Partial Outsourcing: A New Paradigm for Access Control

Joerg Abendroth*
Distributed Systems Group
Trinity College, Dublin
Joerg.Abendroth@cs.tcd.ie

Christian D. Jensen†
Informatics & Mathematical Modelling
Technical University of Denmark
Christian.Jensen@imm.dtu.dk

ABSTRACT

Various security models have been proposed in recent years for different purposes. Each of these aims to ease administration by introducing new types of security policies and models. This increases the complexity a system administrator is faced with. Ultimately, the resources expended in choosing
amongst all of these models leads to less efficient administration.

In this paper, we propose a new access control paradigm, which is already well established in virus and SPAM protection as partial delegation of administration to external expertise centres. Well-known vulnerabilities can be filtered out and known sources of attacks can be automatically blocked. We describe how partial outsourcing can be achieved in a secure way. A framework, which enables this process has already been developed.

Categories and Subject Descriptors

D.4.6 [**Operating Systems**]: Security and Protection—*Access Controls*; H.1.m [**Models And Principles**]: Miscellaneous—*Partial Outsourcing*; K.6.5 [**Management of Computing and Information Systems**]: Security and Protection—*Unauthorized access*

General Terms

Design,Security,Management

Keywords

Access Control, Active Software Capabilities, ASCap framework, Partial Outsourcing

*This work is sponsored by a research grant from IONA Technologies PLC.

†This work was partly completed while the author was working at Trinity College Dublin.

SACMAT'03, June 1–4, 2003, Como, Italy.

1. INTRODUCTION

A number of different security models have recently been proposed, which aim to ease the administration of access control [11, 22, 5, 26, 6, 23]. Some of this research focused on the analysis of specific application domains, as demonstrated in 1987 by the Clark and Wilson paper "A Comparison of Commercial and Military Computer Security Policies" [4]. This research proposes new security models that are well adapted for a particular application environment, such as computer supported cooperative work [5], distributed tasks [26] or coalitions [6]. Regardless of the unquestionable quality of this work, its impact outside the security research community has been relatively limited. In contrast, the first paper on role based access control (RBAC) [8] sparked both further research in different aspects of RBAC, as well as industrial development of RBAC systems. Research in RBAC includes information flow [21] in RBAC systems, RBAC implementation in UNIX [7], role-finding methods [10] and development of a formal logic to reason about RBAC systems [19]. The reason for the success of RBAC might be attributed to the fact that RBAC provides a new way of understanding access control. Moreover, RBAC fulfils the role of a simple abstraction for federation of access rights, which simplifies the specification of security policies and administration of RBAC systems. Administrators no longer think in terms of direct access rights of users to files (or subjects to objects), but use the role as a new layer of simplified abstraction. Introducing this new way of thinking is tantamount to allowing scientists to formulate their mathematical problems in their own language, which was done by FORTRAN [14] years ago.

Major paradigm shifts can also be observed in other areas of computer and communication technologies, e.g. the Internet: being initially a purely information sharing media, it developed into an e-commerce platform, and in latest developments a matter of state security concerns. These Internet security concerns can relate to privacy, easy distribution of dangerous information, or simply attacks on infrastructure resources.

Similarly, while early computer system administration consisted of the installation of required tools, today tasks of maintenance dominate. Using the example of email systems it can be seen that the task of keeping up to date with the latest virus and SPAM security threads is tedious. Companies shifted therefore from fully local administrations to partially outsourcing the administration of their email systems by using externally-maintained filtering software.

The dynamics and economics of security outsourcing is well

presented by Bruce Schneier [3, 24]. He contrasts the high costs of employing local security experts and their low average workload within one company, with the benefit of employing an external security advisor whose cost may be amortised by several companies. This means that outsourcing certain aspects of information security will be cost effective. His example of Pilot Network's failed secure network management consulting service underlined the point - outsourcing is only effective in cases where the local company retains control over its security systems. So far, access control has been seen as only fully outsourcable. However, we believe that access control can be partially outsourced. This allows external expertise companies [1] maintain part of the administration, while the individual companies maintain autonomy with respect to security policies and the ability to override the configuration of the external administration.

We believe that the increasing complexity of security policy, specification and the resulting decrease in usability of security mechanisms, warrant a similar approach to access control, and we believe that outsourcing will become the next paradigm shift in access control. Once access control can be partially outsourced to external expertise companies, different developments can take place.

These developments allow a company to partially outsource the access decision to, for example, various internal divisions. The human resource department could acknowledge that an employee did not resign - thus his access is not revoked, the financial department could introduce policies to determine the maximum amount of expenditure, or the employee's local department might introduce policies about working hours and local particularities. Using partial outsourcing these different rules can be provided by each department with relevant knowledge, and no central administration is obliged to understand the needs of all departments. Furthermore, these developments allow outsourcing abstract elements like "protection against remotely exploitable application weaknesses, as far as they are publicly known" (cf. 5.1).

