A Context-Aware Security Architecture for Emerging Applications*

Michael J. Covington†, Prahlad Fogla, Zhiyuan Zhan, Mustaque Ahamad
College of Computing
Georgia Institute of Technology
Atlanta, Georgia 30332-0280

Abstract

We describe an approach to building security services for context-aware environments. Specifically, we focus on the design of security services that incorporate the use of security-relevant "context" to provide flexible access control and policy enforcement. We previously presented a generalized access control model that makes significant use of contextual information in policy definition. This document provides a concrete realization of such a model by presenting a system-level service architecture, as well as early implementation experience with the framework. Through our context-aware security services, our system architecture offers enhanced authentication services, more flexible access control and a security subsystem that can adapt itself based on current conditions in the environment. We discuss our architecture and implementation and show how it can be used to secure several sample applications.

1 Introduction

As computers become more pervasive and their functionality is more *transparently* integrated into homes and communities, new applications will emerge to make everyday living easier for people. Such applications, which will be enabled by a pervasive computing and communication infrastructure, will provide unobtrusive access to important information, resources and services. Clearly, the successful deployment of such applications will depend on our ability to secure them. In particular, we will have to ensure that access to information and services is granted only to authorized users, without requiring them to deal with complex security policies or burdensome authentication procedures.

Traditionally, security requirements are assumed to be relatively static since access control decisions do not change with context, nor do they account for changing conditions in the environment. As computing technology becomes more tightly integrated into the fabric of everyday life, it is imperative that security mechanisms become more flexible and less intrusive. To address these concerns, our research is focused on providing security services for context-aware computing environments that can adapt to changing conditions when requests are made. Specifically, we target an "aware" or smart home setting, such as the Aware Home [12], in which a pervasive computing environment is used to provide augmented services to the residents and their guests in the home. Sensors are used to capture, process and store a variety of information about the users, their activities and the environment in which they interact. Access to certain appliances may be controlled based on the context of the request; for example, the time at which the request is made or the activity in which the user is involved.

Although considerable work has been done in securing military and commercial information systems, few projects have specifically addressed the needs of a residential computing infrastructure. Current research that specifically targets home environments attempts to move traditional security mechanisms into the residential space. In contrast, we are developing security techniques that are natural, intuitive and non-intrusive for connected homes and community environments. In this paper, we present a middleware-level architecture that has been designed to secure and support context-aware applications in the Aware Home using the authentication and authorization techniques that we presented in [6, 17, 5]. Such an architecture We will show how such applications have special requirements and properties that require a generalized security architecture — one not found in traditional systems — to provide the necessary support for securing pervasive computing applications.

The remainder of this paper proceeds as follows: Section 2 motivates our work by detailing the security challenges that are encountered in a context-aware environment such as the Aware Home. We show how extended access control models are beneficial in context-aware environments and discuss some of the unique features that distinguish these application scenarios from those found in tradi-

*This work was supported in part by NSF grants CCR-9988212, ITR-0081276 and ITR-0121634.

†Contact Author: Michael.Covington@cc.gatech.edu

tional systems. In Sect. 3, we present an implementation of an architecture for securing context-aware applications and discuss our initial experiences with the security services. Section 4 details our experience with building secure applications and provides experimental results that characterize the performance of the services that make up the security architecture. We discuss the benefits of our approach and compare it with related work in Sect. 5. Finally, the paper is concluded in Sect. 6.

2 Security Challenges in Context-Aware Environments

The context-aware applications alluded to in the previous section present new and interesting security challenges. The transparent nature of a pervasive computing environment motivates the need for a security architecture that will transparently determine the sources of requests and handle a high degree of context changes. We can no longer assume that user "sessions" will persist for extended periods with the same authentication and authorization credentials. Thus, we must look at new access control models and authentication techniques that will operate effectively in these next-generation environments.

2.1 Context-Aware Authorization

Security policies in an information-rich environment like the Aware Home can potentially be quite complex. A policy can restrict access to information or resources based on several factors, including attributes pertaining to the subject, the resource or the environment. For example, subjects can be classified into roles such as "resident" or "guest." Access rights can depend on the subject's classification (e.g., "resident"), as well as on his or her actual identity. Access also may be restricted based on the subject's location, or based on environmental factors such as the temperature or the time of day.

While time and location are natural examples of environmental states that could be used in access control, richer contextual information could also impact the result of an access request. We take an approach which makes use of the well-known notion of roles to capture security-relevant state. In particular, we define environment roles based on the security-relevant context or state of the environment [5]. Environment roles are one component in the Generalized Role-Based Access Control (GRBAC) model [6, 17] and a key extension to the core ideas found in traditional Role-Based Access Control [20, 9].

