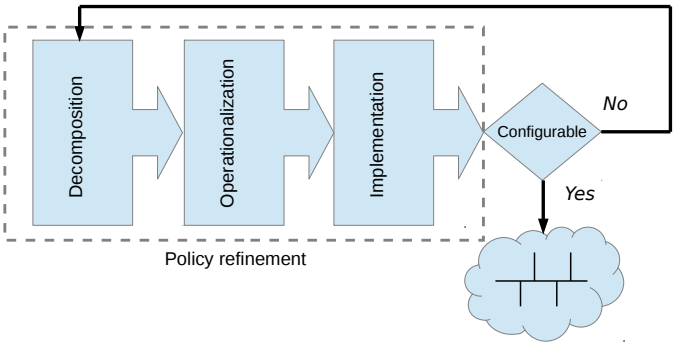

Fig. 1: Policy refinement stages

Decomposition is process of obtaining rules which provide information about how an actions is to be implemented. After decomposition stage, reference to new policies are introduced which need to be operationalized in specific environment. Thus, after an initial operationalization, refinement is carried out, till policies are expressed in terms which policy enforcement point understands and can be implemented. It is desirable to maintain the properties of consistency and completeness,

when refining an abstract high level requirements into a set of more concrete policies. The former means that there should not be any conflict and later means fully implementable policies. In order to maintain these properties, the policies must be fine-tuned, and they should respond to changes in the system.

In context of the Cloud computing, policy based management are becoming effective because they allow the separation of the rules that govern the behavioural choices of a system from the functionality provided by that system. As shown in previous work [5], [6], [7] it is possible to adopt the behaviour of a system without the need to re-design and develop any of the functionality, and changes can be applied without the need to bring system to halt which is crucial for critical service point of view. However, there is still need to apply basic of policy refinement principals for autonomic management in cloud computing. The motivation for the work presented here comes from the previous work [8], [9] and objectives of the project SECCRIT[1] which state:

- to develop a suitable policy vocabulary, policy mechanisms, and a policy editor that supports the user in specifying his security requirements and in deriving the resulting, machine enforceable security policy.
- to develop concepts for policy deployment and redeployment to the cloud in a secure manner.

A. Basics of policy management

Policies are rules which govern the choices in behaviour of the system. In IETF[2], there is an active policy working group which show recognition of this approach for system management. Mainly policies are categorised as obligation and authorisation polices which are also supported by Ponder2[3] which is policy environment used by many in both industry and academia.

Obligation policies specify management operation that must be performed when a particular event occurs given some supplementary conditions being true. These policies follow the Event-Condition-Action (ECA) paradigm and are of the form:

```
on <event>
        if  <conditions>
                do <target> <action>;
```

Therefore, the occurrence of the specific event is a necessary condition for the mandated operation to be performed. The event is a term of the form $e(a_1, \ldots a_n)$, where e is the name of the event and $a_1, \ldots, a_n$ are the names of its attributes. The condition is a boolean expression that may check local properties of the nodes and the attributes of the event. The target is the name of a role (i.e., a placeholder) where the action will be executed and so the service or resource assigned to the target role must support an implementation of the action. The action is a term of the form $a(a_1, \ldots, a_m)$, where a is the name of the action and $a_1, \ldots, a_m$ are the names of its attributes. To simplify notation an obligation policy can have a list of target-action pairs, all evaluated when the event is true and the condition holds. The attributes of an event may be used for evaluating the condition (to decide whether to invoke the action or not), or they may be passed as arguments to the action itself. Implicitly the role to which the obligation policy belongs is the subject of the obligation i.e. the entity enforcing the policy, and the action is invoked on a target role. Note the target may be the same as the subject i.e. a role may perform actions on itself. Obligations can also be used to load other policies (obligations or authorisations) into the system or existing policies may be enabled/disabled to change the management strategy at run-time.

