Behavioural Control

David W Chadwick
University of Kent
School of Computing
Canterbury, UK
+44 122782 3221
d.w.chadwick@kent.ac.uk

Christopher Bailey
University of Kent
School of Computing
Canterbury, UK
+44 122782 3628
c.bailey@kent.ac.uk

Rogério de Lemos
University of Kent
School of Computing
Canterbury, UK
+44 122782 3628
r.delemos@kent.ac.uk

ABSTRACT
We describe the self-adaptive authorization framework (SAAF), an autonomic self-adapting system for federated RBAC/ABAC authorization infrastructures. SAAF monitors the behaviour of users, and when it detects abnormal behaviour, it responds by adapting the authorization infrastructure to prevent any further abnormal behaviour. The models and components of SAAF are described, as well as the current limitations and where future research is still needed.

Categories and Subject Descriptors
D.4.6 [**Security and Protection**]: *Access controls*;

General Terms
Management, Security.

Keywords
Self-adaptation, authorization, autonomic access control, computing security, RBAC, ABAC, behavioural control.

1. INTRODUCTION
Usage control seeks to control the use of a particular resource after its initial access, so that future accesses are also controlled [1]. In this respect it is similar to digital rights management [2]. In this position paper we take a broader look at controlling the use of resources, through analysing users' behaviour. By monitoring all accesses to all resources, we can determine when a series of accesses, by one or more users, becomes outside the expected norms of behaviour. Our system then stops this abnormal behaviour by automatically adapting the access control system so that further abnormal or abusive behaviour is prevented. Our system is thus an example of an autonomic access control system, which is self-monitoring, self-adapting, and self-correcting.

1.1 Motivation
Our work is in part motivated by the case of Private Bradley Manning. During July 2010 it is alleged that Private Manning, a US army intelligence analyst, downloaded over 0.25 million classified US military documents from a US Department of Defence website [3]. Assuming that the US intelligence 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 that the authorization system performed its function correctly. Any monitoring of the authorization system on a request by request basis would not pick up any abnormal behaviour as it processed the analyst's access requests according to its access control policies. Usage control would similarly not have detected any usage problems on any single file, assuming an analyst was allowed to copy an accessed file onto a memory stick for further study and later analysis. Even if usage control had detected a usage problem, such as copying to a memory stick, and had forbidden it, no further action would have been taken even after the multiple occurrences of such events. However to a human administrator, monitoring the system use in real time, numerous similar requests from the same user to access different files in a short period of time would have flagged up inappropriate behaviour.

Unfortunately the cost of performing real time human monitoring is prohibitively expensive in most cases. Furthermore, making rapid changes to the system to stop further abuses is much more problematical for a human administrator. Analysing the misbehaviour, determining the course of corrective action to take, and then activating the chosen actions, might have taken a human administrator a significant amount of time, compared to the speed that a computer can do this. Consequently our research proposes to build an autonomic self-adaptive access control system that can automatically detect abnormal access control behaviour and apply corrective actions to the authorisation infrastructure. We call our system SAAF – A Self-Adaptive Authorization Framework – for policy based authorization systems. Note that SAAF is designed to be able to both restrict and enable user access, when abnormal behaviour is detected. An example of enabling user access, would be when a doctor has break the glass access rights to any patients' records, and indicates on breaking the glass for a patient's record that he now has a therapeutic relationship with that patient. SAAF would update the patient's record to record the new relationship, so that break the glass would no longer be needed.

This paper is an update of our SAAF, which was originally described here [4].