In this paper we present an access control mechanism that allows partial outsourcing of security administration. The mechanism is based on the ASCap framework [1], which aims to provide simultaneous support for multiple security policies and models. We have previously shown that different security models, such as access control lists (ACL), capability based systems or RBAC can be instantiated within the ASCap framework. This is achieved by using the concepts of active software capabilities combined with external security servers.

Another important feature of the ASCap framework is the support for dynamically changing security policies, i.e. it facilitates the evolution of security policies. This can be done in a secure way, and the benefits towards changing the paradigm "*access control is centrally and solemnly administrated by the company*" will be presented in this paper.

The rest of this paper is organised as follows: In section 2, we first present a short introduction of the ASCap framework. We then describe how different elements of the access control mechanism can be outsourced to an external expertise company, and we define different classes of outsourcing and analyse their properties in Section 3. In section 4 a short description of the implementations is given. The proposed mechanism is evaluated in Section 5 through examples of innovative use of partial outsourcing, which illustrate the benefits of this mechanism. Related work is presented in Section 6. Finally, Section 7 presents our conclusions along with directions for future work.

[1]An external company, which provides its expertise, but is not further affiliated with the local company

2. THE ASCAP FRAMEWORK

The basis of our proposal is the active software capability (ASCap) framework, which is described in detail in 'A Unified Security Framework for Networked Applications' [1]. We present here a short summary of the main elements of the ASCap framework.

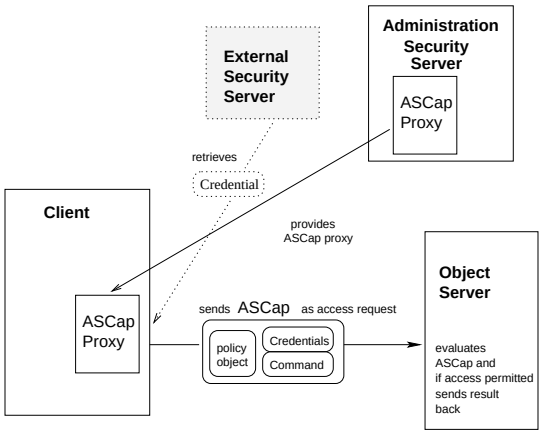

Figure 1: Overview of the ASCap Framework

Figure 1 shows the essential parts of the ASCap framework. Prior to the first access request the client needs to download the *ASCap proxy* from the administration security server, which mostly resides on a different server as the object server. The administration server can influence the overall system behaviour by providing different *ASCap proxies* for different servers. The *ASCap proxy* is instantiated into the security interface of the local client application. It then establishes a secure connection to the object server by use of public key cryptography. The *ASCap proxy* represents part of the object server control interface according to the concept of proxy based authorisation [20]. Access requests are therefore directed to the ASCap proxy, which determines a suitable security policy. Depending on the policy, further network connections to *external security servers* (shown in gray) may be necessary. Each *external security server* returns a credential. The collection of credentials together with the policy object and the full access request (noted as command in Figure 1) is called an ASCap. The *ASCap proxy* sends the access request represented by the ASCap to the *object server*, which evaluates it.

