To appear in the Journal of Computer and System Sciences
(Special Issue on Dependable and Secure Computing). Elsevier

Self-Adaptive Federated Authorization Infrastructures

Christopher Bailey, David W. Chadwick, Rogério de Lemos
School of Computing
University of Kent
Canterbury, UK
{c.bailey, d.w.chadwick, r.delemos}@kent.ac.uk

Abstract— Authorization infrastructures are an integral part of any network where resources need to be protected. They act as the gateway for providing (or denying) subjects (users) access to resources. As networks expand and organizations start to federate access to their resources, authorization infrastructures become increasingly difficult to manage. In this paper, we explore the automatic adaptation of authorization assets (policies and subject access rights) in order to manage federated authorization infrastructures. We demonstrate adaptation through a Self-Adaptive Authorization Framework (SAAF) controller that is capable of managing policy based federated role/attribute access control authorization infrastructures. The SAAF controller implements a feedback loop to monitor the authorization infrastructure in terms of its assets and subjects' behavior, analyze potential adaptations for handling abnormal behavior, and act upon assets of an authorization infrastructure for changing its configuration. A prototype of the SAAF controller is evaluated in a federated authorization infrastructure (federation) built with SimpleSAMLphp, in which a PERMIS standalone authorization server protects a service provider's resources, and identity providers utilize LDAP directories to store subject authentication and authorization attributes.

Keywords— self-adaptation, authorization, policy management, identity management, autonomic security, RBAC, ABAC, SAML, PERMIS

1. *Introduction*

A great deal of research and time is put into securing access to, and ensuring legitimate use of protected resources. There exist a variety of different approaches such as, role based access control (RBAC) [1] and attribute based access control (ABAC) [2], as well as more sophisticated systems involving detection [3], trust and reputation [4], and usage control [5] to compliment authorization. However, once authorization has been setup (i.e., defining authorization policies) there exist few automated mechanisms that both identify when such access is being used incorrectly, and mitigate or prevent further misuse automatically. Traditionally organizations rely on audit trails and human administrators to monitor these systems to identify abnormal behavior [6]. The detection of abnormal behavior, attributed to the misuse of system resources by authorized subjects, is often not at the forefront of concern for organizations. However, it is known that an internally authorized user can cause far greater damage in comparison to an external attacker simply due to their access rights [7]. For example, during July 2010 it is alleged that a US army intelligence analyst downloaded 0.25 million classified US military documents from a US Department of Defense website [8]. Assuming the analyst was an authorized user and that access was requested and granted on a document-by-document basis, we can say that the analyst had appropriate access rights and utilized the authorization system correctly. Any automated monitoring of the authorization system would not have picked up abnormal behavior as the authorization service processed the analyst's access requests according to its access control policies. However, to a human administrator numerous similar requests in a short period of time would have flagged up inappropriate behavior, requiring immediate changes to the authorization infrastructure to mitigate any further damage.

Federated authorization builds upon existing authorization models, including RBAC and ABAC. It provides the method through which large scale distributed access can be granted. For example, within a federated authorization infrastructure that utilizes the ABAC authorization model, a subject is assigned attributes by one organization (e.g., an identity provider), and each attribute is assigned permissions by another organization (the service provider). In federated access, RBAC/ABAC is extended to state which organizations (i.e., identity providers) are trusted to assign which attributes to which subjects. This requires a subject to have a relationship with one or more of these trusted organizations.

Assuming that all subjects act appropriately within their access rights is an increasingly risky assumption to make, especially relevant as organizations work together and federate their access control systems. As the number of subjects with federated access grows, as does the risk of insider threat. This is made even more challenging to manage in federated access, as resource holders are unaware of who is actually being granted access, and thus how to identify and respond to insider threat. This may in part explain why federated access is not widely deployed today.

This paper claims that federated authorization infrastructures must be capable of identifying abnormal behavior, and autonomously change authorization assets (authorization constraints / subject privileges) in order to prevent and mitigate misuse of access (insider threat). In a previous paper [9], we have introduced the conceptual design of a Self-Adaptive Authorization Framework (SAAF) for monitoring, analyzing, planning and executing required adaptations for managing a federated authorization infrastructure, depending on the level of misuse of access rights. The novel aspect of this paper builds on our previous work through the detailed design and implementation of a prototype SAAF controller, highlighting the controller's key phases and use of models for managing authorization infrastructures. We detail the design and generation of these models as part of the SAAF controller's feedback loop, along with adaptation scenarios in which the SAAF controller is deployed. Other contributions of this paper are the effective integration of the SAAF controller with a federated authorization infrastructure; comprising a PERMIS standalone authorization server and SAML based service and identity providers, in order to demonstrate the overall feasibility of SAAF.

The rest of this paper is structured as follows. In Section 2, we describe the domain model of federated authorization infrastructures, and the conceptual design of SAAF. Section 3 outlines the SAAF controller in terms of its key components and the models used. Section 4 describes the SAAF controller prototype, which was deployed in a self-adaptive federated authorization infrastructure. In Section 5, we present an abnormal usage scenario and evaluate how the SAAF controller manages and compare the results to current technology. In Section 6, we discuss current related work in comparison to SAAF. Finally, Section 7 concludes with an evaluation of our work and indicates where future work is still required.

2. Self-Adaptive Authorization Framework

In this paper, we refer to our Self-Adaptive Authorization Framework (SAAF) that is capable of being attached to policy based RBAC/ABAC federated authorization infrastructures. It is designed to integrate with current authorization infrastructures, such as PERMIS [10], Shibboleth [11], and XACML [12] in order to make them adaptable and self-managing, rather than designing an entirely new type of authorization infrastructure. SAAF's objective is to autonomously monitor the usage of an existing authorization infrastructure, make judgments on the behavior of subject interactions (in the form of authorization requests and decisions), and adapt the authorization infrastructure accordingly. SAAF is reactive, in the sense that it monitors the use of a target authorization infrastructure by subjects in order to detect abnormal behavior. Abnormal behavior is defined by a set of rules that capture conditions that exhibit insider threat in the deployment environment of SAAF. Once abnormal behavior is detected a decision is made on whether to adapt the authorization infrastructure or not.

The following section describes in detail the expected target domain that SAAF can manage, along with SAAF's conceptual design.

2.1. Target Domain: Federated Authorization Infrastructures

SAAF's target domain identifies what services can exist, which services are configurable, what can be monitored, and how access is requested and granted, within a federated authorization infrastructure. As SAAF is designed for the management of RBAC/ABAC authorization models, the domain model is specific to these authorization models.

Role based access control (RBAC) and its more generic variant attribute based access control (ABAC) are models of authorization, facilitating access to protected resources through the use of roles/attributes and by assigning permissions to those roles/attributes. A set of rules exist which state that in order for a subject to access a resource (e.g. 'Read Database'), the subject must have a specific set of roles or attributes (e.g. Role of Staff). The RBAC/ABAC authorization model can be extended to include: hierarchy of roles/attributes, static separation of duties, dynamic separation of duties and arbitrary conditions. Our work is focused initially on core RBAC/ABAC over a distributed federated implementation.

A federated RBAC/ABAC authorization infrastructure, as implemented in [10] [11] [12], comprises the following components:

- a set of distributed role/attribute issuing authorities (AAs), also known as Identity Providers (IdPs), which assign digitally signed credentials to subjects in a session,
- a Credential Validation Service (CVS) at the Service Provider's (SP) site, which validates the roles/attributes issued to the subject as credentials [13], and
- a Policy Decision Point (PDP) also at the SP's site, which evaluates if these roles/attributes give the user sufficient permission to access the requested resource.

Through the use of policies, attributes and credentials, subject authorization is provided. We refer to these as 'assets' of a federated RBAC/ABAC authorization infrastructure. These assets demonstrate the parts of an authorization infrastructure that are changeable and therefore can be modified through self-adaptation to impact future authorization decisions.

Figure 1 shows a simplified view of SAAF's target domain model. It shows the three types of service required for effective federated authorization, along with the assets used to define the control and input/output of such services. Services can be categorized as Service Provider (SP) services (Credential Validation Service and Policy Decision Point), and Identity Provider (IdP) services.

The target domain model has six assets that are manageable. These are the attributes, credentials, and valid attributes assigned to the subjects; and the Credential Issuing Policy, Credential Validation Policy and Access Control Policy, collectively referred to as Authorization Policies (AZPs). A 7th (unmanageable) asset is a log of the access requests and access decisions. This can be observed in order to generate usage statistics and a history of access requests.

Through changing the subject's attributes or the credential issuing policy we control what credentials may be issued, thus potentially increasing or reducing the subject's permissions. The revocation of credentials allows for the termination of access sessions midway. Through the adaptation/switching of any of the authorization policies, SAAF is able to impact a group of subjects by controlling authorization at a higher level. The authorization infrastructure interprets these assets in order to provide access control decisions. The modification of these assets by SAAF impacts the access control decision thus preventing abnormal behavior.

In order for an implementation of the target domain model to be manageable by SAAF, we make the following assumptions:

- the authorization infrastructure is capable of generating logs of its actions, e.g., failed and successful access requests, and that these logs are available to be read by SAAF;
- the authorization infrastructure has interfaces that allow it to receive new policies or replace old ones currently in use, and that SAAF can access these interfaces;
- identity providers are capable of allowing SAAF to modify the subject attribute assignments, but if this is not possible then
- identity providers are capable of accepting notifications (from SAAF) about their subject attribute mis-assignments and cases of abuse, and are willing to remove and add new attributes to their subjects and notify SAAF when the requested changes have been effected.