The rest of this position paper is structured as follows. Section 2 describes our models: that of the underlying federated RBAC/ABAC authorisation system, and that of the self-adaptive authorization framework that manages this. Section 3 concludes with a discussion of the current limitations and the research that

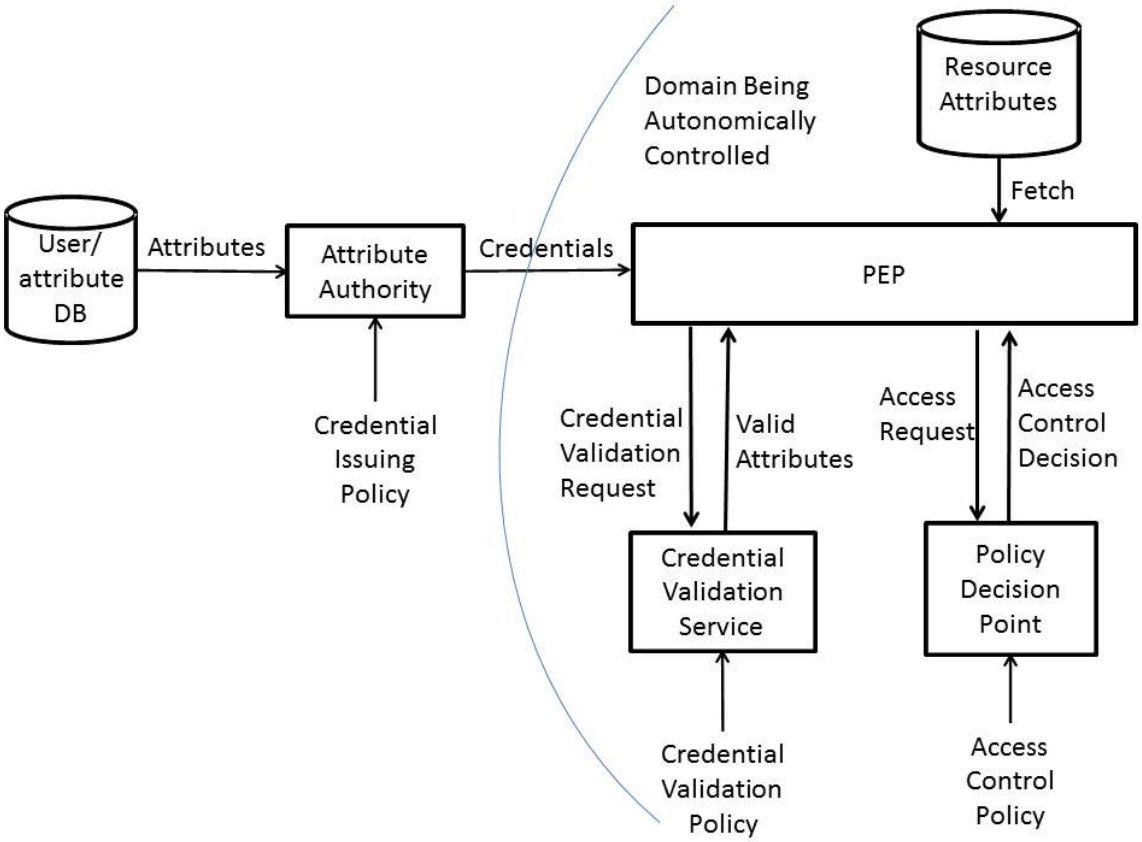

Figure 1. The Federated RBAC/ABAC Model.

still needs to be done in order to build a fully functioning prototype SAAF.

2. MODELS

2.1 Federated RBAC/ABAC Model

Figure 1 shows our model of a federated RBAC/ABAC system that we wish to autonomically control. In this model, we only show the objects that are relevant to of our autonomic access control system. We do not show users, since the system does not actually directly control them. Instead, it controls the *access* of users to resources, via the following system components:

- the Attribute Authorities that assign role/attribute credentials to users,
- the Credential Validation Service that validates user credentials,
- the resource attributes (metadata) that hold user information, and
- the Policy Decision Point which grants or denies users access to resources.

In a federated system, attribute authorities (AA) in different domains hold sets of user attributes in their locally managed databases. When a user wishes to access a federated resource from his web browser, the resource owner or service provider (SP) typically redirects the user's browser to the AA, which authenticates the user then assigns the user a digitally signed attribute assertion (or credential) according to its local Credential Issuing Policy. In Shibboleth [5], for example, this policy comprises the attribute release policies of both the user and the AA.