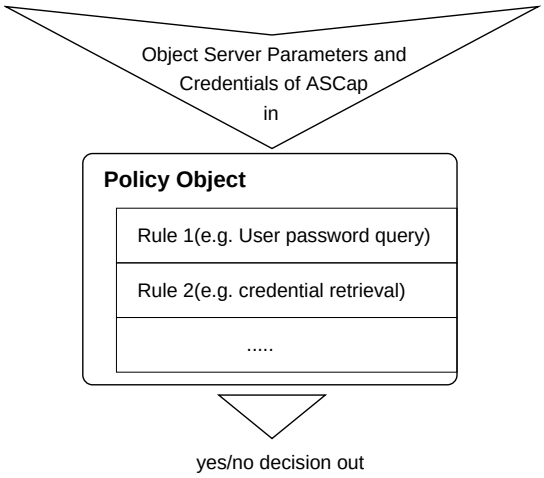

Figure 2: Policy Objects implement Access Control Decision Functions

A schematic overview of the evaluation process is given by Figure 2. The object server uses the credentials of the ASCap together with local parameters as input parameters to the policy object, which returns a simple boolean indicating whether access should be granted or not. The *policy object* of Figure 2 shows another feature – each policy object consists of different rules. The set of rules influences the policy and ultimately the whole system behaviour. That is, a single rule will result in a grant, reject or error decision and the final decision of the policy will be based on these decisions. However, it is possible to allow more then one outcome to be positively accounted toward the final result, e.g. the policy may require the client to contact an accounting server, but whether the server grants or rejects the client is unimportant as long as no error occurs.
To support different security models the ASCap framework differentiates between two modes. The first mode of higher assurance and performance employs a *reference* to a policy object inside the ASCap instead of the full policy object implementation. The *reference* points to an implementation of the policy object stored in a secure repository, on the object server side. It has been shown that various well-known security models can be instantiated using this setup [1].
The second mode has the name '*active capability policy*', and the entire policy implementation is encoded in the policy object and sent along with each access request[2]. The advantage of this mode is a greater flexibility, because dynamic policy changes are made possible by sending different policy objects to the client and ultimately the server. In contrast, the first mode only supports policies stored in the secure repository. In the *active capability mode* a valid concern is security properties. Security depends on the policy object implementation, whether it is safe[3] and "benevolent"[4]. Both can be assured by allowing only signed policy objects from trusted sources.

While security of the access control framework is one concern, usability and ease of administration are other, that are concerns equally important. From this perspective, trusted sources of the active capability mode represent no strict security model, as they can implement any kind of policy or rule they like. Hence, employing trusted sources complicates the task of defining a simple security model. Investigation of current cooperate security infrastructures shows that they already consist of a mixture of different security models. We therefore believe that the flexibility offered by the use of active capability policies are more important than the added complexity of security policy specification.

[2]Certain performance optimising mixtures do exist.
[3]We understand this as integrity preserved.
[4]e.g.not attacking the server.

3. CONCEPTS OF OUTSOURCING ACCESS CONTROL

Mostly, access control in common setups is centrally managed by one reference monitor, trusted computing base or ACL(directory) server. Approaches of decentralised administration or outsourcing in our terms are mainly based on public key infrastructures [30]. These, however, provide only three classes of outsourcing, while partial outsourcing is not possible. In the following we will identify the different classes of outsourcing by examples using the ASCap framework.

3.1 Class α: Single Administration, Internal

This class is understood as the base class, no outsourcing takes place and both administration and object server belong to the same domain. Figure 3 shows this scenario. The circle around the object server symbolises the security domain of a company. Inside this domain all parties are understood to behave with integrity toward the common security goals. Once a client retrieves an ASCap proxy and successfully access the server, the dotted line shows that the client can be understood as obeying the same security goals. Security analysis of this setup is straight forward, because only one point of control exists. Authentication and authorisation of the client are fully handled by the company's own administration server.
Advantages are those of central administration known today. *Disadvantages* include that the full administrational work has to be done by one party and no external expertise can be used.
Implementation example: Access control frameworks employed today, e.g. a domain with a central domain controller, in which password is only known by employees of the local company.

3.2 Class β: Single Administration, External

Figure 4 shows the opposite extreme case: administration is fully outsourced. Again the circle around the object server symbolises the parts fully natively trusted. The external administration server is not included in this domain, because, per definition, the trust in this server is artificial. This time the dotted circle shows the relationship that the client will belong to the domain of the administration server, as it obeys his policies. Finally, the big double circle shows the necessary scope of required trust to make the system work. Only if both the administration server and client are trusted can access take place. The big circle can be reduced to administrational server and object server if appropriate cryptographical protection is in place.
Security depends fully on the behaviour of the external administration server. In outsourcing terms this allows to fully benefit from the expertise of an external company, but requires the external company to also do less sophisticated