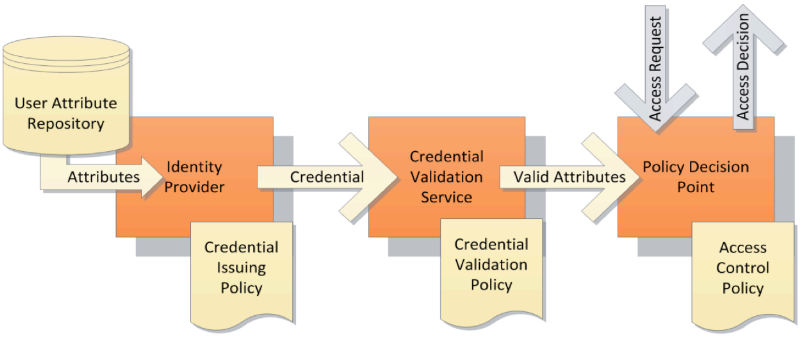


Figure 1. Domain Model for Federated RBAC/ABAC Infrastructures

2.2. *SAAF Conceptual Design*

SAAF is based on the Monitor-Analyze-Plan-Execute-Knowledge (MAPE-K) reference model [14] adapting assets associated with a federated authorization infrastructure. The conceptual design (Figure 2) identifies two major components, the SAAF controller, which utilizes a feedback loop [15], and the authorization infrastructure (the target, which conforms to SAAF's target domain model).

The *Monitor* is a simple component that retrieves assets of the federated authorization infrastructure via system *Probes*, and updates the authorization infrastructure model and behavior model. For example it captures an access request and corresponding authorization decision, and updates the behavior statistics within the behavior model (representing knowledge of usage within the target system), as well as subject-attribute relationships within the authorization infrastructure model. The monitor uses triggers to identify exactly what statistics are required for the behavior model (and trigger the need for adaptation).

The *Analyzer's* objective is to assess the behavior model, in order to identify if abnormal behavior has taken place and identify possible solutions that may prevent the abnormal behavior from continuing. As an exception to the MAPE-K reference model, we introduce the need to analyze possible solutions within the analyzer, as solution analysis is highly correlated to the analysis of abnormal behavior. The set of identified solutions is then sent to the planner. These solutions encapsulate actions that modify rules belonging to authorization policies, and individual subjects' attributes in order to resolve abnormal behavior.

The role of the *Planner is to* select the most relevant solution to solve the identified abnormal behavior. The selected solution is transformed into an executable plan that is sent to the executor, such as, generate a new access control policy then request the authorization service to activate the new policy.

The *Executor* adapts the authorization infrastructure in accordance with the plan, via *Effectors*. These effectors enable the adaptation of authorization services and their assets, and provide an interface to SAAF to execute commands on the authorization infrastructure.

The target federated authorization infrastructure is an implementation of the SAAF target domain model (Figure 1). It conforms to the domain model in such a way that deployed services and assets are an instantiation of the types of services and assets present in the domain model. There can be multiple instances of authorization services and assets. For example, there may be several Identity Providers (IdPs) within one federated authorization infrastructure. An implementation of the target domain model does not have to conform to the domain model completely, whereby only a subset of authorization services need exist (e.g., no configurable Credential Issuing Policy). In these cases SAAF is still able to manage authorization, yet management decisions are restricted to the scope of what can be controlled and monitored.

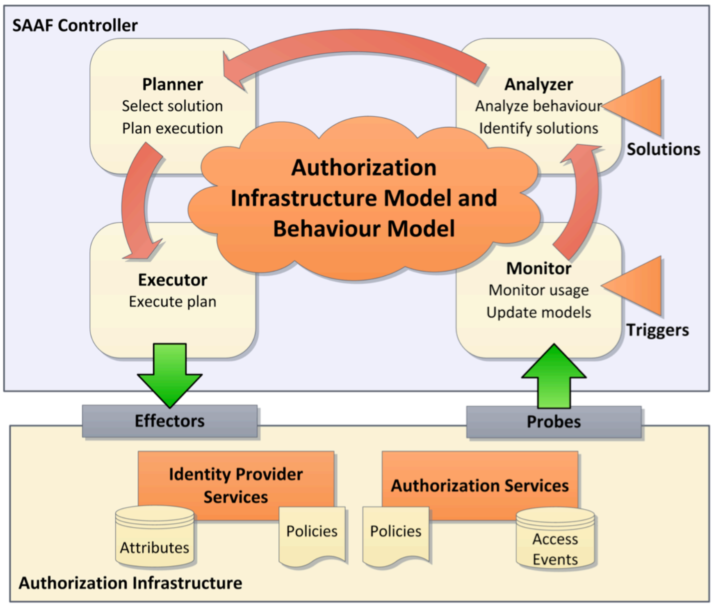


Figure 2. SAAF Conceptual Design

3. *SAAF Controller*

We demonstrate SAAF as an autonomic controller (Figure 3) that conforms to the conceptual design of SAAF. It is designed to operate in a continuous cycle, whereby the monitor (Behavior Gauges and Asset Monitor) constantly update a model that represents the rules and access rights assignments within the federated authorization infrastructure (authorization infrastructure model), as well as a data model that captures statistics about the use of the authorization infrastructure (Behavior Model). Whilst the models are being updated, the Analyzer searches for abnormal behavior. Once abnormal behavior is identified, it attempts to solve this by producing tailored solutions. The tailored solutions are passed to the Planner, which together with the Executor realize the solution. In this section we discuss the operation of each component of the controller in detail and how the SAAF controller manages a federated authorization infrastructure.

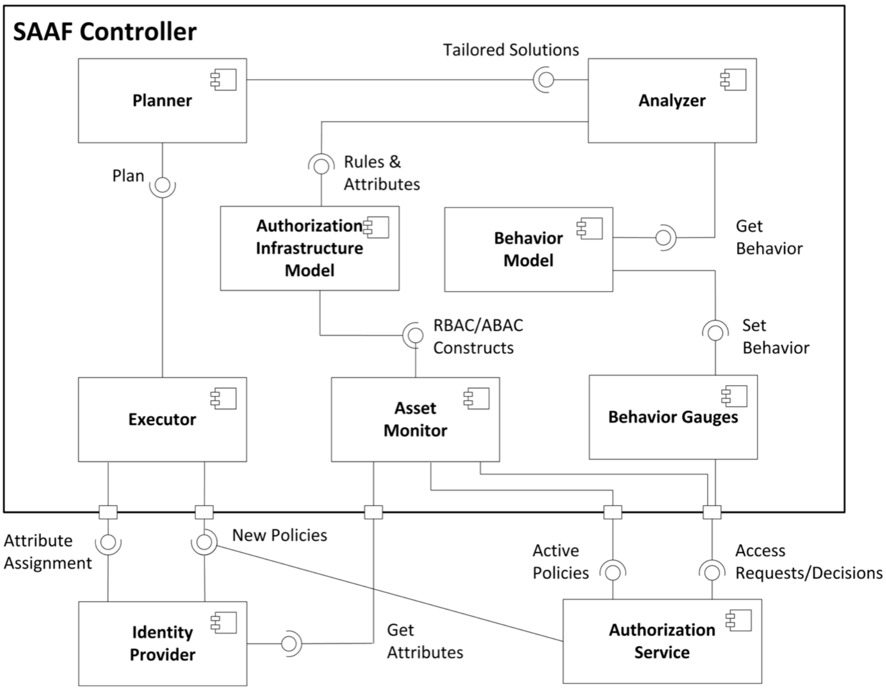


Figure 3. SAAF Controller Architecture and Interfaces

3.1. SAAF Controller Models

The SAAF controller relies on models to facilitate the operation of the Analyzer, Planner, and Executor components. These models breakdown into the authorization infrastructure model, which stores information about authorization assets (Figure 1) active in the target system, and the behavior model, which provides statistics regarding the use of authorization assets.

The *Authorization Infrastructure Model* (Figure 4) provides constructs of RBAC/ABAC rules, such as attribute-permission assignments {Attribute, {Target, Action}}, identity provider-attribute assignments {Identity Provider, Attribute}, and subject-attribute assignments {{Subject, Attribute}, Identity Provider}. Previous research has already provided the basis for a universal construct for RBAC/ABAC system policies [16]. These constructs represent a generic view of the managed federated authorization infrastructure's active authorization policies and subject-attribute assignments. This allows SAAF to assess and validate adaptations, as well as identify what specific constructs must be adapted in the light of abnormal behavior. A key feature of the authorization infrastructure model is that it allows SAAF to adapt at the model level, whereby the authorization infrastructure model is modified and then realized through the generation of new implementation specific authorization policies. These policies can then be activated within the target authorization services to impact future authorization decisions and prevent further abnormal behavior.

The *Behavior Model* is a data model of usage statistics about existing relationships within the authorization infrastructure model that may be adapted. It is populated by the assessment of access requests and decisions made within the target authorization infrastructure. The statistics captured within the behavior model are directly associated to the relationships within the authorization infrastructure model, such as {Attribute, Target, Action} or {Subject, Identity Provider, Attribute, Target, Action}. These relationships contain statistical properties such as, the rate of access requests that a subject from an IdP has accessed {Target, Action}. Statistical properties allow for statements of usage to be drawn about subjects, roles/attributes, and permissions for a certain period of time. For example, the average frequencies of requests by attribute A, or subject S to read target T per minute during the last 30 days. This enables SAAF to identify how subjects are using the system collectively. Note that in the current implementation we only indirectly capture the results of the credential validation process through monitoring the access request. Consequently we do not record all the attributes that the IdP has assigned to the subject (as credentials), only valid attributes.