While GRBAC presents a powerful and flexible model for expressing access control policies for context-aware applications, the model itself clearly requires a more complex system architecture to support the extended roles and intricate policies that are made possible. In particular, the system architecture must support mechanisms to securely collect contextual information that is used to enforce access control policies. In addition, GRBAC requires that a separate authorization component be available to bind subject, object and environment roles together with an operation and corresponding permission. In Sect. 3, we present our approach to building this architecture and discuss the various components that provide the security infrastructure to context-aware applications.

2.2 Non-Intrusive User Authentication

Another challenge presented by a pervasive computing environment like the Aware Home involves relieving the user from the "burden" of authentication. Ideally, information available from sensors in the home should be used to automatically infer a subject's security-relevant attributes (e.g., identity, role, location, etc.). Although it is possible for a resident to use a physical authentication token, it is undesirable to expect the user to carry such an object around at all times. Previous work with physical identification tokens such as the Active Badge system [22] have yielded useful results but are less practical for home use and unreliable for authentication (users can assume a different identity by simply carrying another person's badge).

Several sensor-based technologies such as voice and face recognition can be deployed in the Aware Home to non-intrusively identify humans and track their movements. Many such techniques can establish the identity of a subject with only a partial level of certainty. Such "partial authentication" has important implications for access control models. In particular, some identification mechanisms are known to be more reliable than others. Our model for context-aware user authentication takes these differences into account and provides a mechanism for "parameterized authentication." Parameterized authentication allows a legitimate user to maintain access to a system even when the overall quality of his authentication is not 100%. We do this by granting subsets of access rights based on the current "authentication parameter," a metric that is based on trust in the devices that provide authentication data and the inherent accuracy of those devices. A related notion of various levels of authentication for a user was recently proposed in [10].

3 Implementation

To address the problem of providing security to context-aware applications, we describe a Context-Aware Security Architecture (CASA). CASA provides a security infrastructure upon which emerging applications can be built. Fig-

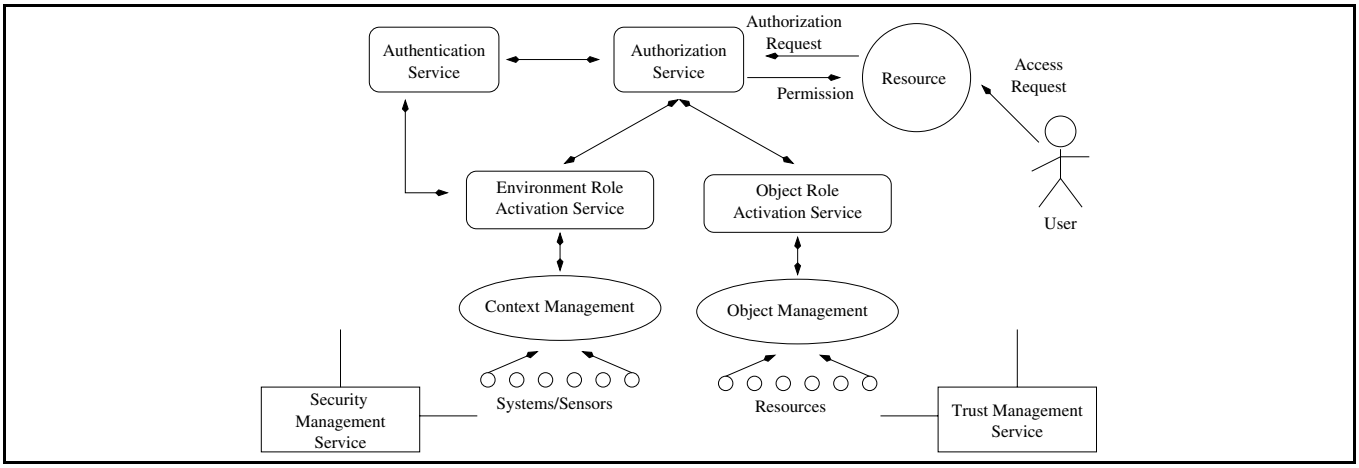

Figure 1. Logical Framework for Securing Context-Aware Applications

ure 1 provides a high-level overview of the various logical components that comprise this security architecture.