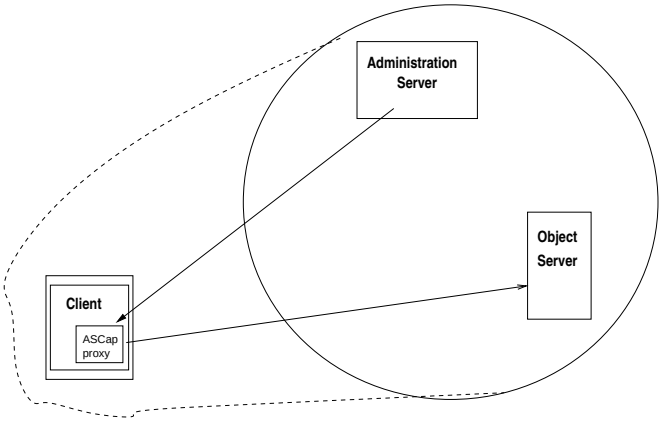

Figure 3: Class α: No Outsourcing, Single Administration

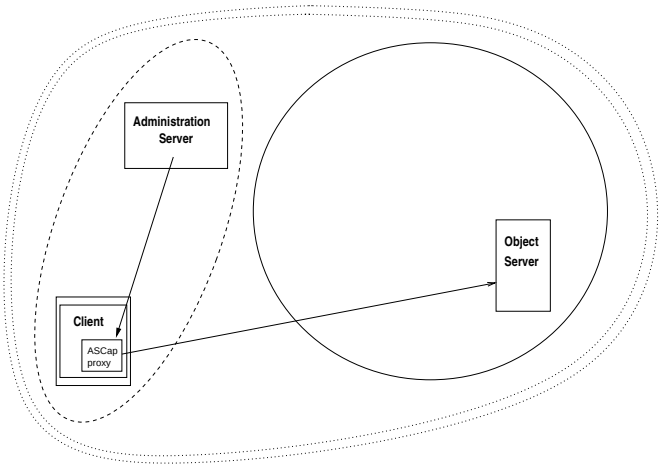

Figure 4: Class β: Fully Administrated by External Company

and time consuming support tasks, such as create user accounts and resetting passwords. *Advantage* is that the full workload is outsourced. *Disadvantage* is that the external company must be trusted.
Implementation example: Similar to class α, but in this case the administration of the central server is done by an external company, e.g. an external company installs and maintains the servers.

3.3 Class γ: Outsourcing via External Security Server

In this class, as shown in Figure 5, administration is done by the company itself. Policies require the client to retrieve a credential of the external security server. This external security server is managed by the external consulting company. The multiple arrows hint that the same external security server might need to hand out credentials to a large number of different clients. Finally, the two dotted circles show that the client is dependent on the external security server, but also has to play by the rules of the administration server. There is no big circle, which indicates that the external security server has no power over the object server, specifically it cannot give away access arbitrarily.
Evaluating *security* of this scenario shows that the company takes the ultimate decision of which security servers to ask, which shows that the company has the most power over the access control. In terms of *security* a malicious external security server can cause, at most, a denial of service. A *weakness* of this setup can be seen in the properties needed for the external security server. The external security server needs to be always accessible. Furthermore, imagining the scenario of a company doing external auditing for several large organisations, it would be the case that all clients of all organisations would need to retrieve a credential of the external company and therefore this external security server would need to deal with a large number of requests. A *benefit* of this setup is the possibility that the external company can decide to only hand out the credentials after different security checks have been successfully performed. Clients known to be malicious can be blocked although they may hold all other required credentials. Moreover as no executable code migrates from the external company into the system, the extent of trust toward the external server is minimal. A certain static is introduced into the system, however, because the number and address of external security servers and credentials will be fixed from the start of the system.
Implementation example: This setup can be instantiated by using external security server policies of the ASCap framework. The client needs to retrieve one or more credentials from these external security servers and send them inside the ASCap to the object server.

3.4 Class δ: Partial Outsourcing Using External Rule Servers

Figure 6 shows the final possibility. The external rule server delivers rule implementations to the local administration server. The administration server combines different rule implementations (here, Rule X+Rule Y) to policy objects and ASCap proxies. The ASCap proxy is sent to the client. The dotted line around the client shows the known trust relationship. The second dotted line around the external rule server and object server indicates a natural trust relationship. This is so, because the external rule implementations are executed by the object server.
For *security* the local administration has the full control over which rule implementations are used. Because the external rule implementations are not the only ones included, the external rule server cannot arbitrarily permit access. However, there is a danger that the rule server providing a rule implementation, which tries to break into the object server. This danger is much smaller than a malicious acting client, and may be further reduced by code checking techniques.
Advantage, as opposed to the previous class, is that the number of administration servers is magnitudes smaller. The flexibility of this approach is also easily demonstrated by letting the rule server provide a rule, which calls an external security server- this class is then converted back to class γ. Further possible advantages are subject to the next section.
Implementation example: The company employs their own administration server, which receives rule implementations as objects from external consulting companies. These rule implementations are compounded to form a policy object, which can be sent inside the ASCap proxy to the client. This time the policy object cannot be stored in a secure reposi-