In a federated system that is capable of attribute aggregation the user may obtain several credentials from different AAs before attempting to gain access to the SP's resources, or the SP may pull credentials from various AAs during the process of granting access. Note that figure 1 does not show the actual protocol messages or web message flows, but only the logical flow of objects that are to be controlled by the autonomic system.

The user's browser presents his/her credentials to the SP's Policy Enforcement Point (PEP) in order to gain access to the SP's resources. The PEP validates these credentials by passing a credential validation request to its locally trusted Credential Validation Service (CVS), and receiving a set of valid attributes in return. The CVS is controlled by the SP's Credential Validation Policy that provides the rules for determining which AAs are trusted to assign which attributes to which users. This is the process of validating the user-role assignments from the traditional RBAC model.

The PEP fetches the attributes of the requested resource, and passes these, along with the user's valid attributes, to the Policy Decision Point (PDP) via an access request. The PDP grants or denies the user access to the requested resources according to its access control policy. This is the process of validating the role-permission assignments from the traditional RBAC model. The PDP returns its access control decision to the PEP, which then acts accordingly.

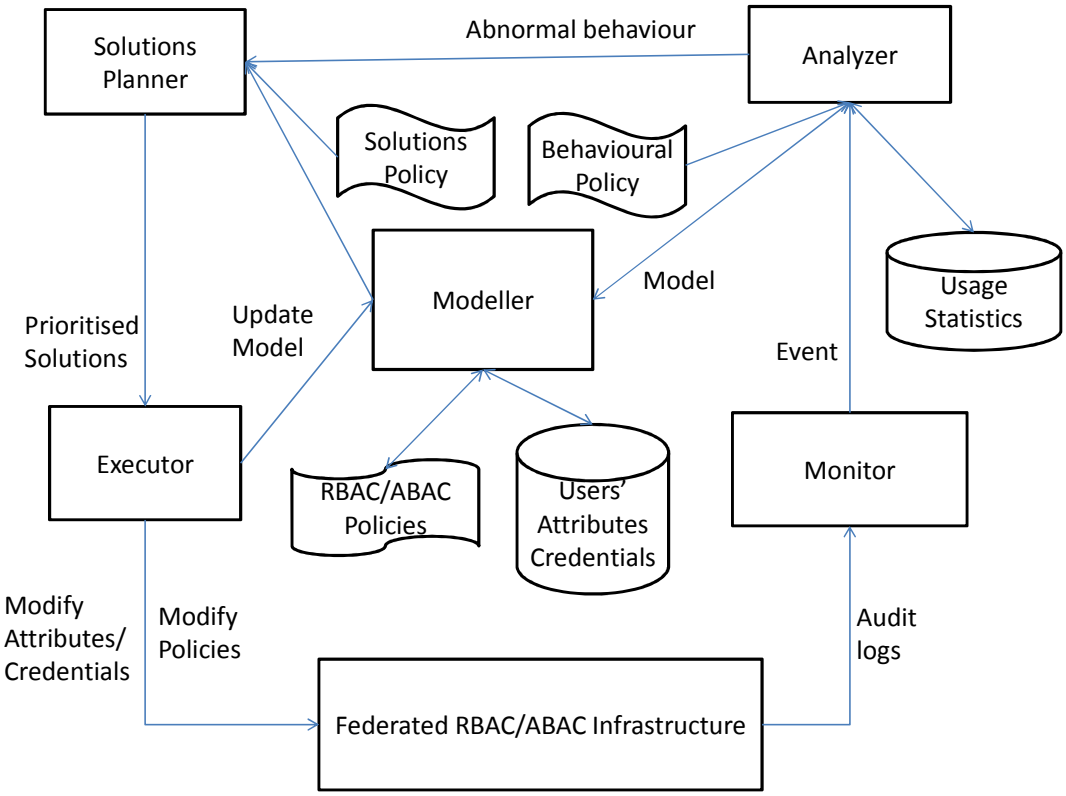

Figure 2. SAAF Components

The three policies, resource attributes, user attributes and user credentials of the RBAC/ABAC system are the six assets that our self-adaptive access control framework (SAAF) will automatically control.