Authorisation policies specify what actions a subject is allowed (positive authorisation) or forbidden (negative authorisation) to invoke on a target. The subject and the target are role names. The action and the condition are defined like in obligations. Authorisation decisions could be made by one or more specific roles in the network, but commonly implementations are based on the target making decisions and enforcing the policy as it is assumed that target roles wish to protect the resources they provide to the network.

```
auth[+/-] <subject>
        if <condition>
                then <target><action>;
```

The paper is organised as follows. Section II describes related work. The policy based resilience management is discussed in Section III, and policy refinement process is explained in Section IV. Section V presents example scenarios and Section VI finally concludes the chapter.

II. Related work

Policies specifies management operations that must be performed when a particular event occurs given that some conditions being true. The policy based management has proven to be very effective for complex system management as evident in previous literature. In [10], authors briefly mention that due to the opacity of cloud service provisioning, cloud based services could benefit from context-awareness policies in general, but they do not explicitly mention the issue of policy refinement and enforcement.

Goyal et al. [11] propose a policy-based event-driven services-oriented architecture (PESA). In PESA, policies are a set of rules that control behaviour and usage of services. The services are managed by hierarchically distributed controllers named mediators. Their architecture focuses on the service layer, more specifically, on the enterprise service bus (ESB)[4]. A relation to the cloud is indicated, as such services may run within the cloud, but their framework is not truly cloud-specific.

In [12], Hamlen et al. propose a policy-compliant data processing framework that contains different modules for policy decision and enforcement, such as a policy reasoning module. They propose to have control over data by replacing specific instructions (e.g., read method invocations) by surrounding the code with their own check (i.e., intercepting

[1] www.seccrit.eu

[2] http://datatracker.ietf.org/wg/policy/charter/

[3] http://ponder2.net/cgi-bin/moin.cgi/Ponder2Overview

[4] http://www.oracle.com/technetwork/articles/soa/ind-soa-esb-1967705.html

the method invocation and asking the reasoning module for a decision) before further processing. The policy-based approach to network and system management proposed in [13], [14] defines a framework for management of polices, policy hierarchies and policy transformation. Ross et al. [15] propose to enrich managed objects with policy goals as required by the management policy. They describe policies in two parts, an active part, containing application specific functionality, and a passive part which can be re-used without any change. Kaikini et al. [16] use an approach to enforce policies by means of rules, but the understanding of a rule is more restrictive.

The above mentioned works describe solutions for the specification and implementation of policies and recently SLAs, but little focus has been given to the refinement of high-level requirements into low-level polices. Verma [17] presented an approach to policy translation that is based on a set of tables. The tables identify the relationships between users, applications, servers, routers and classes of service supported by network. Whilst this technique offers the advantage of being fully automated, it is a inflexible approach, only supporting a very specific type of high-level SLA policy and low-level device configuration policy.

The work in [18] outlines a policy authoring environment that provides a policy tool, called POWER, for refining policies. A domain expert first develops a set of policy templates, expressed as Prolog programs, then the policy authoring tool uses an integrated inference engine that interprets these programs to guide the user in selecting the appropriated elements from the management information model to be included in the final policy. The main limitation of this approach is the absence of any analysis capabilities to evaluate the consistency of the refined policies. Similarly, work presented in [19] allows the translation of service-level objectives into configuration parameters of a managed system. The transformation engine takes the service requirements of the user as input, and searches the database to determine the optimal parameter values that provide a required level of service. A limitation of this technique is its dependence on a sufficiently rich database which is only possible by observing the system for some period of time. It is also unable to deal with situations where a given requirement specification results in different configurations.

The goal elaboration approach for policy refinement has recently received more interest. The approach is based on the work of Darimont et al. [20]. KAOS[5] is a formal approach to goal refinement operationalisation aimed at providing constructive formal support while hiding underlying complexities. The idea behind this approach is to reuse generic refinement patterns from a library structured according to strengthening/weakening relationships among patterns. However, it has limitations in the assignment decision step, and provides very little support for formal reasoning about alternative assignments. Bandara et al. [21] presented an approach to policy refinement by which formal representation of a system based on the Event Calculus is used in conjunction with abducting-reasoning techniques to derive the sequence of operations that will allow a given system achieve a desired goal. The authors in [22] investigated the use of model checking in order to derive operational policies from low-level goals. The refinement framework relies also on the KAOS method, the use of temporal logic, and modelling of system behaviour in terms of event-based labelled transitions. However, the approach does not consider feedback mechanisms to analyse the impact of high-level goals with respect to managing system performance and behaviour.