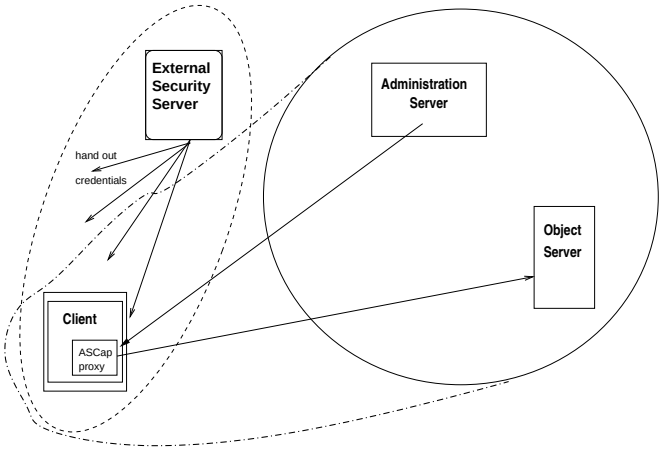

Figure 5: Class γ: Outsourcing Using External Security Server Approach

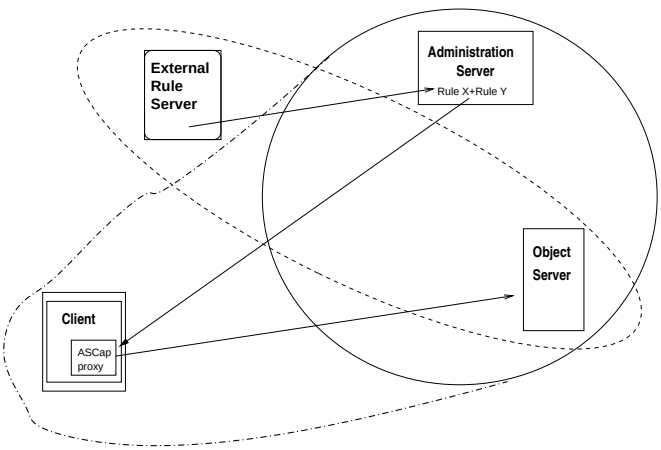

Figure 6: Class δ: Partial Outsourcing Using an External Rule Server

tory, but using the active capability mode the policy object is sent with the ASCap.

3.5 Paradigm Shift: Partial Outsourcing

Today, we believe access control has been seen as only "full" or "not" outsourcable. This is represented by the classes α and β. Kerberos [2, 16] could be seen as implementing class γ. Hereby, the authorisation server could take the role of external security server and the ticket granting server would need to require ticket granting tickets from more than one authorisation server. A simple realm change would instantiate only class β. This remark shows that, although Kerberos would be powerful enough for class γ, this way of access control outsourcing has not been realised.
We believe class δ has not yet been implemented. Allowing access control to be handled jointly by different parties would introduce additional possibilities. As a company is split into different departments, each rule of a policy could be motivated by a different need. For example, the human resource department could introduce checks to determine a person's affiliation, while the financial department might introduce rules about possible responsibilities of making payments. Each department would be able to define and modify its rules independently. Additional "good practise" checks could be introduced by external companies, while none of the parties could cause a full disclosure on its own.

4. IMPLEMENTATION

This section presents a high level overview of the ASCap framework. A detailed description of this framework has been published elsewhere [1], so we only highlight the elements that are necessary to understand the evaluation presented in Section 5.
The ASCap framework has been implemented using JINI 1.0, and jdk.1.3.1-b24 mixed mode[5] Cryptographic algorithms are provided by the cryptix 3.2.0 release. These algorithms are used for establishing a secure communication channel, signing the policy objects and ASCaps. Finally, the reflection package of Sun's Java distribution is used for handling the policy objects inside the ASCaps used at the "active capability mode" (cf. 2).
Using the framework involves setting up a JINI lookup server, which receives a proxy object (*ASCap proxy*) from the *administration server*. The client has to download the ASCap proxy from the JINI lookup server. The ASCap proxy includes the policy object, which is sent in the ASCap to the object server. This allows dynamic change of the policy object and ASCap proxy by modifying the master copy on the administration server. To ensure timely revocation, the JINI lease needs to be short and no lease renewal service employed.
The *External Rule Server* must have direct connection to the administration server, either by secure network connection or other means. The external Rule server provides single rule objects. The administration server combines these rule objects to policy objects, which are signed. Before using external rule implementations the administration server is free to run any code verification mechanism desired on them.
The *External Security Server* is mostly contacted by the client, who wishes to obtain credentials that certify certain properties. External security servers implement their own behaviour and may require the client to fully authenticate itself before providing the credential.
A concern in terms of security might be the use of Java, in which it can be argued that the security of the ASCap framework depends on the ASCap being sent to the Object server. On the client side any programming environment may be used to implement the behaviour of the ASCap proxy, while the server runs the signed policy object. As the policy object originates from a trusted source (the client's own administration server), no danger by it is assumed.

5. EVALUATION

Once the paradigm shift to "partial outsourcing of access control is possible" (cf. 3.5) is fully realised, new benefits can be seen. The scenario used in our evaluation is shown in Figure 7. It includes the client which requests access to objects managed by the object server and an internal administration server that defines the security policies enforced by the ASCap framework. The scenario also includes an exter-

[5]Byte code is partly precompiled or compiled on runtime.