We have implemented a prototype system that uses CASA to provide security services to applications running in an information-rich computing environment. Our implementation is built using the Java 2 Standard Edition Software Development Kit (J2SE SDK)[16]. In the following sections, we describe our instantiation of CASA and provide details on making these services available to context-aware applications

3.1 Policy Specification Language

A Generalized Policy Definition Language (GPDL) is described in [17] for defining GRBAC policies. We have found that in practice, however, GPDL can be frustrating and clumsy for a policy administrator to manage, especially when editing large, complex policy files. Roles are inherently visual, so it would be useful to have a graphical policy editor that displays available roles, their relationships, and policy rules in a clear-cut manner.

We have built a prototype graphical editor and are currently exploring how it can help to define and explain complex security policies. Such an interface is necessary as our access model is deployed in the Aware Home. By using this graphical editor, we are able to display complex security policies using simple constructs and an intuitive layout. This GUI allows a security administrator to associate permissions with various combinations of roles. For example, a *child* can be denied access to a category of resources that is classified using a single role, *dangerous appliance*, during certain environmental conditions (e.g., during a parent's *working hours*). In a more intricate example, we could specify complex, method-level access control on object resources in the system. For instance, the family physician may have access to read and write medical databases in the home, while the family attorney can only obtain a limited view of such records in the event of an emergency.

In our implementation, policies are defined through the graphical management tool and encoded into eXtensible Markup Language, or XML [4]. XML is used to specify access policies, role definitions and relationships and is also used as a common representation to share data between the various services in the architecture. Figure 2 shows the XML-encoded policy that restricts a user in the *child* role from accessing a *dangerous appliance* during specified environmental conditions.

Given well-structured rulesets, XML provides an efficient structure for storing the policy that is generated and enforced by our security services. We take advantage of XML's robust, non-proprietary and verifiable file-format by using it to transmit policies and related information (e.g., environment role definitions) between services in our architecture. Each component in our infrastructure can construct its own efficient runtime structures for local processing, but XML is used to transfer information between services. This allows us to standardize on a platform-independent policy specification that can be read, verified and processed by any authorized component.

3.2 Security Management Service

The Security Management Service (SMS) is responsible for managing and storing system policies and role relationships, as specified by the security administrator. We separate the functionality of policy storage and runtime policy evaluation so as to allow for a more distributed and efficient system design. Policies that are enforced using the GRBAC model are defined in terms of roles. The SMS manages the relationships that exist between roles and provides appropriate mechanisms for secure storage and retrieval of policies.

```
<GRBAC_TABLES>
       <POLICY>
           <SROLE> Child </SROLE>
           <OROLE> Dangerous Appliance </OROLE>
           <ACTION> ALL </ACTION>
           <EROLE> Working Hours </EROLE>
           <PERMS> Deny </PERMS>
       </POLICY>
</GRBAC_TABLES>
```

Figure 2. Example GRBAC Policy Specification in XML

In our architecture, policy enforcement is provided by an Authorization Service, while environment role activation is managed by a separate logical service. The SMS allows for role manipulation to be performed and ensures that any associated roles or policies are updated accordingly. For example, if a security administrator were to make policy changes to an active system, the SMS would notify any role activation service (subject, object, environment) affected by the change.

In addition, the SMS provides a central location for backup and recovery. Since it is responsible for all policy management and for the bootstrapping of other components, the SMS can be distributed and protected more efficiently and effectively than an architecture that would distribute policy and management functionalities. For instance, the SMS could be implemented on top of a distributed data repository such as the Secure Store [14] that would provide data replication and enhanced availability in a potentially hostile environment.

Our implementation provides an SMS that is comprised of two separate components – a persistent storage mechanism that is responsible for storing policy-related data and a front-end processor that provides a communication interface between this storage component and other services in the framework. This modular design allows storage mechanisms to be easily replaced without requiring changes to the communication interface.

Although the SMS provides a centralized location for policy storage and retrieval, no other components are structured in the same way. Authorization, authentication and role-activation services can be centralized or distributed based on the environment in which they operate. In the Aware Home, some resources may instantiate a local Authorization Service to perform their own resource access checks. Other limited-capability resources may opt to offload authorization to a central service. In either case, the SMS serves as the central console for policy updates and ensures consistency and well-formedness of policy and role definitions.

3.3 Authorization Service

The Authorization Service is responsible for retrieving policy definitions from the SMS and for determining whether a particular request should be granted or denied. Our implementation supports an Authorization Service that can boot in two different modes. The first mode does not store any authorization information locally. For every request, the Authorization Service must contact the SMS to retrieve a copy of the relevant policy. In the second mode, the Authorization Service stores all relevant policy definitions in a local runtime structure. The second mode provides greatly enhanced performance but also requires more resources at the Authorization Service.