We assume that the SP/PEP records in some locally secure audit log both its requests to the CVS and PDP and their responses. These log records will be used by SAAF to monitor the behaviour of the federated RBAC/ABAC system.

2.2 Self Adaptive Authorisation Framework (SAAF)

Figure 2 shows the components of our proposed self-adaptive authorisation framework. A federated RBAC/ABAC infrastructure that conforms to the federated RBAC/ABAC model presented above, becomes a single component of SAAF. It produces audit logs and is controlled by its policies as described above. SAAF monitors the behaviour of the users of the federated RBAC/ABAC system by inspecting its logs. When SAAF detects abnormal user behaviour it will attempt to alter this by enacting one or more solutions, which will modify the assets of the RBAC/ABAC system, as described below.

The Modeller contains a model of the assets of the actual RBAC/ABAC system that is being autonomically controlled, modelling the 6 assets shown in Figure 1. If the actual system does not have an asset from the federated RBAC/ABAC model, e.g. no credential validation policy, then this will be reflected in SAAF's asset model. Whilst different

RBAC/ABAC systems will use different policy languages to construct their policies e.g. XACML [8], PERMIS [9], the Modeller uses an abstract representation of these, using model transformations based upon an OWL ontology that we developed in a previous project when writing natural language access control policies (which are themselves policy language independent) [7]. Each of the policies needs to be reproduced in SAAF's model, so that SAAF's asset model reflects the actual RBAC/ABAC system being controlled. As changes are made to the underlying policies, SAAF's view of the RBAC/ABAC policies is kept synchronised with them (by the Executor). We do not expect to duplicate each of the AA's user/attribute databases in SAAF's model. Instead SAAF will be initialised with as much information as each AA is willing to release, which in the worst case could be nothing. As each user tries to access one of the SP's resources, the Monitor will detect this from the audit logs and notify the Analyser. If the Analyser notices that this user/role/attribute/credential is not in SAAF' user database it can add it, so that the user database will grow with time to reflect the AAs' databases. Similarly SAAF can be provided with a model of the SP's resource attribute database, or it can build it itself from the audit logs.

The Monitor component of SAAF is responsible for monitoring the usage of the RBAC/ABAC infrastructure, by reading in the audit logs, in their proprietary format, and extracting from them the events which are of interest to the SAAF Analyser, such as role X accessed resource Y at time t. Depending on the infrastructure of the federated environment, SAAF may use multiple Monitors to gain the information it needs. Each Monitor will be specific to its target application. These events are passed to the SAAF Analyser.

The Analyser keeps a usage statistics database that records the frequency of the various events that are passed to it. One event may produce several sets of statistics, such as the number of accesses a particular role or user has performed in the last minute/hour/day, the frequency a resource has been accessed, the

total number of grants per time period etc. The Analyser is controlled by a Behavioural Policy set by the SP administrator (see Figure 3). This provides the behavioural norms of the RBAC/ABAC system, such as: the number of accesses by a role per minute/hour/day, the frequency of access to a particular resource, the number of invalid credentials that are received per time period, the frequency of system grants and denies, etc. Each behavioural norm has an associated cost, which represents the cost to the SP of the norm being exceeded, and of no corrective action being taken. The Analyser determines if the users of the RBAC/ABAC system are behaving as expected or not, as determined by the behavioural policy. This is akin to behavioural analysis performed in intrusion detection systems (IDSs) [6]. Abnormal behaviour could be due to several different reasons, such as wrongly specified policies in the RBAC/ABAC system, misuse of resources by authorised people, or attacks by unauthorised people. If the Analyser determines that abnormal behaviour has occurred it informs the Solutions Planner about this (see Figure 4).