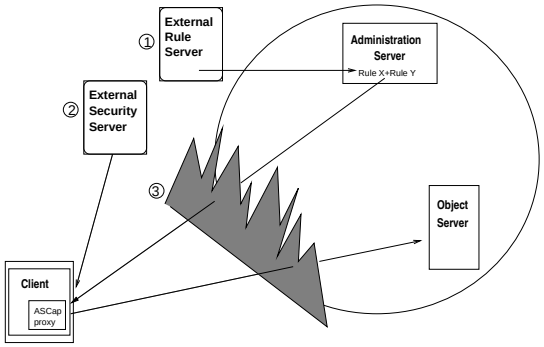

Figure 7: An Example Scenario

nal rule server (1) and possibly an external security server (2) and firewall (3).

5.1 Remote Weakness Filtering

In this case, the external consulting company actively follows security announcements and weakness reports. Whenever an exploit is reported, auditing of customer companies is done and security holes are recorded. However, the external company does not have the power to update the actual software, nor the local administration the office hours to guarantee a timely reaction on their notification. To prevent intrusion during this blackout time, one would want to switch off the vulnerable applications. This can be done in two ways: either the external consulting company employs an external security server, which stops handing out valid credentials (class γ), or the external consulting company updates rules (class δ) for the respective applications to strictly deny access. The latter approach would have the advantage that, instead of a full blackout, only those access requests possibly touching the security hole might be blocked, similar to only email including a virus will be filtered. In an ideal case, security holes could be closed by patching the ASCap framework preventing dangerous access request from getting through.

We simulated this example by having an external rule server transfer rule objects to the administration server's *rulescol* directory via *scp*[6]. The JINI lease time was set to 30 seconds, which might seem short, but because the JINI lookup server was running on the administration server it turned out to be feasible and gave us an acceptable revocation time. A remote exploitable weakness was assumed to be a read request with an information location string longer than 100 bytes. As expected, it was possible to introduce an appropriate check into the system by uploading its implementation as a rule object to the administration server.

5.2 Advanced Pattern Recognition

Advanced patterns are patterns not easily seen from a local perspective. An example is SPAM filtering of email systems. Locally, a domain would not be able to recognise certain spammers, because only a few mails are delivered to this local domain. Being able to monitor email traffic from a broader perspective, it is possible to also recognise these SPAM originators.

Applying advanced pattern recognition to access control could mean monitoring the access request a single client had in a certain period of time. Doing this not on the local, but on a broader level allows the recognition of clients randomly "hacking" into different systems, e.g. by using a newly discovered exploit, and collecting an extraordinary number of access rights.

We implemented a prototype that employed the external security servers of the ASCap framework, thus instantiating class γ. Additional to the standard rules, two different rules, which access external monitoring services were pluggable into the system. This was done by altering the policy object to include also a rule, which requests the credential of this external security server. We implemented a counter on the external security server, which would deny the credential if too many different information directories were accessed in a short amount of time. We assumed that this corresponded to different security domains in a real world setup. Further on, we assumed that one external monitoring service was fully trusted, while the second should only be allowed notification character. Hence, although the behaviour of both external security servers were the same, the rule implementation of checking these required credentials were different. The fully trusted server was able to cause a denial of service (also in the case of unreachability), while the second credentials were not further checked or required - the added value was purely that this external security server would be able to collect statistical information about clients accessing our servers.

5.3 Active Intruder Recognition

Reasoning about intruders shows that different properties can be recognised. In the case of partial outsourcing, an external company can keep up to date about these properties and introduce into the system effective rules to recognise them. Rules in the ASCap framework are not be limited by passive access checks, but active intruder probing can be employed.

Assuming an intruder has, as opposed to a "normal" user, a higher desire to get a service, maybe even any service, his behaviour will differ from the normal users. Or in a different case he may be restricted to security hole exploiting techniques to acquire access rights.

These assumptions can be used to actively recognise intruders without exactly knowing which security hole they used. Given that the system finds itself in an alarming state - for example an unnaturally high system load - it may start to change the environment in a subtle way. The changes will not disturb official users, but cause trouble to intruders. A firewall (Figure 7–3) might disallow a certain protocol or require a reinitiation of the connection. Connections established by a replay attack will become obvious or hijacked credentials will have timed out. Generally, if after such a change in access control rules an user shows different behaviour than before or even does not connect, again a new security hole might have been found. Active intrusion recognition is not automated to this extent today, but only by allowing partial outsourcing it becomes possible to develop such mechanism. *Note: This example has not been implemented, but all required functionalities exist in the ASCap framework. To fully implement this example research about active intruder recognition techniques would need to be conducted. Research into intrusion detection is mostly restricted to passive pattern recognition instead of active probing.*

[6]A network copy tool of the public ssh implementation [32].

