Towards Context-aware Security: An Authorization Architecture for Intranet Environments

Chris Wullems, Mark Looi and Andrew Clark
Information Security Research Centre,
Queensland University of Technology, Brisbane, QLD 4000, Australia
{*c.wullems, m.looi, a.clark*}*@qut.edu.au*

Abstract

This paper introduces a context-aware authorization architecture that is designed to augment existing network security protocols in an Intranet environment. It describes the architecture components, the proposed extensions to RBAC that facilitate context-aware access control policy, details of the prototyped implementation, and a number of performance results.

1. Introduction

Ubiquitous computing has been a rapidly growing research area, however its uses have been predominantly targeted at context-aware applications for smart spaces such as smart homes and workplaces. In this paper we present an authorization architecture that uses the context-aware paradigm to augment existing and time-tested technologies such as Kerberos, facilitating fine-grained access control to network resources and effective enforcement of security policies.

In context-aware security, hosts are granted or denied access to resources based on the perceived security of the host. An example of contexts that are relevant to network security include:

- **Host location:** This includes instances where access to a given resource or operation would constitute a breach of security if outside of a trusted location;
- **Network topology between host and application server:** This includes contexts such as connection security, bandwidth, and routing;
- **Host security:** This includes contexts such as operating system patch level, antivirus signature version, host firewall rules, routing tables and filesystem permissions / file-sharing settings; and
- **Host execution environment:** This includes contexts such as currently executing applications that have conflicts of interest and validating the state of executing applications such as resident virus scanners.

Context-aware authorization has a number of requirements traditional access control mechanisms do not provide. First the authorization system must be able to dynamically grant and revoke permissions based on an access control policy and the continually changing context of a user. It cannot be assumed that a given set of authorization credentials will persist for the lifetime of a session. In order to support such access control policy, permissions must be centralized and a common representation of context data must be used.

The addition of context-awareness to access control systems significantly increases complexity, however it is a requirement of our architecture that administration of security policy must retain the same administrative efficiencies afforded by the use of role based access control.

The proposed architecture supports GSSAPI-based applications through the use of Kerberos. Common applications that currently support Kerberos include SMB file sharing, database servers (e.g. PostgreSQL), CVS and Java-based applications though JAAS[1]. The use of GSSAPI[2] allows existing network applications to be easily migrated to the proposed architecture.

The structure of the paper is as follows. First existing and emerging efforts in pervasive computing security are reviewed in Section 2. The architecture and its components are described in Section 3. Implementation details including a description of the testing environment and several performance results of the prototyped architecture are detailed in Section 4. Conclusions are presented in Section 5.

2. Emerging Solutions for Access Control in Pervasive Computing

Recent security efforts in ubiquitous computing have been targeted at security in pervasive applications for

[1] See Java Authentication and Authorization Service `http://java.sun.com/products/jaas/`

[2] See RFC 2743 - Generic Security Services Application Program Interface

Proceedings of the Second IEEE Annual Conference on Pervasive Computing and Communications Workshops (PERCOMW'04)
0-7695-2106-1/04 $ 20.00 © 2004 IEEE

context-aware homes and offices. While our focus is improving security of network applications through the augmentation of context to access control processes, emerging work in this field has introduced a number of interesting approaches to securing pervasive applications in the smart home/ office environment.

The first attempt of authorization in the smart-home environment was proposed by Al-Muhtadi et. al. [1], who approaches authorization by using a scaled-down version of SESAME[3], an extension to Kerberos providing minimal RBAC support. As SESAME is based on traditional network security paradigms and a push model using privilege attribute certificates, it is unable to provide support for context-influenced credentials, nor can it support changes to credentials within a session.