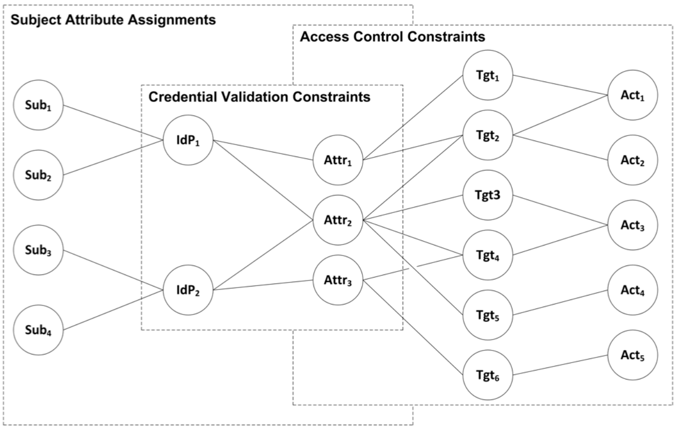


Figure 4. Authorization Infrastructure Model

3.2. Monitor

The SAAF monitor is a combination of behavior gauges and an asset monitor, responsible for updating the SAAF controller models. The monitor relies on system probes that exist within the target system (a federated authorization infrastructure). Authorization service 'policy' probes detect when a policy has been changed or a new policy activated within a particular service. Authorization service 'access' probes detect when new access requests have been made, typically when the policy decision point service has updated its access logs. The monitor uses a set of pre-defined triggers for activating adaptation when abnormal behavior is identified.

3.2.1. Triggers

Each *trigger* describes a relationship within the authorization infrastructure model {Subject, Identity Provider, Attribute, Target, Action}, along with a set of conditions, such as rate of access requests that conform to this relationship, over a given time interval. There are two types of triggers, base triggers and composite triggers. Base triggers use data within the behavior model to trigger the need for adaptation. For example, a base trigger activates the need for adaptation when a subject accesses a resource more than 5 times per minute interval. Composite triggers are composed of base triggers, to detect when multiple trigger conditions have been met over a set amount of occurrences and time. For example, a composite trigger activates the need for adaptation when multiple subjects, from multiple identity providers, all access a resource more than 5 times per minute interval in a 30-day period.

3.2.2. Updating the Authorization Infrastructure Model

Activated policies that have been detected by relevant system probes undergo a process of model transformation [17] within the asset monitor. Each policy document represents a model of rules for the desired authorization service. In order for the SAAF controller to understand a policy, the policy must undergo model transformation. Model transformation allows the conversion of implementation specific formats (i.e., XACML or PERMIS proprietary schema's [10]), to the generic RBAC/ABAC view that the SAAF controller can interrogate and use, as described in Section 3.1. The authorization infrastructure model may hold multiple modeled active policies, requiring that each modeled policy is labeled with meta-information to provide ownership and location data that is required when generating new policies as part of adaptations. Whenever an active policy is changed, the modeled policy, within the authorization infrastructure model, is remodeled.

Subject-attribute assignments are also modeled within the authorization infrastructure model. This allows the SAAF controller to be aware of what subjects have which valid attributes, when forming adaptations. Unlike authorization policies, the SAAF controller does not have a complete view of all subject-attribute assignments, due to the nature of how these assignments are stored (in multiple identity providers) and the fact that the credential validation logs are not interrogated. Therefore only valid attribute assignments are captured through monitoring access control requests (which provide a subject's identifying ID, valid attributes, the target resource being accessed and the action requested) and access decisions. In addition, identity providers can push attribute changes to the SAAF controller; however this is only as confirmation of successful adaptations against subject-attribute assignments.

3.2.3. Updating the Behavior Model

The behavior model is updated through the processing of logged access requests/decisions, and generation of statistics by a set of behavior gauges. Behavior gauges present a means of collecting specific statistics about relationships within the authorization infrastructure model. For each access request and decision that is logged, the relevant behavior gauges update statistics about existing relationships within the behavior model, which in turn drive adaptation (Figure 5). Gauges are based entirely on triggers, and for each trigger there exists a set of gauges, depending on how many observed relationships within the authorization infrastructure model match the trigger. For example, if there are 20 subjects that meet the set of relationships defined by the trigger <any subject, from any identity provider, accessing action 'print' on target 'printer' with attribute 'role=staff'>, then there are at least 20 gauges required for that trigger. If a gauge does not exist, yet the relationship observed in the access request matches a trigger, then a new gauge is created. If gauges already exist for the observed relationship, then the gauges are updated.

In accordance with the two types of trigger, there are two types of gauge: a gauge that generates statistics for base triggers, and a gauge that generates statistics for composite triggers. Both types of gauge operate as described previously, with the only exception that composite gauges are updated as gauges belonging to base triggers become full (indicating multiple base trigger conditions have been met). Once either type of gauge becomes full, based on trigger conditions, abnormal behavior has been identified and is sent as a snapshot of behavior to the analyzer.

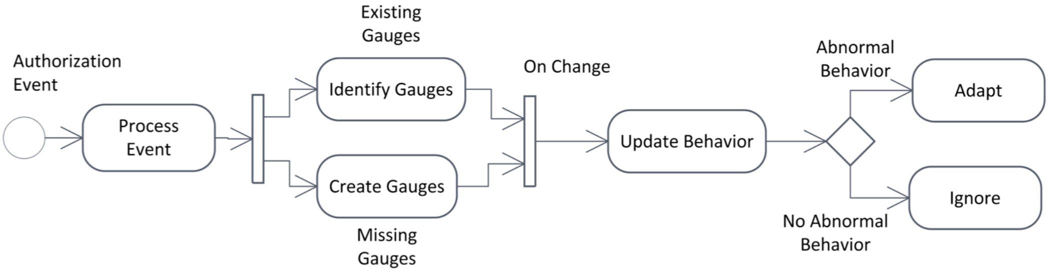


Figure 5. Gauge generation

3.3. Analyzer

The analyzer component fulfills two purposes, related to 1) analyzing misbehavior in order to identify the need for adaptation, and 2) analyze what solutions are applicable to solving the misbehavior.

3.3.1. Behavior Analysis

The analyzer identifies the problem that caused behavior gauges to meet trigger conditions, through *behavior analysis*. For example, if a gauge measures the number of times a file has been retrieved per minute and it triggers when 10 retrievals per minute have been recorded, the analyzer needs to determine if this is the same subject retrieving the file 10 times, or 10 different subjects doing the same thing once, since the solution to either scenario may be different. The analyzer must therefore first determine the exact nature of the abnormal behavior that took place before finding the relevant solution(s) and tailoring it (them) to the specific conditions of the abnormal behavior, in order to prevent the abnormal behavior from continuing.

3.3.2. Solution Analysis

For any given abnormal behavior there can be a set of relevant parameterized solutions, which prevent that behavior from continuing. Solutions may exist in the form of alternative rules that are capable of preventing the identified abnormal behavior (e.g., remove subject attribute or remove attribute permission). Solutions are further defined by a set of actions (such as 'remove ABAC constraint', 'activate policy') and are reusable by other solutions. There are a finite number of actions possible for a solution. These are all actions applicable to the controllable assets described in SAAF's target domain model.

Solution analysis interprets relevant solutions from the set of pre-defined solutions for the identified abnormal behavior (trigger). The analysis to be performed relies on the variables defined by the identified behavior. For example, if a permission was misused by a subject (or attribute), the analysis would be focused around that subject, that attribute and that permission. As solutions are parameterized, the actions stated for each applicable solution must be tailored to the identified behavior, using modeled constructs within the authorization infrastructure model. An instance of each relevant solution and action is created, and tailored to match. For example, a solution instance is tailored to the subject's behavior that caused the need for adaptation, including the attribute used, and the identity provider that assigned the attribute. Each applicable solution is tailored in this manner, and sent as a set to the planner component.

3.4. Planner

The planner component fulfills two purposes, the selection of an ideal solution, in terms of a weighted calculation of impact, from the set of solutions provided by the analyzer, and the generation of an executable plan.

3.4.1. Solution Selection

Selecting an ideal solution is critical to ensuring only necessary adaptations take place, as solutions vary in terms of severity and risk, from the perspective of the context the authorization infrastructure is deployed in. Whilst each trigger specifies the condition of a particular abnormal state, the existing triggers lack the definition of any impact or loss of utility that the abnormal behavior might have. We recognize the need for a multi-attribute function to produce a utility [18] or impact value for each solution, yet for implementation purposes this utility is simply calculated on a single dimension, weight of impact. The weight of impact represents the number of subjects who have caused the abnormal behavior having their access rights removed, against the number of subjects who have not caused the abnormal behavior having their access rights removed as a consequence of the solution. The former must outweigh the latter for adaptation to take place. In other words, adaptation will only take place if more offending subjects can be impacted than non-offending subjects.

For the set of solutions received by the planner, each solution is ordered by this weight of impact. For example, if a subject misuses their access rights to a printer, a solution may be to modify the policy where the attribute required to access the printer is no longer valid. However, the weight associated with this would mean that many subjects are impacted rather than just the offending subject. A more relevant solution might be to request the identity provider (IdP) to remove the subject’s attribute.