5.4 Comparison of Example Implementations

Below in table 1 we have compared the examples, their outsourcing class and employed elements of the ASCap framework. A comment reviews the key element of that specific case.

6. RELATED WORK

Initial development of the ASCap framework (cf. 2) was inspired by research in active capabilities [28, 17]. The goal is to provide a framework that can support different security models by using ideas of hidden software capabilities [9], 'The proxy principle' [25] and 'Protection Reconfiguration for Reusable Software' [13].

The work presented in this paper is similar to projects from Jajodia et. al. [12], which recognises, that in the same system different policies may be desired. Therefore, they develop an authorisation specification language and a flexible authorisation manager to evaluate the policies. The user is able to specify her own policies and a policy library exists. It seems that no further exploration of additional useful properties of their system has been done, therefore it is unclear if it was realised that partial access control delegation might be possible in their system.

Other projects have explored the possibility of delegating access control decisions. Delegation has been the focus of research in the areas of public key cryptography and public key infrastructure [30]. Beside simple delegation [29], previous research has also addressed the problem of decentral administration [27]. While Thompson et al. [27] use a purely certificate based approach, Yialelis et al. [31, 18] provide a way of editing and distributing policies. However, none of the papers envision how a joint administration of access control could be achieved.

Finally we have been inspired by research addressing the consequences of combining systems of different independent administration domains [15]. Kuehnhauser develops a metapolicy language to resolve contradictions. Reading his paper brought us to the conclusion that a change of paradigm in access control is necessary.

7. CONCLUSION AND FUTURE WORK

In this paper we introduced a new paradigm for access control: Partial Outsourcing. We presented a framework ready to provide the required functionalities. The ASCap framework was originally developed for providing a security mechanism to instantiate a wide range of security models. It has a feature called active capabilities, which was used to implement partial outsourcing. Our new paradigm for access control moves away from the idea that access control can only be fully delegated, and allows us to think about only partial outsourcing. Active capabilities allow us to think about adaptive access control. This adaption can be influenced by external sources, which can be included into the policies individually.

We have also discussed different types of outsourcing, their security properties, benefits and disadvantages. It becomes clear that an access control framework can be either very flexible, have a clear security model, or be safe according to a strict security requirements. Therefore, we have shown two different ways of outsourcing as alternatives to central administration. Outsourcing via external security servers seems to be more static, as the only point of influence is the credential provided. Also, when using external security servers no change in rule implementations are possible. A clearer understanding of possible changes in the system might favour this way of outsourcing.

Advantages of partial outsourcing using external rule servers were widely discussed in this paper. Still, we believe the full meaning is broader than expected today. Questions of influence toward instantiated security models, the absolute risk taken in case of where an external administration provider turns malicious and simply: What are other interesting utilisations of partial outsourcing ? - are interesting topics for further research. That extraordinary possibilities exist has been shown by describing how active intruder recognition can use the access control framework to recognise intruders by their behaviour rather than authentication rules.

8. REFERENCES

[1] J. Abendroth and C. D. Jensen. A unified security mechanism for networked applications. In *Proceedings of 18th Symposium on Applied Computing (SAC2003)*, pages 351–357. ACM, March 2003.

[2] J. G. S. B. Clifford Neuman and J. I. Schiller. Kerberos: An authentication service for open network systems. In *Winter 1988 USENIX Conference*, pages 191–201, Dallas, TX, 1988.

[3] C. Bruce Schneier. *Outsourcing Security*. Counterpane webside http://www.counterpane.com/literature.html, 1.12.2002.

[4] D. Clark and D. Wilson. A comparison of commercial and military computer security policies. In *In Proceedings of the IEEE Symposium on Security and Privacy, Oakland, CA*. IEEE, May 1987.

[5] G. Coulouris and J. Dollimore. Security requirements for cooperative work: a model and its system implications. In *ACM European SIGOPS Workshop*. ACM, 1994.

[6] e. a. Eve Cohen. Models for coalition-based access control (cbac). In *7th ACM Symposium on Access Control Models and Technologies (SACMAT 2002)*, pages 97–106, 2002.

[7] G. Faden. Rbac in unix administration. In *Proceedings of the fourth ACM workshop on Role-based access control*, pages 95–101, 1999.

[8] Ferraiolo and Kuhn. Role based access control. In *Proceedings of 15th National Computer Security Conference*, 1992.

[9] D. Hagimont, J. Mossiere, X. R. de Pina, and F. Saunier. Hidden software capabilities. In *International Conference on Distributed Computing Systems*, pages 282–289, 1996.

[10] G. S. Haio Roeckle and R. Weidinger. Process-oriented approach for role-finding to implement role-based security administration in a large industrial organization. In *Proceedings of the fifth ACM workshop on Role-based access control*, pages 103–110, 2000.