Since our system allows for multiple Authorization Services, it is useful for each Authorization Service to maintain a cached list of active environment roles (ERoles) that have already been evaluated. By assuming synchronized clocks between the various services in our architecture, we define a lifetime parameter that can be used to determine the lifetime of cached ERole status. In our implementation, the Authorization Service caches only active ERoles.

When an access request is received by the Authorization Service, it first determines what roles are active for the object being accessed. Based on this active object role set, policies are checked and the Authorization Service must determine if it requires information from the Authentication Service or ERAS. If so, appropriate calls are made to determine active subject roles and/or environment roles that are relevant to the access request. Policies are stored in the form of a tuple:

```
<SRole, ORole, Action, ERole, Perms>
```

where *SRole* specifies a subject role, *ORole* specifies an object role, *Action* specifies an operation, *ERole* specifies environment role(s) and *Perms* determine whether the action is granted or denied.

3.4 Environment Role Activation Service

The Environment Role Activation Service (ERAS) maintains information on system state and manages role activation and deactivation based on conditions that are held by

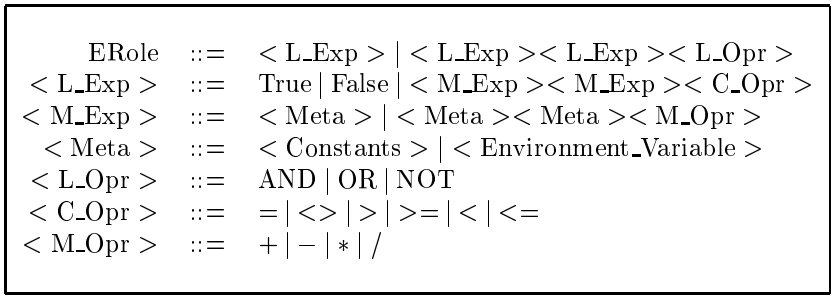

Figure 3. Environment Role Definition

specified environment variables. It is through this component that an administrator specifies the variables that define environment roles and the conditions that must be met in order for the roles to be activated. This service interacts with one or more Context Management Service/sensors to ensure that system state is collected and the appropriate roles are activated when necessary.

Environment roles (ERoles) specify and capture environmental conditions that are relevant to access control. The Environment Role Activation Service (ERAS) is responsible for evaluating the status of ERoles. Environment roles are activated when certain environmental conditions are met. These conditions can change dynamically and hence the active role set is also subject to change. In our system, we have implemented an ERAS that evaluates role status on-demand. Our model maintains two possible states for each ERole, *active* and *inactive*, and we represent these with boolean values. We have chosen to implement ERoles as logical expressions. The major function of the ERAS is to evaluate these logical expressions and determine the state of any given ERole.

In a previous implementation, we used Conjunctive Normal Form (CNF) to represent ERoles (the logical expressions) but found that this approach made it difficult to specify additional operations on environment variables. We have since modified our implementation to one that utilizes a postfix format and allows all mathematical and logical operations (detailed below) to be treated in a unified fashion In terms of our implementation, an ERole is defined as shown in Figure 3.

Figure 4 illustrates a formal definition for the *Party* ERole (equivalent to "$Person\_LivingRoom >= 10$ and $NoiseLevel\_LivingRoom\_DB >= 50$"). ERoles can also inherit definitions from existing ERoles. For example, if ERoles *Party* and *Weekend* have already been defined in the system, a new ERole *Weekend_Party* can be defined simply through logical "AND" of the logical expressions from both of the existing ERoles. The newly created ERole inherits definitions from both of its parents and can also specify additional conditions that must hold in order for the role to be activated. Further discussion of the environment role model and related properties can be found in [5].

To overcome the costs associated with external communication with both the SMS and sensors, we have implemented an ERAS that uses a caching mechanism to maintain the status of an evaluated ERole. Since environmental variables are subject to dynamic fluctuations in value, it is not appropriate to cache such variables. However, once an ERole is evaluated, it may be possible to cache its status for future use. Our implementation assigns *lifetime* and *inactive_lifetime* attributes to each ERole. In other words, ERoles can be cached regardless of status. In either case, a zero lifetime indicates that the ERole must be evaluated each time it is used. In addition, both lifetime values are static. We are currently investigating other methods that would allow us to derive a more dynamic lifetime value for ERole caching.

With caching enabled, the ERAS is implemented as follows: a request from the Authorization Service will trigger the ERAS to evaluate a list of ERoles. The ERAS will first check whether the ERole requested is already in the cache. If so, it will then determine the "freshness" of the cached copy. If the lifetime is still valid, the cached status is sent to the Authorization Service along with a new expiration time. If a cached copy is not found, or if it is too old, the ERAS will then contact the appropriate sensor(s) for current state values which will be used to (re-)evaluate the ERole. A request timestamp is associated with each (re-)evaluated ERole when they are placed into the cache.