[5]http://www.objectiver.com/fileadmin/download/documents/KaosTutorial.pdf

III. Policy based resilience Management

As discussed earlier, resilience becomes of paramount importance to the cloud users and providers as well as critical service providers to ensure correct and continuous system operation even in the presence of challenges [23]. We then re-interpret and define cloud resilience as, *"the ability to maintain an acceptable level of system operation and service even in the presence of challenges."* To provide resilience in cloud environments, a number of mechanisms are needed at various layers (tenant & infrastructure) and locations, such as traffic capturing, anomaly detection and classification. However, it is difficult how these mechanisms should be co-ordinated to mitigate challenges. It is also difficult to define how the configuration of such mechanisms should change over time, in response to new types of challenges or new set of high level requirements. This makes role of problem refinement more evident so that resilience strategy can be modified without interrupting critical service operation. Reconfiguration strategies are represented as policies, which define how the operation of the several components in the cloud should be modified in response to pre-specified events. Such decisions may concern the tuning of parameters of the mechanisms, the re-wiring of their interconnections, and also the dynamic enabling or disabling of the mechanisms currently deployed in a particular strategy.

At any time, the current set of mechanisms may generate new events which in turn may trigger a different set of policies. This continuous process will constitute a policy-driven feedback control-loop, in which events trigger policies for the reconfiguration of resilience mechanisms, which may in turn generate other events that will trigger different policies and so on. The set of policies defining the possible reconfiguration actions is not fixed, and different policies may be loaded or unloaded over time to reflect better resilience practices or a better understanding of the challenges. The framework built in previous work and evaluated in *ResumeNet*[6] based on work by [24], whereby a number of resilience principles are defined, including the resilience strategy $D^2R^2 + DR$: Defend, Detect, Remediate, Recover, Diagnose and Refine is outlined in Fig. 2 below. At its core is a control loop comprising a number of conceptual completions that realise the real time aspect of the $D^2R^2 + DR$ strategy and consequently implements cloud resilience. Based on the resilience control loop, other necessary elements of our framework are derived, namely a resilience metrics framework, an approach to understanding

[6]http://www.comp.lancs.ac.uk/resilience/

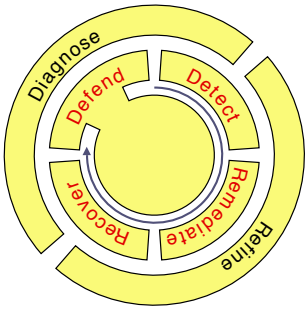

Fig. 2: The resilience strategy[24]

high-impact challenges that cloud may face, a policy driven resilience management, an incremental approach to challenge analysis that aim to build situational awareness, and multilevel information sharing and control mechanisms. Under the $D^2R^2 + DR$ framework, there must exist components capable of reconfiguring devices in response to challenges using policies. Reconfiguration need not apply to the same components on which the detection was based. A resilience engine is responsible for mapping detection events to reconfigurations, and may be (for example) a policy engine, accepting a resilience strategy as a collection of policies.