3.4.2. Plan Generation

Plan generation identifies what actions need to be performed for realizing the chosen solution. It can be viewed as an automatically generated set of step-by-step instructions with specific details on how to execute an adaptation [19]. In the current implementation the actions are pre-defined in the solutions policy. For each action attached to the chosen solution, the planner includes meta-information to enable execution. If it is an action against the authorization infrastructure model, the planner identifies the specific relationship that must change, including ownership information regarding the actual authorization services and identity providers. Actions are ordered in stages of execution, whereby changes to the authorization infrastructure model are made first, followed by generation of new assets (such as policies), then instructions to system effectors (activate policy, remove attribute assignment).

3.5. Executor

The executor component is a simple component that executes a plan generated by the planner. Each action the executor attempts to execute is idempotent, therefore it will continue to attempt to make a request to an external effector until the action is successful or a time out limit is met. The outcome of a plan execution either results in a successful or failed plan. Successful plans are characterized by the positive response from all the external system effectors. Unsuccessful plans are characterized by either an error message or a failure to respond by at least one of the external system effectors.

Upon completion of a plan, successful or not, system probes update the SAAF controller’s monitor components with a new view of the authorization infrastructure, maintaining a consistent view of what is modeled, to what is actually present in terms of rules and subject-attribute assignments.

4. SAAF Deployment

This section describes the deployment of a self-adaptive federated authorization infrastructure, which is shown in Figure 6. The self-adaptive federated authorization infrastructure comprises an authorization server made up of the SAAF controller and PERMIS standalone authorization service [10], a service provider that provides resources for federated subjects to request access to, and two identity providers that maintain subject-attribute assignments for federated subjects belonging to two different organizations. The self-adaptive federated authorization infrastructure is deployed across several virtual machines within a local area network. Basic system effectors and probes have been implemented to facilitate SAAF's controller interfaces, as shown in Figure 3 and Figure 6, however they will not be discussed in this paper.

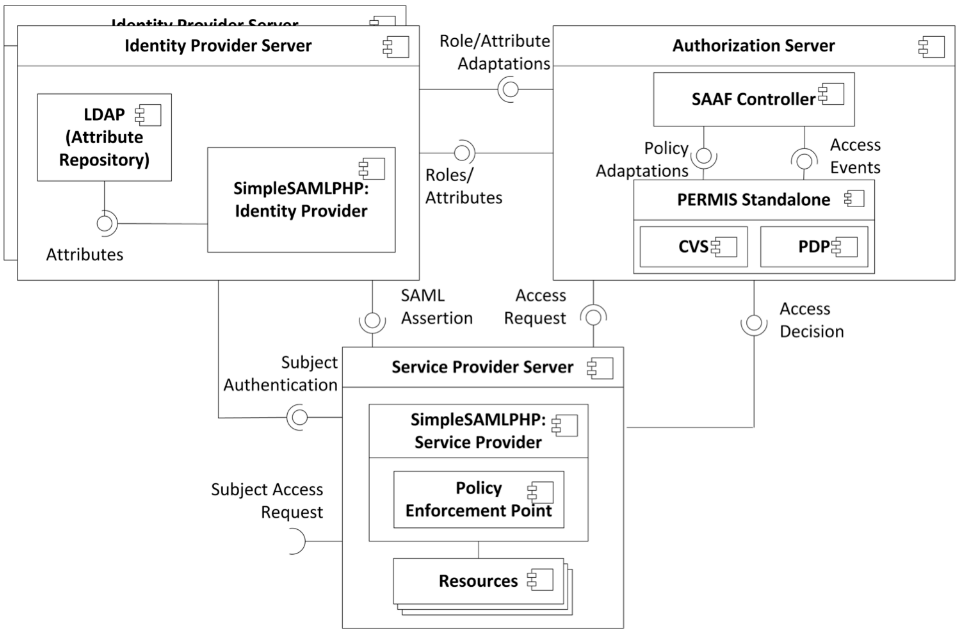


Figure 6. Architecture of the Self-Adaptive Federated Authorization Infrastructure

4.1. Authorization Server

The authorization server runs on a VMware virtual machine, with Ubuntu v10.10, 1GB memory. The virtual machine is hosted on a MacBook pro (OS X Lion) 2.4GHz, 4GB memory. The authorization server contains a deployment of PERMIS as a standalone server, and an implementation of the SAAF controller described in Section 3. The PERMIS standalone encapsulates a credential validation service (CVS) and policy decision point (PDP) for generating access decisions based on a single authorization policy. The authorization policy provides rules on credential validation as well as RBAC constraints, which can be broken up into the credential validation policy and access control policy described in Section 2.1. The PERMIS standalone receives Simple Object Access Protocol (SOAP) messages that define a subject's access request, over an SSL connection. The contents of the SOAP message are assessed for valid attributes and whether or not the valid attributes fulfill the conditions of the requested access. The sender of the SOAP message, a resource's policy enforcement point (PEP), then receives a SOAP response containing an access decision, either: grant, deny, or not applicable. The PEP can use this value to allow the subject requestor access to the resource. All access requests and decisions are logged, recording the actions made by the CVS and PDP.

The deployment of the SAAF controller was implemented in Java, and installed on the same server as the PERMIS standalone authorization server. Models within the SAAF controller are populated as a collection of Java objects, and relational records are stored in a locally accessible MySQL database. In particular, policies modeled within the authorization infrastructure model, stored as Java objects, are populated through model transformation with a tool generated using the Eclipse EMF modeling framework [20]. Behavior statistics are populated and stored as Java objects within the SAAF controller's run-time memory. Subject-attribute assignments are populated and stored as a set of relational records within the MySQL database. The controller's triggers and solutions are written in XML, stored on the server's file system as XML documents. These XML documents represent a 'behavior policy' and 'solutions policy' and are parsed when the SAAF controller is first initialized. Communications between the SAAF controller and its managed services are carried out in a variety of ways: through the host operating system in which the SAAF controller is deployed when managing PERMIS standalone services, and via SOAP messages (over SSL) when managing external system effectors and probes (in the case of federated identity providers).

4.2. *Service Provider*

A single service provider (SP) server is run on a VirtualBox virtual machine, with Debian 6.0.5, 512mb memory. The virtual machine is hosted on a Windows 7 machine, 2.40GHz, 3GB memory. The service provider is deployed as a set of web resources developed in PHP, protected by a single policy enforcement point (PEP). The SP represents an organization and the organization's resources we wish to protect from abnormal behavior. The PEP's role is to facilitate a subject accessing the SP's resources; it acts as a guard to actions within a resource. It does not decide access, only enforces access based on decisions made by an external authorization service (in this case, the PERMIS standalone). The PEP is built in PHP, as part of a simpleSAMLphp [21] installation, in order to allow the SP to communicate with simpleSAMLphp identity providers. The PEP receives access requests from subjects and redirects them to their authenticating identity provider. The identity provider returns SAML [22] assertions that represent the requesting subject's credential assets (described in Section 2.1). The PEP encapsulates this assertion into a PERMIS access request SOAP message, and enforces the corresponding access decision that is returned as a SOAP response from the PERMIS standalone. Based on the decision made by the PERMIS standalone, the PEP provides the requesting subject with access to the requested resource/action.

4.3. *Identity Providers*

Two identity provider servers have been deployed. Each is run on a separate VirtualBox virtual machine, with Debian 6.0.5, 512mb memory. The virtual machine is hosted on a Windows 7 machine, 2.40GHz, 3GB memory. The identity providers (IdPs) are deployed as an authenticating SAML service, implemented using SimpleSAMLphp. The IdP authenticates a subject and then indicates to the SP that the subject is who they say they are. This is achieved by sending a SAML assertion containing the subject's releasable attributes to the SP (i.e., what the IdP is willing to share based on it's credential issuing policy). The SAML assertion also contains a unique persistent ID in which the service provider can identify the subject with. A Lightweight Directory Access Protocol (LDAP) [23] with a Berkley database backend [24] is used to store an identity provider's subject attributes, hosted on the identity provider's own server. The identity provider is configured to release all authorization-based attributes, such as the 'permisRole' attribute that is used by PERMIS authorization policy to denote a subject's role, as well as a unique identifier for the subject. Identity providers are managed by SAAF via a SimpleSAMLphp effector [25].

5. *Experiments*

This section describes the qualitative evaluation of the deployed self-adaptive authorization infrastructure, described in Section 4. We simulate a case of abnormal behavior, caused by a group of malicious subjects from one identity provider, against the resources of the service provider. We discuss how the SAAF controller identifies and responds to this case of abnormal behavior, conveying our results with snapshots of before and after states of an active authorization policy, and subject-attribute assignments stored in an LDAP directory. We follow up our case study with an evaluation of results, a comparison of the SAAF prototype to usage control techniques built into an authorization service, ending with a discussion of current limitations of the self-adaptive authorization infrastructure.

5.1. *Case Study*

A *business* organization shares access to its online resources with its own employees, and with employees belonging to a separate *contractor* organization. The business relies on the deployed self-adaptive federated authorization infrastructure described in Section 4, and manages its own identity provider (IdP) server, along with its service provider (SP) server. The contractor organization also manages its own IdP server, along with their employees' roles and attributes. The business shares access to its resources with the contractor IdP in order to allow the contractor organization to perform payroll operations on a web based payroll system. The contractor has an automated system that runs the payroll once per month, and there are occasional manual payroll operations during the month to deal with exceptional circumstances. The SAAF controller is used to control the number of exceptional circumstances that are deemed to be normal.