3.5 Authentication Service

We have developed algorithms that allow for the secure and transparent authentication of users in context-aware environments. By making use of security-specific measurements in our calculations, we are able to take into account the likelihood of a sensor device being compromised, as well as its inherent ability to accurately identify users in the system. Such information is used to determine the level of authentication that can be provided to the source of a request.

We have defined our API for the authentication service

$$
\begin{array}{rcl}
\text{Party} & ::= & \text{L\_Exp}_1\ \text{L\_Exp}_2\ \text{AND} \\
\text{L\_Exp}_1 & ::= & \text{M\_Exp}_{11}\ 10\ >= \\
\text{L\_Exp}_2 & ::= & \text{M\_Exp}_{21}\ 50\ >= \\
\text{M\_Exp}_{11} & ::= & \text{Person\_LivingRoom} \\
\text{M\_Exp}_{21} & ::= & \text{NoiseLevel\_LivingRoom\_DB}
\end{array}
$$

Figure 4. Example Environment Role Definition: *Party*

and are currently building the components that will implement our algorithms and expose the API to other services in our framework. In the mean time, the CASA architecture provides an authentication service that can both verify credentials and retrieve them from the environment.

The authentication service for CASA supports two models of authentication: push and pull. The push model is similar to the traditional model where a user sends credentials to the system at the time access is requested. The pull model, on the other hand, only requires authentication to be performed when necessary. In the above example, the Authorization Service may determine that any user in the home may access the family newspaper subscription and, therefore, no authentication needs to be performed. However, if the Authorization Service were to receive a credential-less request to access the medical database, it could dynamically pull authentication information from the environment based on the source of the request.

The CASA authentication service expects requests for either credential verification or credential retrieval. In verification, credentials are passed to the authentication service where their status (validity, expiration, etc.) is checked and a user ID is returned, which can be used to activate subject roles. For credential retrieval, the authentication service is sent information pertaining to the source of the access request and a least-acceptable authentication parameter. The authentication parameter details the level or quality of authentication that is required by the Authorization Service.

Details of the algorithms that comprise the authentication service are beyond the scope of this document. Additional details regarding the authentication service and its interaction with the Authorization Service are provided in Sect. 4.

3.6 Context Management Services

To facilitate the collection of environment variables and their associated values, we make use of Context Management Services (CMS). The ERAS monitors one or more such services to maintain a snapshot of current environmental conditions. Example CMSs include services based on the Simple Network Management Protocol (SNMP), the Context Toolkit (CTTK) [8] and similar services that monitor environmental conditions (e.g., system, network, etc.) and maintain a record of environment state.

Sensors are placed throughout the environment to collect useful security-relevant data. Sensors can include motion detectors, fingerprint scanners, cameras and numerous other sensing devices. In addition to authentication-related data, sensors can also collect information related to environmental state, such as temperature, ambient noise in a room, network bandwidth and CPU usage. Sensors communicate directly with one or more CMSs that are responsible for managing the received data.

Our current implementation makes use of the Context Toolkit [8, 5] and a collection of distributed sensors for the purpose of managing environmental context. The Context Toolkit is a software infrastructure that provides useful abstractions for collecting and organizing environmental state information; it allows for the seamless incorporation of sensed context into "aware" applications. The overall organization of the software is shown in Fig. 5. Our implementation of CASA makes several important changes to the Context Toolkit to ensure that security-relevant environmental context is collected in a secure fashion. This section provides details on those changes and the Context Toolkit in general.

In the CTTK, *Context widgets* represent abstractions over sensors that hide details of how sensing and interpretation of the environment occurs. Widgets are essentially wrappers around an underlying sensor or service; they provide an interface to automatically deliver information to interested components or services in the system. *Aggregators* collect information for relevant entities of an application. In the home, there could be aggregators for rooms in the house (Room Aggregators) and residents of the household (Person Aggregators). Finally, *Interpreters* are responsible for abstracting low-level context to higher-level information. An interpreter can convert state information to another format or meaning. For example, a complex interpreter can take location, identity and sound information and subsequently determine that a meeting is underway.