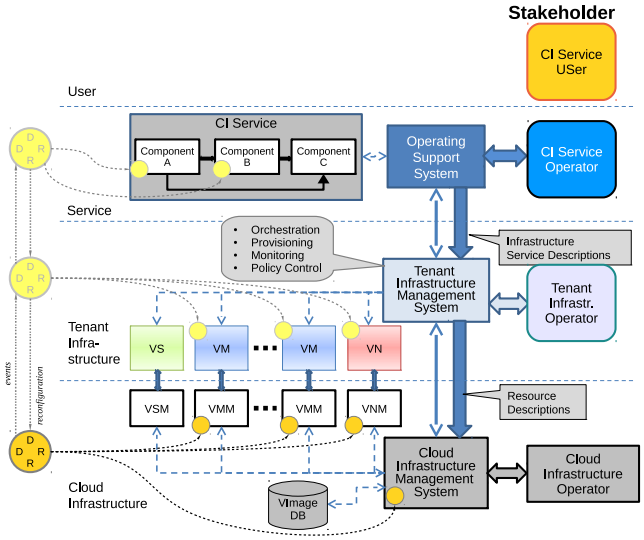

Fig. 3: A Resilience-oriented view

The SECCRIT consortium have developed an architectural model for deploying critical infrastructure services in the cloud which provide basis for the development of technical solutions such as policy based resilience management. We would refer reader to work [8], [25] for further details. We can apply D^2R^2 to the SECCRIT architectural framework to provide a resilience view of it (Fig. 3). At the physical layer, the cloud-infrastructure operator has access to physical nodes and network, which can be monitored to inform the detection process. The operator can also reconfigure these devices, in response to detected challenges using policies. Cloud-infrastructure D^2R^2 may exist as monitoring and reconfiguration points on physical hosts and networks, and on some virtual components. Resilience engines and detectors need not exist on any physical equipment used directly to provide virtual resources to the above layer. At the tenant-infrastructure layer, the tenant has access to VMs, and possibly virtual taps on VNs, which can inform detection. In response to challenges, the tenant may reconfigure the hosted machines, and some functionality of the virtual networks might also be exposed. Thus, tenant-infrastructure D^2R^2 is spread across components visible this layer. Within the inner D^2R^2 loop, some interaction between these layers may exist in the form of events and reconfigurability exposed by the lower layer. For details on policy and resilience view points we refer reader to SECCRIT architecture white paper [26].

IV. Policy Refinement Process

Policy refinement is the process of transforming a high-level, abstract policy specification into a low-level, concrete one in terms of operations of the system [27]. More formally policy refinement could be defined as follows:

Definition *If there exists a set of policies P_{ref}:$p_1, p_2, ...p_n$, such that the enforcement of a combination of these policies results in a system behaving in an identical manner to a system that is enforcing some base policy P_b, it can be said that P_{ref} is a refinement of P_b.* (see Fig. 4 below).

In [28] authors identify the main objectives of a policy refinement process as:

1) Determine the resources that are needed to satisfy the requirements of the policy.
2) Translate high-level policies into operational policies that the system can enforce.
3) Verify that the lower level policies actually meet the requirements specified by the high-level policy.

The first of these objectives involves mapping abstract entities defined as part of a high-level policy to concrete objects/devices that make up the underlying system. The second specifies the need to ensure that any policies derived by the refinement process be in terms of operations that are supported by the underlying system. The final objective requires that there be a process for incrementally decomposing abstract requirements into successively more concrete ones, ensuring that at each stage the decomposition is correct and consistent.

Table. I below summarizes simple refinement example as presented in [29], and slightly re-interpreted, more detailed examples will follow later in the Section V. In light of above example we need formal technique for refining high-level goals into more concrete ones; and finally a means of inferring the combination of operations that will achieve these concrete goals such of the work done in the [20] for refining goals into implementation specifications for policy refinement. They have proposed following properties of the goals for refinement process:

- **Correctness**: a refinement is said to be correct if the conjunction of all the members of that subset is a refinement of the base policy.
- **Consistency**: a refinement is said to be consistent if there are no conflicts between any of the policies in the refined policy set.

TABLE I: Policy refinement example

1. **High level requirement** • *only provide service X to a user if he logs in from location L*
2. **More refined requirements looks like** • ***provide*** *service* ***to*** *user on event* ***where*** *condition* where (service = X) $\wedge$ (event = user logs in) $\wedge$ (condition = from IP range = x.x.x.x && CN==L)
3. **Concrete implementation** • further steps will be required to provide concrete implementation for policy enforcement point (PEP)