The Solutions Planner is driven by a solutions policy, set by the SP administrator, which contains the various solutions that are available to counteract the detected abnormal behaviour. For example, abusive user behaviour can be counteracted by denying the abusive user(s) further access to the SP's resource. Federated users can be denied access to a federated resource via any of the following actions:

- removing the user's attributes from the AA's database
- modifying the AA's credential issuing policy
- revoking a user's already issued credentials
- removing resource attributes which identify the user
- modifying the SP's credential validation policy
- modifying the SP's access control policy

Each of these solutions has an associated cost. For example, revoking the credentials of a single user is far less costly to the SP than modifying the PDP's access control policy so as to deny all users access to the abused resource(s). The SP administrator is required to place a cost against each of the proposed solutions, so that they can be compared to the cost of the detected abnormal behaviour. The Solutions Planner needs to compare the Solutions Policy against the model of the federated RBAC/ABAC system, as held by the modeller, in order to draw up a list of prioritised solutions which are more cost effective than leaving the system alone. It may be that in some cases of minor abnormal behaviour, such as a student downloading dozens of journal papers in a few minutes, the cost of preventing the abnormal behaviour is greater than the cost of the abuse, and so no corrective action will be taken. However, if the abuse were to continue in a sustained fashion, then at some point it would become cost effective to take the corrective action, for example, once the student's downloads exceed a hundred journal papers per hour. The Solutions Planner sends its prioritised list of cost effective solutions to the Executor.

The role of the Executor is to implement the most cost effective solution, but if this fails, to implement the next highest priority one until one succeeds. The Executor comprises an Orchestrator and many different Interface Components (ICs) that communicate with their respective components of the RBAC/ABAC infrastructure. The Orchestrator converts the most cost effective solution into a set of instructions, which it sends to the ICs that are capable of modifying the various components of the federated authorization infrastructure. Once a solution has been completed and executed by all relevant ICs the Orchestrator updates SAAF's asset model to ensure that SAAF has a synchronised view of the actual RBAC/ABAC authorization infrastructure. The Executor needs to know the specific protocols, policy languages etc. being used by the monitored RBAC/ABAC system so that it can incorporate the correct ICs.

Some of the RBAC/ABAC assets being controlled are held in the SP's local domain, and therefore SAAF can be given permission to modify these directly. However, some of the assets belong to the domains of the remote AAs (i.e. the credential issuing policies and users' attributes and credentials), and therefore SAAF would not normally have permission to modify these. We propose to solve this in the following way. As part of the federation agreement, an AA must either give the SP's SAAF permission to directly update its assets (i.e. credential issuing policy, user attribute database or credential revocation list) or agree to provide a web listening service for SAAF to send update messages to, and to respond to these updates with confirmation messages within a specified time period. In this way the Executor can either directly perform the solutions itself, or can notify the remote AA of the required solution, then wait for the specified time for a response. If no response is received in the specified time it can determine that the solution has failed to be enacted and can move to the next solution in the list.

Note that only the Executor and the Monitor are dependent upon the implementation details of the monitored RBAC/ABAC system, as all the other SAAF components use their own internal formats for modelling the RBAC/ABAC system, recording usage statistics and specifying their policies. Thus the majority of SAAF is independent of the implementation details of the RBAC/ABAC system that is being controlled, allowing SAAF to be usable with different implementations of RBAC/ABAC through the implementation of application specific Monitors and Executor ICs.

3. DISCUSSION, LIMITATIONS, CONCLUSION

This position paper presents our current research on "behavioural control", which is an attempt to monitor and autonomically control the behaviour of users within a federated RBAC/ABAC authorisation system. The research is still at an early stage. To date we have concentrated on specifying the models, their essential components, and the authorisation assets that can be managed in order to control users' behaviour.