[11] R. S. S. Jaehong Park. Towards usage control models: beyond traditional access control. In *7th ACM Symposium on Access Control Models and Technologies (SACMAT 2002)*, pages 57–64, 2002.

[12] S. Jajodia, P. Samarati, V. S. Subrahmanian, and E. Bertino. A unified framework for enforcing multiple access control policies. In *SIGMOD International Conference on Management of Data*, pages 474–485. ACM Press, 1997.

[13] C. Jensen and D. Hagimont. Protection reconfiguration for reusable software. In *Second Euromicro Conference on Software Maintenance and Reengeneering*, pages 74–81, Florence, Italy, March 1998.

[14] e. a. J.W.Backus. The fortran automatic coding system. In *Proceedings of the Western Joint Computer*, 1957.

Example	Class	External Security Server	External Rule Server	Tested	Comment
4.1 Remote Weakness Filtering	γ	x	-	-	External security server stops handing out credentials to block access
4.1 Remote Weakness Filtering	δ	-	x	x	External rule server updates rule implementation on administration server, which updates policy objects
4.2 Advanced Pattern Recognition	γ	x	-	x	External security server receives statistical information as clients need to contact for access. If pattern is recognised credential can be denied and may cause a blockout of the client depending on the actual rule.
4.3 Active Intruder Recognition	δ	x	x	-	Alarm by e.g. increased system load is raised. The external security rule server actively investigates by modifying the environment causing an intruder to show suspicious behaviour.

Table 1: Comparison of the different setups

[15] W. E. Kühnhauser. On paradigms for security policies in multipolicy environments. In *Proceedgins fo 11th International Information Security Conference (IFIP/SEC'95), Cape Town, South Africa*, 1995.

[16] J. Kohl and C. Neuman. The kerberos network authentication service (v5). RFC 1510, Digital Equipment Corporation/ISI, September 1993.

[17] H. M. Levy. *Capability-Based Computer Systems*. Digital Press, Bedford, Massachusetts, 1984.

[18] D. A. Marriott, M. S. Sloman, and N. Yialelis. Management policy service for distributed systems. Technical Report DoC 95/10, Imperial College, London, 1995.

[19] F. Massacci. Reasoning about security: A logic and a decision method for role-based access control. In *ECSQARU-FAPR*, pages 421–435, 1997.

[20] B. C. Neumann. Proxy-based authorisation and accounting for distributed systems. In *Proceedings of the 13th International Conference on Distributed Computing Systems*, pages 283–291, Pittsburgh, Penn, U.S.A., May 1993.

[21] S. L. Osborn. Information flow analysis of an rbac system. In *7th ACM Symposium on Access Control Models and Technologies (SACMAT 2002)*, pages 163–168, 2002.

[22] A. Ott and S. Fischer-Hübner. Rule set based access control as proposed in the 'generalized framework for access control' in linux. In *Karlstadt Univeristy Studies, 2001:28, ISBN 91-89422-63-5*, 2001.

[23] M. Röscheisen and T. Winograd. A communication agreement framework for access/action control. In *Proceedings of the IEEE Symposium in Security and Privacy*, 1996

[24] B. Schneier. *Secret and Lies*. John Wiley & Sons; ISBN: 0471253111, August 2000.

[25] M. Shapiro. Structure and encapsulation in distributed systems: The proxy principle. In *Proceedings of the 6th International Conference on Distributed Computer Systems*, pages 198–204, Cambridge, Massachusetts, U.S.A., 1986.

[26] R. K. Thomas and R. S. Sandhu. Towards a task-based paradigm for flexible and adaptable access control in distributed applications. In *ACM SIGSAC New Security Paradigms Workshop*, pages 138–142, 1993.

[27] M. Thompson, W. Johnston, S. M. and Gary Hoo, K. Jackson, and A. Essiari. Certificate-based access control for widely distributed resources. In *Proceedings of the Eighth USENIX Security Symposium (Security 99)*, pages 215–228, 1999.

[28] W. L. Tin Qian. Active capability: An application specific security and protection model. Technical report, University of Illinois at Urbana-Champaign, 1996.

[29] T. D. Tock. An extensible framework for authentication and delegation. Master's thesis, University of Illinois at Urbana-Champaign, 1994.

[30] Various. Open source pki book, http://opensourcepkibook.sourceforge.net, 1.12.2002.

[31] N. Yialelis and M. Sloman. A security framework supporting domain based access control in distributed systems. ISOC Symposium on Network and Distributed Systems Security, 1996.

[32] T. Ylonen. SSH - secure login connections over the internet. In *Proceedings of the 6th Security Symposium) (USENIX Association: Berkeley, CA)*, pages 37–42, 1996.