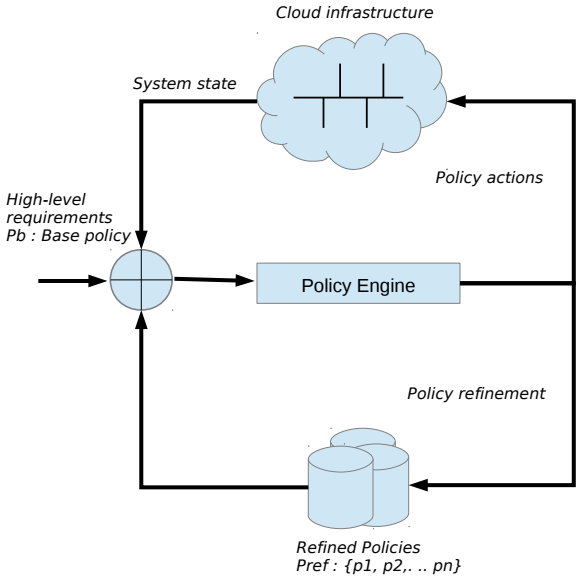

Fig. 4: A policy refinement process

- **Minimality**: a refinement is said to be minimal if it is correct and if removing any policy from the refined policy set causes the refinement to be incorrect.

So, an essential requirement when refining a policy is to ensure that the goal achieved by that policy would still be achieved by the set of sub-policies that it is refined into. In addition, a policy refinement can be said to be complete if all the properties defined above hold. However, in the policy refinement domain it may be acceptable to have a single policy that is a refinement of some base policy, provided that the refinement uses subjects, targets and actions that map to different physical entities.

Policy refinement techniques have received great research attention and few good techniques have been proposed which we briefly discuss in related work Section. II. When developing a policy refinement technique the main compromise is between the generality of the technique and the amount of automation. As show in in figure 5 below, the trade-offs means that techniques that are highly automated tend to have a very narrow domain of applications in which they work. On the other hand, techniques that have general applicability are not particularly automated.

V. Example scenarios

A. Availability of critical service

As discussed above the policy refinement is a process which convert the information from a policy description in

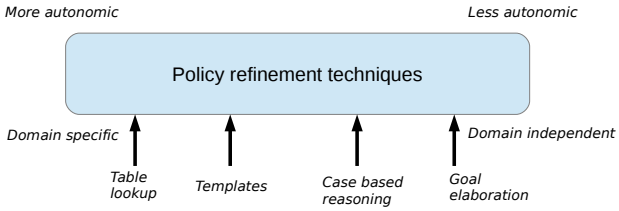

Fig. 5: Policy refinement techniques

the the form of a policy statement and context containing parameters/variables into a series of rules which govern the behaviour of the system. This process makes it dependent onto the underlying system because certain function calls required are technology dependent to configure actions such as migrating VM from one location to another need to invoke a call by interacting with specific API of a vendor such as management stack of OpenStack[7] or VMware[8]. Below we present an example for policy refinement where we consider cloud environment which consists of various servers hosting critical services for a tenant. The tenant in this case specifies requirements which says:

```
High level policy P1:
The cloud provider must ensure that the
infrastructure in which critical service is
hosted is available under high load and service
should not be co-located.
```

Before this policy can be applied, it require further decomposition and analysis, for example how availability is defined and what is acceptable rate of load and how it can be achieved. Also, which tenant to whom the service should not be co-located. In [30] describes the approach to that integrates the definition of a policy hierarchy to allow refining executable policies from high-level requirements systematically. A policy hierarchy is formed by a high-level policy which is then refined in several steps into lower-level policies. On lower level the resulting policies are called actions which can be carrier out without further analysis. The policy P1 can then be refined as follows:

```
Middle level policy P2:
The mean time between failure MTBF must be
minimised and load utilization need to be
reported and if affinity with tenant 'X' is
detected that need to be reported.
```

This policy is clearly more refined compared to Policy P1 and further refinement would lead to policy P3:

```
Refined policy P3:
If MTBF is greater than 0.3% or service request
```

[7]www.openstack.org
[8]www.vmware.com

```
queue length is greater than 0.8% than
instantiate another <VM> and if tenant X is to
be placed at same physical location as VM of
critical service then notification is sent.
The report is required daily at 1700HRS.
```

Policy P3 is on a level which allows the satisfaction of policy goals without further refinement and the policy is compiled into the following collection of rules:

```
E: Queue-->NewServicerequest(ServiceID, VMaddress)
 C: Queue<--Get(ServiceRequestInQueue) = 0.8 || Get(
     VMaddress!=CIServer)
  A: Queue<--Instantiate(newVMaddress,ServiceID)
     Queue<--TransferRequest(ServiceID, newVMaddress
         )
     Tenant<--sendNotification(LoadReport())
```

```
E: Server--> avgMTBF(CurrentMTBF)
    C: Server<--getMTBF(CurrentMTBF >0.3)
        A: Server<--Set(newServer,sendNotification(
            LoadReport())
```

```
E: Timer-->Alarm(time==1700)
    C: true
        A: Server<--Get(LoadReport)
           Tenant<--LoadReport(LoadReport)
```