The architecture proposed by Covington et. al. [5] [4] introduces a pull based model that supports changes to credentials within a session. Access control policy in this architecture is based on an extension to RBAC, the Generalized RBAC model [3], where object and environment roles are introduced in addition to traditional subject roles. Object roles contain a membership of resources and environmental roles contain a membership of environmental conditions. Access control decisions are then made based on a policy combining environment roles, object roles and subject roles. While GRBAC provides an improvement in the flexibility of security policy, it introduces complexity in administering access control policies and difficulty in ensuring they are conflict free.

The architecture proposed by Al-Muhtadi et. al. [2] differs from the architecture proposed by Covington et. al. [5] [4], in that it uses first order predicate logic to form its access control policy. While this offers increased flexibility to GRBAC, it is also complex to administer due to the requirement of specifying each access control rule and associated actions individually.

Our approach to access control policy differs in that our work extends RBAC to provide a more flexible activation mechanism for roles, as well as providing role-centric context constraints. This allows for simple access control policy, rather than complex policy definitions that attempt to bind context data to credentials.

3. Architecture Overview

The architecture is designed for use in an Intranet environment due to the requirement of application servers having reliable access to the authorization server, and the difficulty of managing access control policy between multiple administrative domains. It is assumed that access to authorization servers over an Internet link would not perform adequately given the required response times and frequency of communications required by the use of centralized permissions and access control logic. The implementation of the architecture operates in the context of a Kerberos (Microsoft Windows 2000) domain.

The proposed architecture has been designed for context-transparency to application servers in that they do not process any context information. For future applications that may have a requirement for certain types of context data, the application server must have activated an appropriate role that allows updates of required type of context data to be requested. This mechanism allows for the centralized storage of user privacy policies which can determine whether such permission is granted to an application server. Figure 1 illustrates a component view of the architecture.

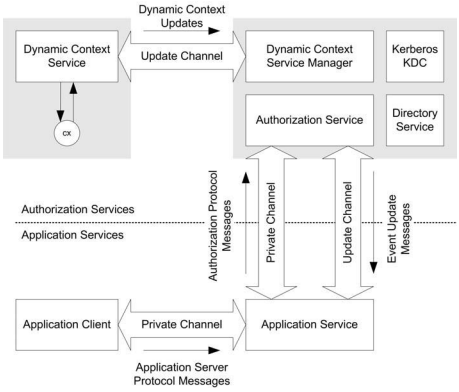

Figure 1. Architecture Components

The following subsections detail the services provided by the architecture.

3.1. Authentication Service

The authentication service is based on Kerberos [7]. An LDAP directory provides the data storage for authentication and authorization data, linking Kerberos principals with roles in the authorization architecture. Kerberos V5 is based on symmetric key cryptography, and is well suited to resource constrained mobile devices. For additional security, devices capable of public key cryptography can perform the initial authentication to the Key Distribution Center (KDC) using public key cryptography as defined in PKINIT [8]. Constrained devices can use a mobile version of PKINIT such as the service proposed by Harbitter [6]. Kerberos is a widely accepted network authentication protocol that is implemented on most Unix variants as well as Microsoft Windows 2000/XP/2003. In addition, support for Kerberos is included in Microsoft Windows CE .NET 4.2.

[3]See A Secure European System for Applications in a Multi-vendor Environment `https://www.cosic.esat.kuleuven.ac.be/sesame/`

3.2. Authorization Service

In this architecture, security context-awareness is modeled through dynamic activation of roles. Standard RBAC roles are extended such that role constraints are evaluated based on activation triggers which may include user request for activation, the activation of other roles, or a context event. Role activation for a given principal depends on the evaluation result of a role's constraints, which are evaluated by the Dynamic Context Service Manager (DCSM). Role activation requests made to the authorization service by the DCSM are only actioned if system-wide constraints, evaluated by the authorization service, are fulfilled.