The business's resources are protected by a single PERMIS authorization policy (AZP), which is activated in the PERMIS standalone authorization server. This AZP defines credential validation rules and RBAC rules, written in PERMIS's own proprietary policy schema, and stored as a digitally signed X.509 policy certificate on the authorization server's file system.

```
<RoleAssignment ID=”ContractorIdPAssignment”>
        <SubjectDomain ID=”Contractor”/>
        <RoleList>
                <Role Type=”permisRole” Value=”Contractor”/>
        </RoleList>
        <Delegate Depth=”0”/>
        <SOA ID=”ContractIdP”/>
        <Validity/>
</RoleAssignment>
<TargetAccess ID=”ContractPayrol”>
        <RoleList>
                <Role type=”permisRole” Value=”Contractor”/>
        </RoleList>
        <TargetList>
                <TargetDomain ID=”PayrollSystem”/>
                <AllowedAction ID=”getEmpPayslip”/>
                <AllowedAction ID=”runPayroll”/>
        </TargetList>
</TargetAccess>
```

Figure 7. Excerpt from the business's PERMIS Authorization Policy

The AZP[1] in Figure 7 allows members of the attribute permisRole=Contractor to access the business payroll system, in order to perform manual operations on the payroll system and retrieve employee pay slips (Figure 7, TargetAccess). The AZP also defines a role assignment rule (credential validation rule) whereby the Contractor IdP is trusted to assign the permisRole=Contractor attribute to its subjects (Figure 7, RoleAssignment).

We simulate the case where the contractor IdP has been hijacked by a malicious entity (a hacker). Once in control, the hacker has the ability to issue the permisRole=Contractor attribute to a set of rogue subjects so that they can access the business's resources with legitimate access rights. Rogue subjects abuse this access right to mine private information within the business's resources, with a focus on retrieving employee pay slip information. As the Contractor IdP is trusted to assign the Contractor attribute, the rogue subjects are able to mine information from the permissible resources. There are 10 subjects assigned to the Contractor attribute, any number of which may be rogue subjects. These subjects are identified by their persistent ID (PID), which is provided by the subject's IdP when attributes are released in SAML assertions.

5.2. Anomaly detection and adaptation

The SAAF controller is initialized with a behavior policy and solution policy containing, respectively, the triggers that identify abnormal behavior states, and the solutions that the business trusts the SAAF controller to execute for each trigger. These two policies are defined by the service provider, and relate to the AZP deployed in the PERMIS standalone.

The behavior policy (Figure 8) defines the conditions that activate the need for adaptation. It is comprised of a single base trigger, and a single composite trigger. For this specific case study we

[1] Figure 7 only shows the subset of the rules from the deployed AZP that are relevant to the abnormal behavior described in this case study.

have defined the limits of normal behavior as base trigger bt1, and composite trigger ct1. The base trigger (bt1) specifies a state of abnormal behavior when any subject, from any provider, uses the attribute permisRole=Contractor to access the *get employee pay slip action* on the payroll system, greater than 5 times per minute. Should these conditions be met, adaptation may be required. The composite[2] trigger specifies that should conditions meet the base trigger 'bt1' more than 4 times within a 1-day interval, further adaptation may be required.

```
<BehaviourPolicy>
        <BaseTrigger ID=”bt1”>
                <Subject/>
                <Provider/>
                <Attribute type=”permisRole”>Contractor<Attribute>
                <Target>PayrollSystem</Target>
                <Action>getEmpPayslip</Action>
                <Rate>
                        <Threshold>5</Threshold>
                        <Interval>1</Interval>
                        <TimeScale>min</TimeScale>
                </Rate>
        </BaseTrigger>
        <CompositeTrigger ID=”ct1”>
                <BaseTriggerID>bt1<BaseTriggerID>
                <Rate>
                        <Threshold>4</Threshold>
                        <Interval>1</Interval>
                        <TimeScale>day</TimeScale>
                </Rate>
        </CompositeTrigger>
</BehaviourPolicy>
```

Figure 8. SAAF Behavior Policy

The SAAF controller generates behavior statistics from each successful access request that matches the relationship and conditions described in each trigger. At first no adaptation is triggered, as the rogue subject's initial set of access requests will not fire any triggers within the behavior policy. However, access requests are continually monitored by the monitor components within the SAAF controller, building up a view of the rogue subject's usage and their attribute assignments.

The first abnormal behavior that the SAAF controller responds to is when the conditions of the base trigger 'bt1' are met, identifying that one subject has accessed the *get employee pay slip action* more than 5 times per minute. The SAAF controller must now analyze the identified misbehavior, before solution selection begins, by identifying which subject from which IdP has performed the 5 access requests within a minute interval.

The solutions policy (Figure 9) specifies four solutions to solve any abnormal behavior identified by conditions in the behavior policy. Solution one (S1) allows for the removal of a subject's attribute from his/her identity provider, impacting only a single individual. Solution two (S2) impacts everyone with the permisRole=Contractor attribute, regardless of the IdP, by removing the ABAC/RBAC permission that allows the Contractor attribute to execute the get pay slip payroll permission. Solution three (S3) impacts all subjects from an IdP, by removing the credential validation rule stating the IdP can assign the attribute contractor. Finally, Solution four (S4) impacts every subject that is managed by the AZP by removing the ABAC/RBAC policy that exists within the deployed AZP. There is also a default solution of 'do nothing', which applies to all instances of abnormal behavior. The default solution is important as the realization of all defined solutions may cause a greater negative impact on the federation, than opposed to allowing a case of malicious behavior to continue.

[2] The composite trigger (ct1) builds upon base triggers. In this scenario, the composite trigger only builds upon one base trigger, however for producing more complex conditions, several base triggers can be used.

```
<SolutionPolicy>
        <Solution>
                <Action>
                        <Operation>removeSubjectAttribute</Operation>
                </Action>
                <TriggerID>bt1</TriggerID>
                <TriggerID>ct1</TriggerID>
        </Solution>
        <Solution>
                <Action>
                        <Operation>removeAttributePermission</Operation>
                </Action>
                <Action>
                        <Operation>buildPolicy</Operation>
                </Action>
                <Action>
                        <Operation>activatePolicy</Operation>
                </Action>
                <TriggerID>ct1</TriggerID>
        </Solution>
        <Solution>
                <Action>
                        <Operation>removeAttributeAssignment</Operation>
                </Action>
                <Action>
                        <Operation>buildPolicyFile</Operation>
                </Action>
                <Action>
                        <Operation>activatePolicy</Operation>
                </Action>
                <TriggerID>ct1</TriggerID>
        </Solution>
        <Solution>
                <Action>
                        <Operation>deactivatePolicy</Operation>
                </Action>
                <TriggerID>ct1</TriggerID>
        </Solution>
</SolutionPolicy>
```

Figure 9. SAAF Solutions Policy

The SAAF controller executes solution analysis for the identified abnormal behavior by evaluating each applicable solution and tailoring it to the relationship identified by the trigger 'bt1'. Solution analysis tailors the first solution (S1 – removeSubjectAttribute) by identifying the unique persistent ID (PID) associated with the subject (provided by the subject's identity provider), along with the valid attributes used to gain access. This results in a SOAP message (Figure 10) being generated, requesting the contractor IdP effector to remove the contractor attribute from the rogue subject, identified by the subject's PID (bb85c0aa1b55ade46a047bd60375ed9c872a6b58). The contractor IdP effector is capable of identifying the subject's location in their attribute repository by the PID supplied in the SOAP message (in this case, mapping the PID to an LDAP distinguished name).

```
<soapenv:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
<soapenv:Header/>
<soapenv:Body>
  <requestAdaptation soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
     <operation xsi:type="xsd:string">removeAttribute</operation>
     <subjectID xsi:type="xsd:string">
               bb85c0aa1b55ade46a047bd60375ed9c872a6b58
     </subjectID>
     <serviceProviderID xsi:type="xsd:string">https://SP.localhost</serviceProviderID>
     <attributeType xsi:type="xsd:string">permisRole</attributeType>
     <attributeValue xsi:type="xsd:string"></attributeValue>
     <reason xsi:type="xsd:string">Malicious behaviour</reason>
  </requestAdaptation>
</soapenv:Body>
</soapenv:Envelope>
```

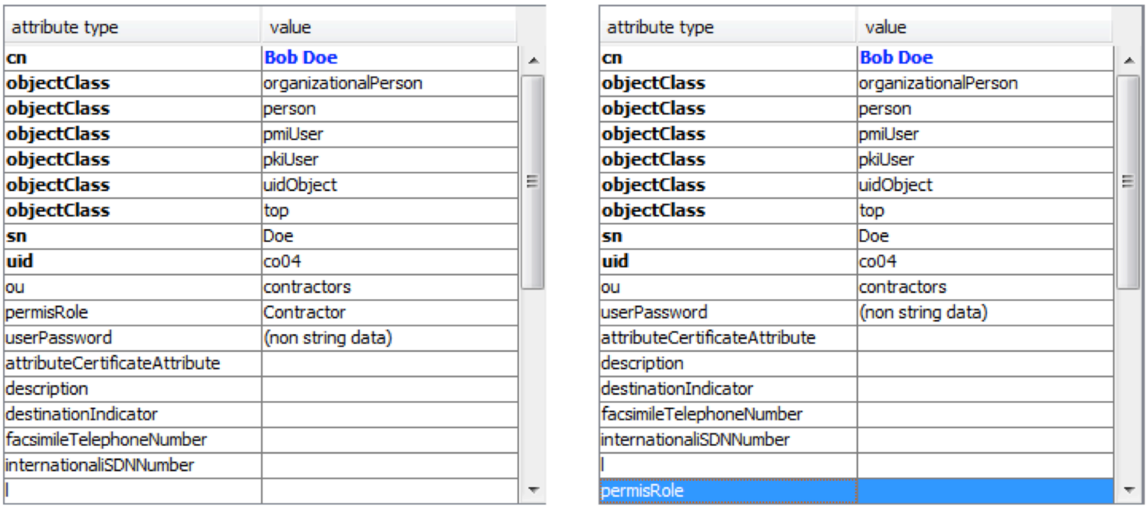
Figure 10. Contractor IdP effector request, remove subject-attribute assignment

However, as the Contractor IdP has been hijacked, the abnormal behavior progresses to a state where multiple rogue subjects are now identified as conforming to the base trigger 'bt1'. Each time SAAF undergoes solution analysis and solution selection, the SAAF controller removes each offending subject's permisRole=Contractor attribute assignment.