B. Resilience against DoS

This example highlight the use of refined policies to response to challenge such as Denial of service (DoS) attack targeted at cloud infrastructure layer. In order to confront these challenges, a resilience mechanisms can be used that must co-ordinate and co-operate to ensure resilience. Clearly, it is important that an attack be mitigated rapidly to reduce the impact to the other tenants and protect the infrastructure. Such mechanism include but not limited to anomaly detector, flow classifier, resilience metrics reporter, malware differentiators etc.

Policies have been proven quite effective to configure and coordinate the interactions between these mechanisms. For example, specific root causes will require distinct remediation strategies, and, when a victim VM is identified as a possible DDoS attack, a preventive action may be applied. By having a resilience strategy implemented with the aid of policies, as opposite to having it hard coded, one can easily change it by adding or removing policies, thereby permitting the modification of the strategy during run-time. This is of particular importance to us, as strategies for resilience are subject to frequent refinements, due to changes in high level requirements. Following, we list a few obligation policies that can be used to reconfigure resilience services in response to events generated by monitoring mechanisms, anomaly detection systems and root cause analysis.

Listing 1: Example policies to remediate DoS attack

```
on highUtilisation(link, VM_ID)
        do FlowExporter enable(link, VM_ID) &&
            sandBox (VM_ID, newLocation);

//Flow exporter is disabled when link utilisation
    decreases nd VM will not be sandboxed for
    further analysis (P1).

on lowUtilisation(link, VM_ID)
    if (LocalManager.anomalyList isEmpty(link, VM_ID
        ))
        do FlowExporter disable(link, VM_ID);


//Policy for handling high risk alert (P2)
on highRisk (link, src, dst, VM_ID)
    do
     {
      FlowExporter notify(highRisk(link, VM_ID));
      LocalManager.anomalyList add(link, src, dst);
     }

//Policy for handling high risk alert (P3)
on highRisk (link, src, dst, VM_ID)
    if (LinkMonitor getUtilisation() >= 75%)
      do RateLimiter limit(link, 60%);

//Configure fine grained policy (P4: classification)
on classification(flow, value, confidence, VM_ID)
     if ((value == DDoS) && (confidence < 0.4))
      do
       {
        Visualisation notify(alert(high));
        RateLimiter limit(flow.src, flow.dest, x%);
       }
     if ((value == DDoS) && (confidence >= 0.4) && (
          confidence <= 0.8))
      do
       {
        Visualisation notify(alert(high));
        RateLimiter limit(flow.src, flow.dest, y%);
       }
     if ((value == DDoS) && (confidence > 0.8))
      do
       {
        Visualisation notify(alert(high));
        Firewall block(flow.src, flow.dest);
       }

//Configure local manager for handling low risk
    alert (P5:recovery)
on lowRisk (link, src, dst)
     if ((LocalManager.anomalyList remove(link, src,
          dst, VM_ID)) isEmpty(link))
      do
       {
        FlowExporter notify(lowRisk(link, VM_ID));
        RateLimiter limit(link, 100%);
       }
```

VI. Conclusions

Management and resilience of the cloud environments are closely linked. Resilience in cloud environments need to be managed to disseminate relevant policies to the cloud provider that will implement them. Policies specifies actions which are needed to deal with challenges and due to varying nature of challenges these policies need to be further refined to be adaptive in response to challenges. We have presented the recent work on refinement process for policies that has iterated phases of decompositions at its core. There are many avenues for future work in this discipline but we are particularly interested in policy based resilience management in the cloud. We want to explore how example polices e.g described in templates can be optimized with refinement process. We aim to merge our refinement procedures develop in course of this work with the policy analysis framework to see whether this can be used to guide the resilience against challenges such as DDoS attack.