The architecture contains two types of roles: a standard role as defined below, and a simplified type of role, the task. A task represents a work function, containing a set of permissions (allowed operations for a given object), and is assigned to a role. For example, an "Administrator" role may include a "AddDomainUser" task. This task contains permissions required to add a domain user. Tasks differ from roles in that they have no members and tasks cannot be assigned to another task, but otherwise have the same attributes as a role. Roles in the architecture are stored in the directory and contain the following attributes:

Inherits: Role inheritance is supported through this attribute, where the role can inherit the members, permissions and constraints from parent roles assuming separation of duties constraints are fulfilled. The attribute contains a list of distinguished names of parent roles in the directory.

Members: This attribute contains the role members, represented as a list of distinguished names of users in the directory.

Tasks: This attribute contains the tasks the role contains, represented as a list of distinguished names of tasks for the given application in the directory.

Permissions: This attribute contains the permissions of the role, represented as a list of distinguished names referencing permissions specified for the given application. A permission contains: (1) Permission Class, the type of the permission represented; (2) Permission Name, the name of the object this permission describes; and (3) Actions, the name of actions granted on the specified object.

ActivationTriggers: This attribute contains the names of supported triggers. The currently supported trigger types include context class, role activation / deactivation, or the principal that requests role activation / deactivation. When an event is fired, the roles containing the corresponding trigger are evaluated. The evaluation considers the constraints before making a decision of whether to activate the role for the principal the event was targeted at.

Constraints: The following attributes represent the supported constraints:

- *Requires:* This attribute contains the names of the prerequisite roles that must be active in order for this role to be activated, represented as a list of distinguished names.
- *MutuallyExclusive:* Dynamic separation of duties are implemented as mutual exclusion sets. This attribute contains the name of mutual exclusion sets represented as lists of distinguished names.
- *ContextRule:* This attribute contains a set of statements that define contextual constraints for role activation. The statements are constructed using the following syntax: `<OBJECT> OPERATION <PARAMETER>`. Boolean operators (and, or, not) can be used between consecutive statements: `(<STATEMENT>) <BOOLEAN_OP> (<STATEMENT>).` The variables `$PRINCIPAL`, `$CONTEXT(`*`context_class`*`)`, `$LDAP(`*`distinguished_name`*`)` reference the principal, the current principal's context and the directory object referenced by the LDAP distinguished name. The example ContextRule (Figure 2) allows the role to be activated only if the principal performed mutual authentication with the application and context location is at the principal's registered home or office location.

```
($CONTEXT(dcaa.dcsm.ConnectionSecurity)
  hasMutualAuth) and
(($CONTEXT(dcaa.dcsm.Location) within
  $LDAP(cn=OfficeLocation;cn=...)) or
($CONTEXT(dcaa.dcsm.Location) within
  $LDAP(cn=HomeLocation;cn=$PRINCIPAL;cn=...)))
```

Figure 2. Example ContextRule

The authorization service maintains the roles for each principal that are active in each application in the domain. For a principal to become active, the application service must initialize an access control context with the requested roles on behalf of the user. When a user session ends, the application service ends the context on behalf of the user.

The application service can request an access decision by sending a checkPermission request to the authorization service. The checkPermission function enumerates all the permissions for the given principal's active roles and tasks. The function then checks that the permission, for which the access decision is requested, is implied by a permission in the set of enumerated permissions.

3.3. Dynamic Context Service Manager

The Dynamic Context Service Manager (DCSM) is responsible for the activation and deactivation of roles and tasks in the authorization service based on a principal's context. The DCSM has a context event listener where the triggers defined in roles are registered. The following subsections will provide an overview of its interactions with context services and detail the dynamic context and event update mechanisms.

3.4. Dynamic Context Services

Dynamic Context Services (DCS) are trusted services responsible for acquiring context information either directly or via a third party. A DCS notifies the DCSM of context changes based on its policy that specifies how trust and accuracy are quantified given the raw context data, the frequency at which the context is acquired and updated, and the thresholds for notifying the DCSM.