Modelling work that is still required is to:

- Specify the Behavioural Policy in detail,
- Specify the Abnormal Behaviour in detail,
- Specify the Solutions Policy in detail,
- Determine the full set of statistics that need to be recorded
- Specify the algorithms for determining abnormal behaviour and determining solutions.

We have to determine which semantics and rules the behavioural policy language will support based on the complexity of the constructs and the time it will take to evaluate the rules against the monitored behaviour.

```
<BhrRule id="FreqGetSameRes">
 <Resource>"+"</Resource>
 <Action>Get</Action>
 <Op>GT</Op>
 <Rate>
   <Number>5</Number>
   <Time>1</Time>
   <Unit>Min</Unit>
   <Cost>250</Cost>
 </Rate>
 <Rate>
   <Number>20</Number>
   <Time>1</Time>
   <Unit>Day</Unit>
   <Cost>500</Cost>
 </Rate>
</BhrRule>
```

Figure 3. An Example Behavioural Rule

An example behavioural rule is given in Figure 3. This states that the rate of requests for the Get action on the same resource (indicated by "+") must be no greater than 5 requests per minute, or 20 requests per day, and the cost of violating each rule is 250 and 500 units respectively. This is a very simple behavioural rule. More complex rules may involve specifying sequences of actions, such as downloading a file followed by copying it to a flash disk, on the same or different resources. Even more complex rules may involve identifying the same sets of actions being carried out by different users. Significant research is still needed in this area.

Figure 4 shows an example of flagging abusive abnormal behaviour. This signals which subjects (identified by their attributes) have performed which abnormal actions on which resources, and what the cost of this is to the organisation. In this example one user, a student with ID 123456, from Kent, has performed abusive Get actions on two different resources, at a cost of 1000 units to the organisation (500 per resource as stated in Figure 3).

```
<Abuse>
 <Subjects>
   <Subject>ID="123456",Role="student", O="kent.ac.uk"
   </Subject>
 </Subjects>
 <Actions>
  <Action>ID="Get"</Action>
 </Actions>
 <Resources>
  <Resource>ID="www.kent.ac.uk/library"</Resource>
  <Resource>ID="cs.kent.ac.uk/projects"</Resource>
 </Resources>
 <Cost>1000</Cost>
</Abuse>
```

Figure 4. An Example of Abusive Abnormal Behaviour

The Solutions Policy describes the various corrective actions that can be taken, and the cost to the organisation of performing each one of them. Figure 5 shows an example.

```
<SolutionsPolicy>
 <RemoveSubject>
  <ID>Type=Role,Value="Student"</ID>
  <Cost>100</Cost>
 </RemoveSubject>
 <RemoveSubject>
  <ID>Type=Role,Value="Professor"</ID>
  <Cost>1000</Cost>
 </RemoveSubject>
 <UpdateCVP>
  <RemoveAA>LDAPDN="O=Kent,O=AC,C=UK"</RemoveAA>
  <Cost>100000</Cost>
 </UpdateCVP>
 <UpdateCVP>
  <RemoveAtt>Type=Role,Value="Student"</RemoveAtt>
  <Cost>20000</Cost>
 </UpdateCVP>
 <UpdateCVP>
  <RemoveUA>
   <Attribute>Type=Role,Value="Student"</Attribute>
   <Subject>LDAPDN=""</Subject>
   <AA>LDAPDN="O=Glasgow,O=AC,C=UK"</AA>
  </RemoveUA>
  <Cost>2000</Cost>
 </UpdateCVP>
 <UpdateACP>
  <RemovePA>
   <Attribute>Type=Role,Value=Student</Attribute>
   <Action>ID=Get</Action>
   <Resource>ID="www.kent.ac.uk/library"</Resource>
  </RemovePA>
 <Cost>1000</Cost>
 </UpdateACP>
</SolutionsPolicy>
```

Figure 5. An Example Solutions Policy

This policy states that removing a single student user from the system has a cost of 100 units, whereas removing a single professor has a cost of 1000 units. Updating the credential validation policy to completely remove the Kent attribute authority (which means that no credentials issued by Kent will be trusted) costs 100,000 units, whereas completely removing the student role (which means that no students from anywhere will be able to access any resource) has an associated cost to the organisation of 20,000 units. In comparison, removing the user-attribute assignment from Glasgow, so that only its student roles are no longer considered valid, has an associated cost of 2000 units. Updating the access control policy permission attribute assignment for the student role, so that students can no longer Get files from Kent's online library, has an associated cost of 1000 units.