After four such events (within a single day interval), the trigger conditions for composite trigger 'ct1' are met. Now more definitive solutions are analyzed (S1, S2, S3, S4) in order to respond to this persistent case of abnormal behavior. Solution selection eventually results in solution three (S3) being selected, whereby the solution enforces the removal of a credential validation rule that trusts the Contractor IdP to assign the Contractor attribute to its subjects. This is due to a greater weight of subjects causing misbehavior from the Contractor IdP, over subjects who are not causing misbehavior from the Contractor IdP or the Business IdP. The selected solution results in a set of actions being generated, whereby the credential validation rule (RoleAssignment, Figure 7) of {Contractor IdP, permisRole=Contractor} is removed from the authorization model, and a new authorization policy is generated that can be used within the PERMIS standalone. The impact of this solution means that all subjects from the Contractor IdP, with the Contractor attribute, will no longer be able to execute permissions associated with the Contractor attribute.

5.3. *Results*

The case study resulted in two types of solutions executed within the self-adaptive authorization infrastructure. The first (S1) refers to individual requests of subject-attribute removal, whereby several SOAP requests were sent to the Contractor IdP effector to remove the permisRole=Contractor attribute from abusive subjects. As the Contractor IdP effector is configured to trust the request from the SAAF controller, these attributes are removed. Figure 11 provides a snapshot taken prior to and after one execution of one of the adaptation requests to the contractor IdP's effector. The LDAP entry for Bob Doe (PID: bb85c0aa1b55ade46a047bd60375ed9c872a6b58 in SAAF) has an objectClass of pmiUser[3], which allows it to hold a permisRole attribute. Prior to adaptation this attribute contains the Contractor value, post adaptation the Contractor value has been removed as a result of the request described in Figure 10. Effectively this prevents the subject Bob Doe from using the permisRole=Contractor attribute within future access requests, as the SAML assertion, issued by the IdP to the service provider, does not contain the permisRole=Contractor attribute.



attribute type	value
cn	**Bob Doe**
objectClass	organizationalPerson
objectClass	person
objectClass	pmiUser
objectClass	pkiUser
objectClass	uidObject
objectClass	top
sn	Doe
uid	co04
ou	contractors
permisRole	Contractor
userPassword	(non string data)
attributeCertificateAttribute	
description	
destinationIndicator	
facsimileTelephoneNumber	
internationaliSDNNumber	
|	

Pre-Adaptation

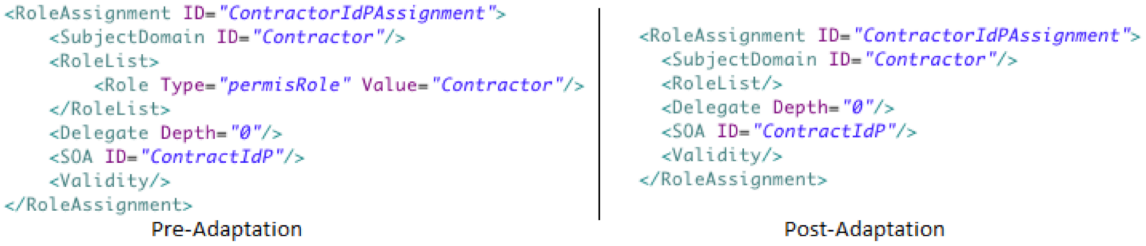
attribute type	value
cn	**Bob Doe**
objectClass	organizationalPerson
objectClass	person
objectClass	pmiUser
objectClass	pkiUser
objectClass	uidObject
objectClass	top
sn	Doe
uid	co04
ou	contractors
userPassword	(non string data)
attributeCertificateAttribute	
description	
destinationIndicator	
facsimileTelephoneNumber	
internationaliSDNNumber	
|	
permisRole	

Post-Adaptation

Figure 11. Snapshots of the Contractor IdP LDAP directory, captured in Jxplorer LDAP viewer

[3] The 'permisRole' attribute is added manually to the LDAP schema in order to provide the PERMIS standalone authorization server a particular type of attribute to use for authorization.

Subject-attribute adaptations provide a fine-grained solution to solving abnormal behavior over policy adaptations. These adaptations rely on the IdP abiding by the SAAF controller's request. However, IdPs are capable of reissuing a subject's attributes allowing the conditions for abnormal behavior to either remain or be reinstated. The abuse may escalate when the IdP reissues the removed attributes. This results in the 2nd type of solution (s3) being realized, a modification to the service provider's authorization policy in regards to credential validation. Figure 12 displays a snapshot of the PERMIS authorization policy deployed in the PERMIS standalone, pre- and post-adaptation. The RoleAssignment rule in the policy defines the trust relationship between the service provider and the Contractor IdP. Previous to adaptation, the service provider trusts the Contractor IdP to issue the permisRole=Contractor attribute to any subject in the Contractor's domain for any validity period. Post adaptation the Contractor IdP is trusted to issue no attributes (defined by an empty role list). This prevents any subject from the Contractor IdP being authorized by the PERMIS standalone, regardless of whether the subject's SAML assertion contains the permisRole=Contractor attribute or not, as it is no longer valid. This solution provides a coarser grained and less volatile solution to solving abnormal behavior, over subject-attribute adaptations, as the service provider's authorization policy overrules external factors (such as IdP issued attributes). However, there is greater risk, as all subjects within the Contractor IdP will be affected, not just the subjects that have misused their access.



```
<RoleAssignment ID="ContractorIdPAssignment">
    <SubjectDomain ID="Contractor"/>
    <RoleList>
        <Role Type="permisRole" Value="Contractor"/>
    </RoleList>
    <Delegate Depth="0"/>
    <SOA ID="ContractIdP"/>
    <Validity/>
</RoleAssignment>
```

Pre-Adaptation

```
<RoleAssignment ID="ContractorIdPAssignment">
  <SubjectDomain ID="Contractor"/>
  <RoleList/>
  <Delegate Depth="0"/>
  <SOA ID="ContractIdP"/>
  <Validity/>
</RoleAssignment>
```

Post-Adaptation

Figure 12. Snapshots of the Service Provider's PERMIS Authorization Policy

The case study was executed in six stages, whereby in each stage an additional malicious subject was introduced to execute a high throughput of authorization requests (which would ultimately break behavior rules). Through our simulation we captured the escalation in planning, solution selection and execution of adaptations as shown in Table 1. At the end of each stage a successful adaptation had occurred, preventing one or more subjects from continuing the identified abuse. Each stage lasted 90 seconds with remaining non-malicious subjects executing authorization requests throughout the 90-second period, in conformance to usage limits defined in behavior rules (with a constant throughput of 3 requests per minute). Once normal subject throughput had stabilized, the malicious subject was introduced.

In addition to solutions executed, we captured performance measures that denote the response time of the SAAF controller, from the point of identifying a case of abnormal behavior to the point that the behavior can no longer continue (for the subject concerned). This includes the time taken for system effectors to carry out an adaptation successfully and respond to the SAAF controller with confirmation. To gain a performance average, the case study was repeated 10 times, under the same conditions with the same instance of the executing SAAF controller.

For the identification of malicious activity in stages 1 to 3 relating to trigger bt1, and resulting in the execution of solution 1 (S1), the average response time ranged from 113.5ms to 162.4ms. However, by the forth stage the SAAF controller has now identified four rule breakages in relation to trigger bt1. As a result, trigger ct1 is invoked, requiring the SAAF controller to consider additional solutions; this causes the response time to increase to an average of 297.67ms. In the fifth stage, despite continued non-conformance in relation to trigger ct1, solution 1 is repeatedly executed, as solutions 2 to 4 are deemed too consequential in SAAF's solution selection phase. Finally in the sixth stage solution 3 is selected, resulting in the contractor credential validation rule being removed from the authorization policy. The average response time of solution 3 is significantly higher in comparison to the execution of solution 1; this is due to a required restart of the PERMIS standalone, in order to activate a newly created authorization policy.