We have built a secure version of the CTTK that will allow us to collect environment information in a manner that is both secure and reliable. It is reasonable for us to assume that the individual components of the toolkit are se-

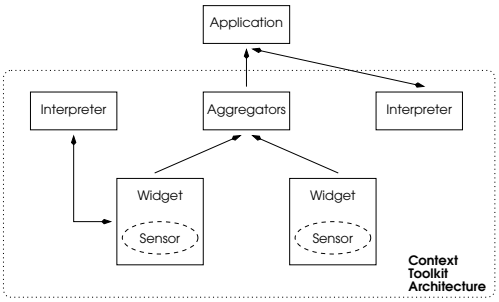

Figure 5. The Context Toolkit

cure as stand-alone services. For example, we assume that sensors and widgets are securely bound together in such a way that information from a sensor (e.g., an RF transmitter) can be securely transmitted to its associated widget. However, the same may not be true for communication links between components.

We have built a certificate-based PKI to support component authentication and data encryption. Specifically, we use HTTP over SSL to allow for authenticated and encrypted sessions. Given the component-based nature of the Context Toolkit, this approach allows us the flexibility to select either secure or insecure channels, depending on the properties of the communication that is taking place. We are currently expanding our implementation to support multiple communication protocols (in addition to HTTP) and a more flexible key-sharing infrastructure (such as SPKI).

4 Securing Applications with CASA

Our context-aware security architecture – through the incorporation of GRBAC and our enhanced authentication techniques – presents a powerful approach for enforcing security policies in a ubiquitous computing environment. This section shows how our security infrastructure can be applied, in practice, to the home environment. It also demonstrates some of the additional benefits not offered by traditional approaches to system security. We present an operational scenario that requires an application to leverage the security services of CASA. In addition, we provide a performance analysis of the architecture.

Using our graphical Management Tool, we begin by creating a simple environment role hierarchy. In our example, we define a series of basic time-related environment roles that capture different times in an academic calendar. We also specify the CMS/sensors that will be used to collect the required information to determine the ERole status. Finally, we provide subject role and object role information that is subsequently used to form a policy statement. When complete, our role definitions (and the relationships that exist between them) and policy statements are encoded into an XML format similar to that in Fig 6.

Using this access control policy we have run several experiments to show the performance of CASA when processing authorization requests from an application. The four primary components (SMS, ERAS, Authentication and Authorization) were started as separate services on distributed machines. The experiments were conducted on a cluster of workstations using dual-2.20GHz Intel Xeon processors, running RedHat Linux 7.2, all connected by a 100 Mb Ethernet switch. The Java virtual machine was part of the J2SE SDK version 1.4 from Javasoft.

For our first set of experiments, we generated a series of access requests using different sets of active ERoles. There were a total of eleven access requests sent to the Authorization Service. The first request involved a policy that granted access regardless of environmental state; no ERoles needed to be active and, therefore, no check with the ERAS was necessary. All other requests in the series involved access checks that made use of more complex policies; each policy specified an environment role that used from one to ten unique variables (sensors) in its definition. In addition, authentication services were provided to verify credentials that were "pushed" with the access requests.

This first set of experiments was run using HTTP as the transport protocol and allowed us to generate an initial set of measurements to demonstrate the efficiency of our implementation. Table 1 shows "round-trip time" for the experiment, starting with the time that the access request is generated and ending with receipt of an access response from the Authorization Service.

To illustrate the performance increase that was observed through the use of caching mechanisms in both the ERAS and the Authorization Service, the same set of experiments was run, first with only an ERAS cache and secondly with a fresh and fully-populated cache at the Authentication Service. As demonstrated here, the CASA implementation can provide access requests in a range of 1 to 40 milliseconds, depending on cache state and contents. Clearly, when the number of ERoles requiring evaluation increases, the time spent evaluating ERole status also increases. This applies

```
<GRBAC_TABLES>
       <EROLE_DESC>
              <NAME> Academic Year </NAME>
              <L_EXP> ROOT-EROLE </DC_NAME
       </EROLE>
       <EROLE_DESC>
              <NAME> Spring Semester </NAME>
              <L_EXP> Spring-Semester </DC_NAME>
       </EROLE>
       ...
       <POLICY>
              <SROLE> Teaching Assistant </SROLE>
              <OROLE> Classroom Computer </OROLE>
              <ACTION> ALL </ACTION>
              <EROLE> Spring Semester </EROLE>
              <PERMS> GRANT </PERMS>
       </POLICY>
</GRBAC_TABLES>
```

Figure 6. Example XML encoding of Role Definitions and Policy Statements

ERole Defs	No Cache	ERAS Cache	AS Cache
No ERoles	25	9	9
1 Variable	26	10	6
2 Variables	29	10	2
3 Variables	32	12	5
4 Variables	30	13	6
5 Variables	35	14	7
6 Variables	36	13	7
7 Variables	39	13	6
8 Variables	57	15	6
9 Variables	37	18	5
10 Variables	38	16	5