This allows a DCS to collect context information using paradigms other than the event driven model used in the architecture. In addition, context data can be sourced from other context acquisition frameworks,[4] whilst maintaining the trust and security requirements of context acquisition for access control decisions.

For example, we have implemented a GSM location DCS that queries a GSM location service (Gateway Mobile Positioning Center) every 30 seconds for active principals. The DCS maintains a cache of location data. If a context change exceeds the specified threshold, the updated context is communicated to the DCSM in a dynamic context update message containing a common context representation and the principal the update is relevant to. The DCS is responsible for mapping the principal in the context to corresponding principal in the directory. For example, the GSM location service maps the MSISDN[5] of the user to a principal in the directory.

ContextObjects are used to represent context information as well as containing default methods that manage trust, time of last update and accuracy level determined as by the DCS. An example is the more specific instance of ContextObject, LocationContextObject, which contains a common representation of location, and constructors that convert a DCS' location data (e.g. WGS-84, lat/long, GSM-Timing Advance arc, etc.) to the common representation. This allows multiple sources of location to have the same representation, such that it can be used by policy and contextRule evaluators.

[4] Such as the Georgia Institute of Technology Context Toolkit `http://www.cc.gatech.edu/fce/contexttoolkit/`

[5] Mobile Subscriber ISDN, the number callers use to reach a mobile subscriber.

3.5. Dynamic Context Update Mechanism

The following processes are executed when a dynamic context update is received from a DCS, or an activation / deactivation request event is received from the authorization service.

When a dynamic context update is received, the last ContextObject for a given principal is retrieved from the context cache. The ContextCombiner method of the new ContextObject (and its subclasses) is then used to create a ContextObject combining the context data from the update and the cache. For example, the combining functionality may use historic data to calculate a more accurate context with the potential to increase trust. Finally, the context cache is updated with the new ContextObject.

An event is fired for the given class of ContextObject, such that all roles or tasks with a trigger of this class are evaluated and activated according to the outcome of the evaluation. In the case of an activation / deactivation request event, the specific role requested is evaluated. Successful activation of a role or task fires the evaluation of any roles or tasks which are triggered based on the activation of that role. Note that tasks will not be evaluated unless a role to which the task is assigned is active for a given principal.

3.6. The Event Update Mechanism

The authorization service has an Event Manager that is responsible for informing application services of events that occur. After an application service initiates a secure context with the authorization service, an authorization service-side listener is registered for each application service session. When an event is fired, all event listeners that the event update is relevant to are informed of the event. The event listener for the given application service session sends an event update message over the update channel as shown in Figure 1. The application service can then take appropriate action.

The architecture is designed in such as way as to allow customized events to be implemented. The following events are currently implemented in the architecture:

- **Access Control Context Changed Event:** If a role or task is activated or deactivated for a given principal, the applications in which the principal is active are notified that the access control context for the given principal has changed. The application service can then check that the principals are still authorized to perform an operation by sending a checkPermission request to the authorization service.
- **Dynamic Context Service Manager Failure Event:** In the case there is a failure in the DCSM, the application service is notified and can take evasive action if required.

- **Update Channel Not Responding Event:** In the case of an outage or attack against an update channel, the application service is notified and can take evasive action such as suspending the user's session until the update channel is restored. Failure of the update channel is detected through the use of a heartbeat.

The event update mechanism has been implemented such that future applications that may require context data can be supported through a "Context Data Update Event".

4. Implementation Details

The architecture is centered around the use of open standards such as Kerberos, LDAP and XML messaging. This allows the architecture to operate over different platforms. Microsoft Windows 2000 Server was used as the platform to host the directory and Kerberos functionality. The framework implementation can also run on Linux with MIT Kerberos and OpenLDAP.

The testing platform, as illustrated in Figure 3, was built using a number of Pentium III 833Mhz PCs with 256MB RAM. The authorization data centre was configured with Windows 2000 Advanced Server, and hosted the authorization service and DCSM. To date GSM and Wireless LAN location DCSs have been implemented, although the Wireless LAN location tamper resistance is still being investigated. These services were hosted on the context server. The GSM service requires users preregister their MSISDN, stored as an attribute in the User object in the active directory. We are currently developing host-based security evaluation context services and intend to present them in a future paper.