There are a number of different conflicts that can arise from policies. For example, some policies will trigger complex management procedures which require the execution of actions that may be specified as part of different policies. Determining the existence of conflicting configurations for cloud environments is of critical importance. We assume that the system may be loaded with a number of policies to address many different challenges simultaneously. Multiple policies that need to coexist may specify conflicting actions on the same virtual resources, or may trigger the activation of incompatible mechanisms, thereby rendering a particular resilience strategy ineffective. We intend to investigate approaches for the automatic identification and resolution of policy conflicts. One possible approach around solution of this problem is the use of meta-policies which resolve conflicting situations during run-time.

REFERENCES

[1] S. Berman, L. Kesterson-Townes, A. Marshall, and R. Srivathsa, "The power of cloud. driving business model innovation," *IBM Institute for Business Value*, 2012.

[2] M. Dekker, "Critical cloud computing-a ciip perspective on cloud computing services," *white paper, December*, 2012.

[3] C. Goh, *A Generic Approach to Policy Description in System Management*. Hewlett Packard Laboratories, 1997.

[4] R. Craven, J. Lobo, E. Lupu, A. Russo, and M. Sloman, "Policy refinement: Decomposition and operationalization for dynamic domains," in *Network and Service Management (CNSM), 2011 7th International Conference on*. IEEE, 2011, pp. 1–9.

[5] P. Smith, A. Schaeffer-Filho, A. Ali, M. Schöller, N. Kheir, A. Mauthe, and D. Hutchison, "Strategies for network resilience: capitalising on policies," in *Mechanisms for Autonomous Management of Networks and Services*. Springer, 2010, pp. 118–122.

[6] A. Schaeffer-Filho, P. Smith, and A. Mauthe, "Policy-driven network simulation: a resilience case study," in *Proceedings of the 2011 ACM Symposium on Applied Computing*. ACM, 2011, pp. 492–497.

[7] Y. Yu, M. Fry, A. Schaeffer-Filho, P. Smith, and D. Hutchison, "An adaptive approach to network resilience: Evolving challenge detection and mitigation," in *Design of Reliable Communication Networks (DRCN), 2011 8th International Workshop on the*. IEEE, 2011, pp. 172–179.

[8] S. N. Shirazi, S. Simpson, S. Oechsner, A. Mauthe, and D. Hutchison, "A framework for resilience management in the cloud," *e & i Elektrotechnik und Informationstechnik*, vol. 132, no. 2, pp. 122–132, 2015.

[9] S. N. Shirazi, S. Simpson, A. K. Marnerides, M. Watson, A. Mauthe, and D. Hutchison, "Assessing the impact of intra-cloud live migration on anomaly detection," in *Cloud Networking (CloudNet), 2014 IEEE 3rd International Conference on*. IEEE, 2014, pp. 52–57.

[10] L. Mei, W. Chan, and T. H. Tse, "A tale of clouds: Paradigm comparisons and some thoughts on research issues," in *Asia-Pacific Services Computing Conference, 2008. APSCC '08. IEEE*, Dec 2008, pp. 464–469.

[11] P. Goyal and R. Mikkilineni, "Policy-based event-driven services-oriented architecture for cloud services operation management," in *Cloud Computing, 2009. CLOUD '09. IEEE International Conference on*, Sept 2009, pp. 135–138.

[12] K. W. Hamlen, L. Kagal, and M. Kantarcioglu, "Policy enforcement framework for cloud data management." *IEEE Data Eng. Bull.*, vol. 35, no. 4, pp. 39–45, 2012.

[13] B. Meyer, F. Anstötz, and C. Popien, "Towards implementing policy-based systems management," *Distributed Systems Engineering*, vol. 3, no. 2, p. 78, 1996.

[14] H.-G. Hegering, S. Abeck, and R. Wies, "A corporate operation framework for network service management," *Communications Magazine, IEEE*, vol. 34, no. 1, pp. 62–68, 1996.

[15] J. Roos, P. Putter, and C. Bekker, "Modelling management policy using enriched managed objects," in *Proceedings of the IFIP TC6/WG6. 6 Third International Symposium on Integrated Network Management with participation of the IEEE Communications Society CNOM and with support from the Institute for Educational Services*. North-Holland Publishing Co., 1993, pp. 207–215.