Table 1. Request over HTTP, in milliseconds

ERole Defs	No Cache	ERAS Cache	AS Cache
No ERoles	775	536	212
1 Variable	885	485	221
2 Variables	935	490	230
3 Variables	1181	438	206
4 Variables	1610	462	215
5 Variables	1946	438	217
6 Variables	1737	500	275
7 Variables	1925	426	243
8 Variables	2173	436	227
9 Variables	2417	525	213
10 Variables	2450	415	193

Table 2. Request over HTTPS, in milliseconds

to both uncached and cached values. However, we feel that our implementation is highly efficient in determining ERole status – the majority of time spent evaluating ERole status is actually spent in communicating with the environmental sensors that monitor variable conditions.

In order to ensure that policies are protected and that credentials are not observed during transmission, we also support encrypted communication channels in the CASA implementation. We have performed a second set of experiments in which we generate the same access requests as before but use an SSL-encrypted HTTP channel over which the services communicate. As before, there were a total of eleven access requests sent to the Authorization Service. Table 2 shows the results of our application test.

The results presented here were all obtained using SSL-encrypted HTTP communication channels and, when compared to the previous results, demonstrate the cost of establishing and using secure communications. This protocol selection resulted in a delay that was caused by SSL-related key exchange and session establishment. The overhead introduced by the processing at the services is not significant when compared to the connection establishment and communication times. We are currently working on a communication subsystem that utilizes session key-reuse; we expect this to eliminate a significant portion of the overhead cost associated with the HTTPS-based communication.

The experimental results show that services in our architecture can authorize requests in a reasonable amount of time. Furthermore, by taking advantage of our caching framework and other runtime efficiencies, such as leaving established (secure) connections open between requests, context-aware security does not present any significant overhead to the application.

In future work we will complete our analysis of the overall architecture and are currently building a secure implementation of a smart intercom application that will provide a foundation for our experiments. Not only will we focus on comparing an insecure, context-aware application with a secured version, but analyzing the effectiveness of the interfaces that are exposed to residents in the Aware Home as well.

5 Related Work

Despite considerable interest and research in pervasive computing [23, 19], security concerns in such environments have received little attention. In this section, we briefly highlight several existing projects and technologies that have influenced our work with providing security services in context-aware environments.

There are a number of well-known architectures that have been used to build secure systems in the past. The Kerberos system [13] implements a protocol that can be used to efficiently authenticate entities in a distributed system. Satyanarayanan [21] discusses the architectural issues surrounding the security services that are incorporated into the Andrew distributed computing environment. In addition, significant work has been focused on the modeling, building and analysis of secure Public Key Infrastructures (PKIs). Other issues related to building secure distributed systems, including subject role authentication and delegation are addressed by Lampson et. al. [15].

In addition to these architectures, other security infrastructures have been presented to specifically address the needs of authorization logic. In [2], an architecture is proposed for securing distributed document management system. The authors address the need for an access control logic in a complex distributed system. In particular, they discuss a logic that supports linked local name spaces and the management of a large system spread across administrative domains.

Al-Muhtadi et al. [1] present an approach to transfer traditional security solutions—specifically the Kerberos extension SESAME—into "smart spaces" such as the home. Their work focuses on the integration issues that must be addressed when placing computing-intensive security mechanisms into devices with limited resources. While a GRBAC-like component could conceivably be added to their solution, our architecture is appropriate in the context-aware environments that we target. The architecture that we propose can support partial authentication, context-aware access control and dynamic policy generation based on roles – elements not found in traditional computing environments.

We have also discussed traditional Role-based Access Control (RBAC) [20, 9] and acknowledge the tremendous influence it has had on our research efforts. Our work expands the RBAC model by providing a more versatile and more expressive framework that incorporates the use of environment and object roles. By using the uniform notion of a role to capture user, environmental and resource attributes, our model allows for the definition of context-aware security policies. In addition, roles make it easy to define and understand complex security policies.

The use of XML in security policy definition has been studied in numerous contexts, though work in this area has focused primarily on the development of access control models and methods to address encryption in the language. Some of the more relevant work in XML security includes [7] in which Damiani et al. present an access control model that uses an XML-based approach to define and enforce access restrictions directly on the structure and content of Web documents. In [3], Bertino et al. present an XML-compliant formalism for specifying security-related information for Web document protection. In addition, Herzberg et al. present a Trust Policy Language (TPL) [11] that defines policies using a well-formed XML document. Finally, Netegrity [18] has presented S2ML, a Security Services Markup Language that provides a mechanism for describing existing security models using XML syntax.