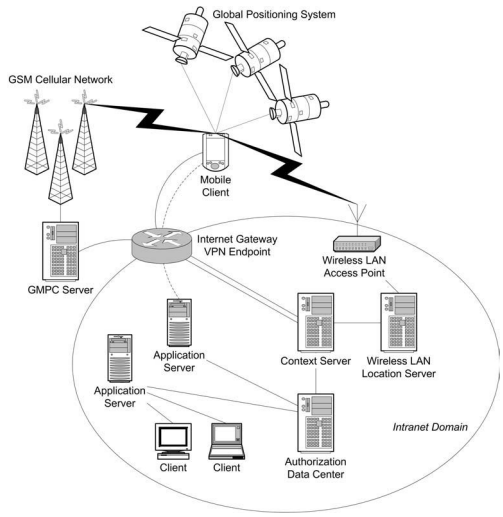

Figure 3. System Architecture

The authorization and dynamic context service management services were implemented using Java 1.4.2. The Kerberos implementation in the architecture components uses Java Authentication and Authorization Services (JAAS) for Kerberos authentication and key establishment between architecture entities. The Java security architecture was customized, such that a new Security Manager and Policy class were implemented. The new classes facilitate the initialization of a secure Kerberos context for the standard communications channel and update channel with the authorization server, subsequently allowing XML messages to be sent and received in these channels. This approach allows standard Java Security methods such as doAsPrivileged(), checkPermission() and standard access control context constructs to be used, resulting in transparent use of the authorization architecture. An additional listener class was implemented, such that event listener methods would be instantiated on events such as "Access Control Context Changed".

A benchmark of the architecture performance was conducted to quantify the efficiency of the architecture. The performance results detailed in Table 1 are average execution times for 1000 consecutive executions of checkPermission() for each application service running. Dynamic context updates were continually sent to the DCSM for a principal with an established context in each application service. The benchmark was conducted with 1, 5, and 10 simultaneous application services, requesting an access control decision continuously, simulating high load. The dynamic context updates were benchmarked with a load of 10, 100 and 1000 active principals over 1, 5, and 10 application services.

checkPermission()		Dynamic Context Updates		
No. of App Svrs	**Average Time (ms)**	**10 Principals (ms)**	**100 Principals (ms)**	**1000 Principals (ms)**
1	24.209	328.55	448.55	1079.55
5	53.330	341.61	461.61	1092.61
10	97.341	571.13	691.13	1322.13

Table 1. Performance Results

As can be seen from the results, the architecture appears to scale well with better than linear growth for context updates as the number of active principals increases. The authorization service and DCSM are designed such that they can be distributed to improve performance for large domains.

4.1. Architecture Usage Scenario

A prototype fileserver was developed to illustrate possible uses of the architecture. The fileserver is based on Samba 3.0, which supports Kerberos authentication. The following scenario illustrates the context-aware architecture in the use of a fileserver that supports the requirements of "commercial in confidence" file access.

There are three DCS' that are used in this scenario:

1. **GSM Location DCS:** This DCS acquires a trusted location from a GSM cell phone using the methods we have developed in [9]. The DCS authenticates the user and confirms the user is associated with the cell phone.

2. **iButton[6] Location DCS:** This DCS provides the location of a user associated with an authenticated iButton based on the known location of a given iButton reader.

3. **Kerberos Connection Security DCS:** This DCS is a component of the fileserver which communicates the properties of the authentication and subsequent communications to the DCSM. The properties supported by the DCS are `hasMutualAuth`, `hasIntegrity`, and `hasPrivacy`.