	Triggered Rule	Malicious Subject ID	Identified Solutions	Executed Solution	Avg. Response Time (ms)	STDEV
1	bt1	0f23c42b…	S1	S1	113.5	26.77
2	bt1	36c29160…	S1	S1	162.4	52.91
3	bt1	01afed25…	S1	S1	151.3	52
4	bt1+ct1	bb85c0aa…	S1, S2, S3, S4	S1	297.67	33.8
5	bt1+ct1	566f86da…	S1, S2, S3, S4	S1	248.67	71.77
6	bt1+ct1	c81c6d12…	S1, S2, S3, S4	S3	824.78	75.31

Table 1. Escalation of case study adaptations, and performance results

5.4. Comparison to Current Technology

Current technology is confined to authorization infrastructures that build upon authorization models yet include further controls to implement concepts to assert whether or not subjects should be awarded access. One such example is the use of obligations and conditions within RBAC / ABAC policies to compliment authorization constraints (i.e., subject ‘s’ with role ‘x’ can execution permission ‘y’). Obligations and conditions allow for specific rules that can help reduce the risk of insider threat. For example, a subject must conform to obligations where the subject is required to read a ‘license agreement’ before continuing, or meet the condition that they are not accessing a resource past a particular time of day. In particular, conditions combined with conventional authorization constraints can establish usage control rules, similar to the base triggers defined within the SAAF controller.

The PERMIS policy schema allows such obligations and conditions within a PERMIS authorization policy. We have used a condition to replicate the base trigger ‘bt1’ described in the case study for comparison. Figure 13 identifies the replicated base trigger, which simply states an environment variable labeled ‘ratePerMin’ must be less than a value of 6. The subject’s actual rate per minute is maintained and calculated by the resource policy enforcement point (PEP) that the subject is requesting access through. Upon each access request the subject makes, their ratePerMin environment parameter is sent alongside the subject’s valid attributes and requested resource/action in the access request to the PERMIS standalone. The PERMIS standalone decides upon access whilst taking into account the condition rule defined in the given policy permission.

```
<TargetAccess ID=”ContractPayrol”>
        <RoleList>
            <Role type=”permisRole” Value=”Contractor”/>
        </RoleList>
        <TargetList>
            <TargetDomain ID=”PayrollSystem”/>
            <AllowedAction ID=”getEmpPayslip”/>
            <AllowedAction ID=”runPayroll”/>
        </TargetList>
        <IF>
            <AND>
               <OR>
                   <LT>
                      <Environment Parameter=”ratePerMin” Type=Integer”/>
                      <Constant Type=”Integer” Value=”6”/>
                   </LT>
               </OR>
            </AND>
        </IF>
</TargetAccess>
```

Figure 13. Trigger ‘bt1’ represented in the PERMIS Authorization Policy

Usage control configured into the PERMIS authorization policy is limited to only managing individual subject usage at a per resource level, therefore it is only possible to compare the SAAF prototype based on base triggers and individual adaptations taken against a subject’s IdP. Composite triggers consider a wider range of data from multiple resources and subjects, and are therefore out of scope of this comparison.

To compare, we repeated the first stage of the case study experiment with usage control configured into the PERMIS standalone. We maintained the same conditions as the experiment performed with SAAF, whereby 10 normal subjects executed a throughput of 3 requests per minute in a period of 90 seconds. Once throughput had stabilized we introduced a single malicious subject with a high throughput of 100 requests per minute. This was repeated 10 times to gain an average of performance for measuring PERMIS's response time in denying the subject access (after the subject's usage limit was met). We found that PERMIS was able to deny access in response to a usage control violation with an average of 10.8ms, and standard deviation of 5.58. However, the 'deny' in authorization could only temporarily prevent the subject from gaining access. Once the subject's rate of requests dropped below 5 requests per minute, the subject began receiving grants of access again.

In comparison to the SAAF prototype, usage control in PERMIS is predominately faster in responding to usage violations, yet only temporarily prevents the malicious subject from continuing. It is plausible to argue that lengthy usage control limits (such as limits defined in weeks, months, years) will temporarily prevent the malicious subject from continuing for a greater amount of time, making it possible for human controllers to respond in a timely manner. However, this approach relies on the human controller to respond and would be inefficient in preventing malicious activity carried out over a short interval of time.

5.5. *Discussion and Limitations*

We have demonstrated the feasibility of managing authorization, by autonomic adaptation of authorization policies and subject-attribute assignments, in a federated authorization infrastructure based on behavioral analysis. Model transformations have been shown to be an effective way in adapting authorization policies, considering the fact that these policies were never intended to be adapted autonomously. Regarding subject-attribute assignments, which are traditionally managed by administrators, we have also shown that these can also be adapted autonomously. However, the solution selection in the current implementation, despite solving the abnormal behavior detected, does not represent a best choice solution for the given scenarios.

We have compared the SAAF prototype to the limits of current technology in authorization services (PERMIS standalone). As a result we identify that although techniques such as usage control can impact and potentially slow down malicious activity, it cannot prevent identified malicious activity from continuing permanently. The SAAF prototype (in comparison) can be considered to impose additional risks within the federation. This is especially the case if considering the subject privileges removed belonged to a critical subject, as opposed to simply temporarily denying access. However, the damage caused by a malicious subject through persistent abuse (for instance as a result of credential stealing) could equally present as much risk in not taking permanent actions, as the SAAF prototype has been shown to achieve. In addition, if usage control techniques were to be deployed across multiple resources it would not be possible for an authorization service, such as the PERMIS standalone (or similar services), to assess total usage across all subject sessions, whereas a SAAF controller is capable of monitoring and assessing combined sessions of usage at multiple resources.

In light of these risks, we identify that the current implementation of the SAAF prototype has a number of limitations that we propose to address as the research continues. First and foremost we have no metrics for describing the scale of misbehavior. Not all misbehaviors are equally disastrous. Some may only cause a minor irritation or inconvenience to the organization, whilst some may be serious enough to jeopardize the on-going viability of the business. Consequently we need to introduce a scale component into the behavior policy, which we have termed impact. Related to this, we also have no equivalent metric for describing the scale of a solution. Removing the permissions from all role/attribute holders is clearly orders of magnitude greater in impact than either of the previous modifications, and is dependent upon the number of role/attribute holders.

The next limitation is that solutions are currently pre-defined for each of the misbehaviors (in the solutions policy), and these solutions are chosen based on a calculation of the subjects that are impacted, as described in Section 3.4.1. The dimensions used represent an artificial utility for a solution, which alone is not enough when considering which solutions to realize. Multiple dimensions that compute the utility or impact must be considered, such as, impact to organizations and their subject base (e.g., through loss of functionality, ability to service customers, process orders and invoices etc.), and the probability that the behavior is indeed abnormal. Once solutions have appropriate impact dimensions associated with them, SAAF will have a scale by which to compare one solution with another, and with a given misbehavior, so that the direct linking of solutions to misbehaviors via their policies can be removed.

6. *Related Work*

There are few works that attempt to solve the problem of misuse of access rights during run-time and using self-adaptive techniques, although there are some approaches that attempt to rule out misuse completely in an attempt to reduce the risk of insider threat.

Usage control (UCON) [5] extends traditional access control methods through further definition of rules to primarily manage a subject's access by assessing subject usage. It uses mutable attributes (captured by conditions and obligations) about the subject's access usage as part of the access control decision process. The pretext to this could arguably be that incorporating these mutable attributes as part of the decision process can prevent abnormal behavior. Whilst the UCON model is sophisticated in identifying and managing a subject's usage, it only allows for short-term solutions in managing abnormal behavior. Once a subject's level of usage has 'cooled down' the subject can continue. In comparison to the SAAF prototype if a subject repeatedly meets their usage limits, we assume the subject to be potentially malicious, which requires persistent solutions like those that are implemented in SAAF. An advantage UCON does provide over the SAAF prototype is the ability to impact a subject's access during their session of access, whereby if UCON rules are broken access is disrupted immediately during the subject's session. SAAF is confined by its ability to only react post subject access requests.

Trust PDPs [26] and trust policies [27] also can improve upon traditional authorization. The use of trust policies is a method in which either a group of users or an individual's trust is calculated, for example, based on the attributes they own. In some cases, the level of trust of a user is associated with the cost of carrying out an action, e.g., associating cost to a credential. The combined cost of those credentials will establish how trustworthy that user is. This particular method, although may improve upon more deserved access decisions, does not cover the potential that a trusted user could turn rogue, whereby using their gained trust to abuse their access. Trust could also be viewed from a different perspective whereby reputation (behavior) is involved [4]. For instance, a user's level of trust is calculated based on how they use the different services and whether they use services correctly. This method is better suited to preventing a subject's ability to abuse access rights, as abuse over time would result in the subject becoming untrustworthy. However a trust approach is limited, as no concrete actions are taken to prevent the subject from continuing abuse completely, meaning services with lower levels of required trust can still be abused. Logical attestation [28] builds on authorization, yet purposed towards the reasoning of behavior exhibited by applications, rather than human subjects. It allows for the assessment of trust of applications within an operating system as part of the authorization process, which is successful in preventing untrustworthy behavior. However, applications and systems are far more predictable than human users, meaning the classification of behavior is a more concrete process and irrelevant in application to SAAF's own analysis requirements.

Some systems attempt to actively resolve abnormal behavior, yet not in the context of federated authorization infrastructures. Examples include active intrusion detection systems, such as WebStalker [29], and credit card profiling systems [30]; however both are highly tuned to their target domain. Active IDSs work at the network level and adapt firewall rules to prevent certain types of network traffic. Credit card profiling is aimed at preventing fraud, and is limited to nature of credit card actions, in comparison to multiple target resources and actions with associated different risks and impacts. Other works attempt to resolve abnormal behavior through the dynamic configuration of security policies [31], where adaptations are defined within security policies, in which security constraints have alternative branches based on conditions. However, similar to logical attestation, the work is purposed predominately for the control of access by mobile programs (applications).