[16] P. Kaikini, L. Lewis, R. Malik, E. Rustici, W. Scott, S. Sycamore, and S. Thebaut, "Method and apparatus for defining and enforcing policies for configuration management in communications networks," Feb. 16 1999, uS Patent 5,872,928.

[17] D. C. Verma, *Policy-based networking: architecture and algorithms*. New Riders Publishing, 2000.

[18] M. Casassa Mont, A. Baldwin, and C. Goh, "Power prototype: Towards integrated policy-based management," in *Network Operations and Management Symposium, 2000. NOMS 2000. 2000 IEEE/IFIP*. IEEE, 2000, pp. 789–802.

[19] M. S. Beigi, S. Calo, and D. Verma, "Policy transformation techniques in policy-based systems management," in *Policies for Distributed Systems and Networks, 2004. POLICY 2004. Proceedings. Fifth IEEE International Workshop on*. IEEE, 2004, pp. 13–22.

[20] R. Darimont and A. Van Lamsweerde, "Formal refinement patterns for goal-driven requirements elaboration," in *ACM SIGSOFT Software Engineering Notes*, vol. 21, no. 6. ACM, 1996, pp. 179–190.

[21] A. K. Bandara, E. C. Lupu, J. Moffett, and A. Russo, "A goal-based approach to policy refinement," in *Policies for Distributed Systems and Networks, 2004. POLICY 2004. Proceedings. Fifth IEEE International Workshop on*. IEEE, 2004, pp. 229–239.

[22] J. Rubio-Loyola, J. Serrat, M. Charalambides, P. Flegkas, G. Pavlou, and A. L. Lafuente, "Using linear temporal model checking for goal-oriented policy refinement frameworks," in *Policies for Distributed Systems and Networks, 2005. Sixth IEEE International Workshop on*. IEEE, 2005, pp. 181–190.

[23] P. Smith, D. Hutchison, J. Sterbenz, M. Schöller, A. Fessi, M. Karaliopoulos, C. Lac, and B. Plattner, "Network resilience: a systematic approach," *Communications Magazine, IEEE*, vol. 49, no. 7, pp. 88–97, July 2011.

[24] J. P. Sterbenz, D. Hutchison, E. K. Çetinkaya, A. Jabbar, J. P. Rohrer, M. Schöller, and P. Smith, "Resilience and survivability in communication networks: Strategies, principles, and survey of disciplines," *Computer Networks*, vol. 54, no. 8, pp. 1245–1265, 2010.

[25] M. Scholler, R. Bless, F. Pallas, J. Horneber, and P. Smith, "An architectural model for deploying critical infrastructure services in the cloud," in *Cloud Computing Technology and Science (CloudCom), 2013 IEEE 5th International Conference on*, vol. 1, Dec 2013, pp. 458–466.

[26] R. Bless, M. Flittner, J. Horneber, D. Hutchison, C. Jung, F. Pallas, M. Schöller, S. N. ul Hassan Shirazi, S. Simpson, and P. Smith, "Whitepaper "af 1.0" seccrit architectural framework," 2014.

[27] A. K. Bandara, E. C. Lupu, and A. Russo, "Using event calculus to formalise policy specification and analysis," in *Policies for Distributed Systems and Networks, 2003. Proceedings. POLICY 2003. IEEE 4th International Workshop on*. IEEE, 2003, pp. 26–39.

[28] J. D. Moffett and M. S. Sloman, "Policy hierarchies for distributed systems management," *Selected Areas in Communications, IEEE Journal on*, vol. 11, no. 9, pp. 1404–1414, 1993.

[29] R. Boutaba and I. Aib, "Policy-based management: A historical perspective," *Journal of Network and Systems Management*, vol. 15, no. 4, pp. 447–480, 2007.

[30] J. Rubio-Loyola, J. Serrat, M. Charalambides, P. Flegkas, and G. Pavlou, "A methodological approach toward the refinement problem in policy-based management systems," *Communications Magazine, IEEE*, vol. 44, no. 10, pp. 60–68, 2006.