Commercial in confidence projects have a project manager and a series of consultants who occupy the roles `project1_manager` and `project1_consultant`, which are triggered by principal activation. The fileserver has a shared area for each project and a personal area for each consultant. Consultants are only permitted to work at the office or home where appropriate security arrangements exist. The home location of a consultant is registered in the directory.

The `project1_consultant` role has a number of tasks relating to file and printer access, one of which is `project1_filesystem_access`. This task contains permissions for accessing the project share and the consultant's personal share. The task has triggers of the dcaa.dcsm.Location context class and activation of the `project1_consultant` role. In order to ensure the shares are only accessed in approved locations and with appropriate connection security, the task contains the ContextRule illustrated in Figure 2.

A user wanting to work on "Project1" in the office would tap their iButton at the office door on entry and exit, causing a ContextUpdate with a dcaa.dcsm.Location object to be sent to the DCSM. When the user attempts to access the share, a Kerberos ticket is presented to the fileserver. The fileserver initiates an access control context with the authorization service and requests the role of `project1_consultant`. Assuming the user is a member of the role and no constraints prohibit the activation of the role, the role will be activated. As activation of the role triggers evaluation of the `project1_filesystem_access` task, it too will be activated assuming its constraints are fulfilled.

A remote user would have to prove their location via the GSM location DCS. If a user is to leave a trusted location area, the task will be deactivated following the appropriate DCS notifications. Similarly with the fileserver ConnectionSecurity DCS, it notifies the DCSM of the established Kerberos context security properties.

[6] iButton® is a tamper-resistant token that has a unique identifier and may provide support for cryptography. `http://www.ibutton.com`

5. Conclusion

In conclusion, this paper has introduced a new authorization architecture for Intranet environments that supports context-aware authorization using both local and remote security contexts. We propose extensions to RBAC that facilitate efficient context-aware authorization with simple policies and administration. The implementation of the architecture has been described with the currently implemented dynamic context services as well as the description of a demonstration application that utilizes the architecture. Future work in this architecture will involve investigating cross domain context-aware authorization.

References

[1] J. Al-Muhtadi, M. Anand, M. D. Mickunas, and R. H. Campbell. Secure smart homes using jini and uiuc sesame. Uiucdcs-r-99-2142, University of Illinois at Urbana Champaign, December 1999.

[2] J. Al-Muhtadi, A. Ranganathan, R. Campbell, and M. D. Mickunas. Cerberus: A context-aware security scheme for smart spaces, March 2003.

[3] M. Covington, M. Moyer, and M. Ahamad. Generalized role-based access control for securing future applications, 2000.

[4] M. J. Covington, P. Fogla, Z. Zhan, and M. Ahamad. A context-aware security architecture for emerging applications. In *Proceeding of the Annual Computer Security Applications Conference (ACSAC)*, December 2002.

[5] M. J. Covington, W. Long, S. Srinivasan, A. Dey, M. Ahamad, and G. Abowd. Security context-aware applications using environment roles. In *Proceedings of the 6th ACM Symposium on Access Control Models and Technologies (SACMAT '01)*, May 2001.

[6] A. Harbitter and D. A. Menasce. The performance of public key-enables kerberos authentication in mobile computing applications. In *ACM CCS'01*, November 2001.

[7] J. Kohl and C. Neuman. *The Kerberos Network Authentication Service (V5)*. Networking Working Group Request for Comments, September 1993.

[8] B. Tung, C. Neuman, M. Hur, A. Medvinsky, S. Medvinsky, J. Wray, and J. Trostle. *Public Key Cryptography for Initial Authentication in Kerberos*. Kerberos WG Working Group of the IETF, 16 edition, September 2002.

[9] C. Wullems, M. Looi, and A. Clark. Enhancing the security of internet applications using location: A new model for tamper-resistant gsm location. In *Proceedings of the Eighth IEEE Symposium on Computers and Communications (ISCC 2003)*, volume 1, pages 1251–1258, July 2003.