Once the schema for the two policies has been completed, we still will not know how practical or difficult it will be for administrators to set and manage them. The more complex the behavioural rules and solution policies are, the more difficult it will be for administrators to specify all of them. Conversely, if they are too simplistic, they will not be sufficient to control all types of abusive behaviour. Thus significant research is still needed here.

For SAAF to effectively manage a federated RBAC/ABAC authorization infrastructure requires accurate and relevant

adaptations against the infrastructure's assets, in light of abnormal behaviour. However, the effectiveness of each adaptation is directly correlated to how well the mechanism for identifying unexpected behaviour operates and the behavioural rules that exist. For example, SAAF can only execute an adaptation as a result of a user breaking rules defined in the behavioural policy. If only a small subset of rules are defined to capture behaviour on critical/sensitive access requests then SAAF will only be able to adapt the infrastructure's assets in relation to those sensitive requests.

It is essential than SAAF's view of the RBAC/ABAC infrastructure, defined by SAAF's asset model, is synchronised with the actual RBAC/ABAC infrastructure. Policies that are currently active in the infrastructure must also be portrayed in the asset model. If a policy changes in the target infrastructure then the asset model must also change. From SAAF's perspective this is maintained through the Executor updating the asset model after every successful adaptation. However this does not cover human interactions, whereby security administrators change active policies without SAAFs knowledge. In a federated environment this becomes even more of a problem, because multiple distributed credential issuing policies are at risk of being changed by many different AA administrators. Therefore a mechanism for monitoring changes in policies must be utilised, with automated updates to SAAF's asset model. This will require the SP to trust the external AAs and give them direct or indirect access to update SAAF's asset model.

We have not yet started implementation. This is the next step. Our plan is to use the PERMIS authorisation system as the first actual RBAC/ABAC system to be controlled. PERMIS contains APIs for accessing and updating all its policies, as well as user attributes and credentials. So SAAF will be able to directly control all of the assets. PERMIS also records the access requests and responses using the XACML request/response context format, so parsing the log records should not be too difficult. We therefore believe that integration with PERMIS will be relatively straightforward. Designing the algorithms for determining abusive behaviour and the appropriate solutions will be more challenging aspects of the research.

4. ACKNOWLEDGMENTS

This research is funded by an EPSRC studentship.

5. REFERENCES

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

[2] Subramanya, S.R., Yi, B.K. "Digital Rights Management". IEEE Potentials, March-April 2006. Vol.25 Issue 2. pp 31 - 34

[3] The Guardian "WikiLeaks cables: Bradley Manning faces 52 years in jail" http://www.guardian.co.uk/world/2010/nov/30/wikileaks-cables-bradley-manning [accessed: 20 Feb 2011]

[4] C. Bailey, D.W. Chadwick, R. de Lemos, "Self-Adaptive Authorization Framework for Policy Based RBAC/ABAC Models," Proc. 9th Internationl Conference on Dependable, Autonomic and Secure Computing, (DASC 11), 2011, pp. 37-44.

[5] 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

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

[7] L. Shi and D. W. Chadwick. "A controlled natural language interface for authoring access control policies". Proceedings of the 2011 ACM Symposium on Applied Computing. TaiChung, Taiwan, 21-24 March 2011, pp 1524-1530.

[8] OASIS "eXtensible Access Control Markup Language (XACML) Version 2.0" OASIS Standard, 1 Feb 2005

[9] 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.