As our work builds upon self-adaptive systems, it takes inspiration from systems that have already achieved autonomic management, yet in different contexts. The Rainbow Framework [32] manages architectural self-adaptation, and demonstrates the management of a web based client-server system to ensure optimal availability of web assets (e.g., by increasing the amount of available servers). SAAF follows a similar process to Rainbow, yet rather than adapting the system architecture it adapts the controlling assets of a system. Rainbow also utilizes a self-adaptive language called Stitch [33]. Stitch has provided the basis for our event-response model used within the SAAF controller, referred to as triggers and solutions.

7. *Conclusion*

There is an inherent need for autonomic management of authorization infrastructures given the spread of protected resources and the existence of authorized users over multiple domains. In this paper, we have presented a Self-Adaptive Authorization Framework (SAAF), in which the SAAF controller is a key component, as a solution to autonomic management of federated authorization infrastructures. The approach used is focused on managing federated role/attribute based authorization models (RBAC/ABAC), and the MAPE-K autonomic computing reference model. We have described SAAF's conceptual design as well as the implementation of a prototype, focusing on how SAAF generates adaptations based on configuration and behavioral models of the authorization infrastructure. One advantage of SAAF, compared with more traditional approaches, is its responsiveness when reacting to circumstances that require the authorization infrastructure to protect itself against attacks. Although, we have demonstrated SAAF's capabilities and benefits, in its current form there are some limitations. First, SAAF requires a large amount of trust to be placed on it. In particular, SAAF must play the role of trusted ROOT, and act as the Source of Authority for both service providers and identity providers (IdP). The reason being that not all IdPs would be comfortable to allow a third party to affect their user attribute assignments. Second, the accuracy of SAAF adaptations is also reliant on the specification by the service provider of applicable solutions to patterns of malicious behavior, and this is not the most appropriate solution for socio-technical systems that are able to change in unpredictable terms.

Our future work involves the further development of SAAF, specifically, the definition a multi-attribute decision problem to improve the utility function used to select adaptation solutions. We will draw upon work from trust access control [27], cost associated trust access control [4], and utility [18] in order to build a formal framework for specifying clear controls that prevent wrongful adaptation. Further research into SAAF will also focus on the marriage of SAAF with other technologies that aid in identifying misuse, such as intrusion detection technologies that are capable of analyzing misuse at the resource level.

Acknowledgment

Co-financed by the Foundation for Science and Technology via project CMU-PT/ELE/0030/2009 and by FEDER via the «Programa Operacional Factores de Competitividade» of QREN with COMPETE reference: FCOMP-01-0124-FEDER-012983, and an EPSRC grant for studentship.

References

[1] ANSI. "Information technology – Role Based Access Control". ANSI INCITS 359-2004.

[2] ITU-T Rec X.812 (1995) | ISO/IEC 10181-3:1996 "Security Frameworks for open systems: Access control framework".

[3] H. Debar, M. Dacier and A. Wespi, "Towards a taxonomy of intrusion-detection systems," Comput. Netw 31, Apr 1999, pp. 805-822.

[4] M. Serrano, S. Meer, J. Strassner, S. Paoli, A. Kerr and C. Storni, "Trust and Reputation Policy-Based mechanisms for Self-protection in Autonomic Communications," In Proceedings of the 6th International Conference on Autonomic and Trusted Computing, (ATC 09), Springer-Verlag, 2009, pp. 249-267.

[5] R. Sandu and J. Park, "Usage Control: A Vision for Next Generation Access Control," In Computer Network Security 2776, Springer-Verlag, 2003.

[6] ID:Analytics, White paper.: Analysis of Internal Data Theft, 2008.

[7] A.P. Moore, D.M. Cappelli, T.C. Caron, E. Shaw, D. Spooner and R.F. Trzeciak, "A preliminary model of insider theft of intellectual property," Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications, vol. 2, 2011.

[8] R. Booth, H. Brooke, and S. Moriss, "WikiLeaks cables: Bradley Manning faces 52 years in jail". In The Gaurdian, November 2010. Available from http://www.guardian.co.uk/world/2010/nov/30/wikileaks-cables-bradley-manning.

[9] C. Bailey, D. W. Chadwick, and R. de Lemos, "Self-Adaptive Authorization Framework for Policy Based RBAC/ABAC Models". Proceedings of the 2011 IEEE Ninth International Conference on Dependable, Autonomic and Secure Computing (pp. 37–44). Washington, DC, USA: IEEE Computer Society. doi:10.1109/DASC.2011.31.

[10] D.W. Chadwick, G. Zhao, S. Otenko, R. Laborde, L. Su and T.A. Nguyen, "PERMIS: A modular Authorization Infrastructure", Concurrency and Computation: Practice and Experience 20, Aug. 2008, pp. 1341-1357.

[11] R. L. "Bob" Morgan, Scott Cantor, Steven Carmody, Walter Hoehn, and Ken Klingenstein, "Federated Security: The Shibboleth Approach". Educause Quarterly. Volume 27, Number 4, 2004.

[12] OASIS "eXtensible Access Control Markup Language (XACML) Version 2.0".

[13] D.W. Chadwick, S. Otenko and T.A Nguyen, "Adding support to XACML for multi-domain user to user dynamic delegation of authority," International Journal of Information Security 8, Feb. 2009, pp. 137-152.

[14] J.O. Kephart and D.M. Chess, "The Vision of Autonomic Computing," Computer 36, Jan. 2003, pp. 41-50.

[15] Y. Brun, G.M. Serugendo, C. Gacek, H. Giese, H. Keine and M. Litoiu, "Engineering Self-Adaptive Systems through Feedback Loops," In Software Engineering for Self-Adaptive Systems, Springer-Verlag, 2009, pp. 48-70.

[16] L. Shi and D.W. Chadwick, "A controlled natural language interface for authoring access control policies," Proc. ACM Symp. Applied Computing (SAC 11), ACM, 2011, pp. 1524-1530.

[17] Czarnecki and Helsen, "Feature-based survey of model transformation approaches". IBM Systems Journal, 2006. doi:10.1147/sj.453.0621.

[18] J. O. Kephart, and R. Das, "Achieving Self-Management via Utility Functions," Internet Computing, IEEE, vol.11, no.1, pp.40-48, Jan.-Feb. 2007doi: 10.1109/MIC.2007.2.

[19] C. da Silva and R. de Lemos, "Dynamic plans for integrations testing of self-adaptive software systems," Proc. 6th International Symp. Software engineering for adaptive and self-managing systems (SEAMS 11), ACM, 2011, pp. 148-157.

[20] D. Steinberg, F. Budinksy, M. Paternostro and Ed. Merks, "EMF: Eclipse Modeling Framework, 2nd Edition". Addison-Wesley Professional. Dec 2008. ISBN-10: 0-321-33188-5.

[21] SimpleSAMLphp Version 1.9.2. Available from http://simplesamlphp.org.

[22] OASIS "Security Assertion Markup Language (SAML) Version 2.0".

[23] V. Koutsonikola and A. Vakali, "LDAP: Framework, Practices, and Trends," In IEEE Internet Computing, September/October, 2004, pp. 66-72.

[24] A. Olson, K. Bostic and M. Seltzer, "Berkeley DB". Proc. FREENIX Track, USENIX Annual Tech. Conf, 1999.

[25] C. Bailey, D.W. Chadwick, R de Lemos, and K. W. Siu, "Enabling the Autonomic Management of Federated Identity Providers". In Proceedings of the 7th International Conference on Autonomous Infrastructure, Management and Security (AIMS 2013). Submitted for publication.

[26] K. Böhm, S. Etalle, J. Den Hartog, C. Hütter, S. Trabelsi, D. Trivellato and N. Zannone, "A flexible architecture for privacy-aware trust management," In Theoretical and Applied Electronic Commerce Research 5, Aug. 2010, pp. 77-96.

[27] S. Bistarelli, F. Martinelli and F. Santini, "A formal framework for trust policy negotiation in autonomic systems: abduction with soft constraints". In Proceedings of The 7th International Conference On Autonomic And Trusted Computing, Springer-Verlag, Berlin, Heidelberg 2010, pp 268—282.

[28] E. Gun Sirer, W. de Bruijn, P. Reynolds, A. Shieh, K. Walsh, D. Walsh, and F.B Scneider, "Logical Attestation: An authorization architecture for trustworthy computing". In ACM SOSP, 2011.

[29] Haystack Labs, Inc.Stalker 1997. Available from http://www.haystack.com/stalk.htm.

[30] T. Fawcett and F. provost, "Adaptive Fraud Detection". Data Min. Knowl. Discov. 1, 3 (January 1997) 291-316. DOI=10.1023/A:1009700419189 http://dx.doi.org/10.1023/A:1009700419189.

[31] B. Hashii, S. Malabarba, R. Pandey, and M. Bishop, "Supporting reconfigurable security policies for mobile programs". In Proceedings of the 9th international World Wide Web conference on Computer networks: the international journal of computer and telecommunications networking, June 2000, p.77-93.

[32] D. Garlan, S. Cheng, A. Huang, B. Schmerl and P. Steenkiste, "Rainbow: Architecture-Based Self-Adaptation with Reusable Infrastructure" In Computer 37, 10 (October 2004), 46-54. DOI=10.1109/MC.2004.175 http://dx.doi.org/10.1109/MC.2004.175.

[33] S. Cheng, D. Garlan, and B. Schmerl, "Stitch: A language for architecture-based self-adaptation," In Journal of Systems and Software 85, 12 (December 2012), 2860-2875. DOI=10.1016/j.jss.2012.02.060 http://dx.doi.org/10.1016/j.jss.2012.02.060