6 Conclusion

In this paper we have introduced a new model for securing context-aware environments and describe why we believe it will be useful for securing applications in the highly-connected world of tomorrow. Much of this work is focused on providing more adaptive security services than those found in traditional computing environments.

We have presented a framework for providing authorization services in context-aware environments and applications. This framework supports the collection of contextual information from resources, the environment and the users who interact in that environment. In addition, we have explored, through the context aware security architecture, an implementation of the Generalized Role-Based Access Control model. We have discussed our initial experience with the security framework and provided experimental results regarding its effectiveness in securing context-aware applications.

References

[1] Jalal Al-Muhtadi, Manish Anand, M. Denis Mickunas, and Roy H. Campbell. Secure smart homes using Jini and UIUC SESAME. In *Proceedings of the Annual Computer Security Applications Conference (ACSAC)*, December 2000.

[2] Dirk Balfanz, Drew Dean, and Mike Spreitzer. A security infrastructure for distributed java applications. In *Proceedings of the IEEE Symposium on Security and Privacy*, pages 15–26, 2000.

[3] Elisa Bertino, Silvana Castano, and Elena Ferrari. On specifying security policies for web documents with an XML-based language. In *Proceedings of the 6th ACM Symposium on Access Control Models and Technologies*, pages 57–74, Chantilly, Virginia, USA, May 2001.

[4] World-Wide Web Consortium. eXtensible Markup Language (XML). W3C Specifications. http://www.w3.org/TR/WD-xml-lang.html.

[5] Michael J. Covington, Wende Long, Srividhya Srinivasan, Anind Dey, Mustaque Ahamad, and Gregory Abowd. Securing context-aware applications using environment roles. In *Proceedings of the 6th ACM Symposium on Access Control Models and Technologies*, pages 10–20, Chantilly, Virginia, USA, May 2001.

[6] Michael J. Covington, Matthew J. Moyer, and Mustaque Ahamad. Generalized role-based access control for securing future applications. In *Proceedings of the 23rd National Information Systems Security Conference (NISSC)*, pages 40–51, Baltimore, Maryland, USA, October 2000.

[7] Ernesto Damiani, Sabrina De Capitani di Vimercati, Stefano Paraboschi, and Pierangela Samarati. A fine-grained access control system for XML documents. *ACM Transactions on Information and System Security*, 5(2):169–202, May 2002.

[8] Anind K. Dey, Daniel Salber, and Gregory D. Abowd. A context-based infrastructure for smart environments. In *Proceedings of the 1st International Workshop on Managing Interactions in Smart Environments (MANSE '99), Dublin, Ireland*, pages 114–128, December 1999.

[9] David F. Ferraiolo, John F. Barkley, and D. Richard Kuhn. A role based access control model and reference implementation within a corporate intranet. In *ACM Transactions on Information and System Security*, volume 1, February 1999.

[10] Gregory R. Ganger. Position summary: Authentication confidences. In *Proceedings of the IEEE Workshop on Hot Topics in Operating Systems (HotOS)*, 2001.

[11] Amir Herzberg, Yosi Mass, and Joris Mihaeli. Access control meets public key infrastructure, or: Assigning roles to strangers. In *Proceedings of the IEEE Symposium on Security and Privacy*, pages 2–14, 2000.

[12] Georgia Tech Broadband Institute. The Aware Home Research Initiative. Research Initiative Web Page, 2002. http://www.cc.gatech.edu/fce/ahri/.

[13] John T. Kohl, B. Clifford Neuman, and Theodore Y. T'so. The evolution of the kerberos authentication system. *Distributed Open Systems (IEEE Computer Society Press)*, 1994.

[14] Subramanian Lakshmanan, Mustaque Ahamad, and H. Venkateswaran. A secure and highly available distributed store for meeting diverse data storage needs. In *International Conference on Dependable Systems and Networks (DSN)*, July 2001.

[15] Butler Lampson, Martin Abadi, and Michael Burrows. Authentication in distributed systems: Theory and practice. *ACM Transactions on Computer Systems*, 1992.

[16] Sun Microsystems. Java 2 standard edition. Software Development Kit. http://java.sun.com.

[17] Matthew J. Moyer and Mustaque Ahamad. Generalized role based access control. In *Proceedings of the IEEE International Conference on Distributed Computing Systems (ICDCS)*, Mesa, Arizona, USA, April 2001.

[18] Netegrity. S2ml: The XML standard for describing and sharing security services on the internet. Technical report, 2001.

[19] Daniel M. Russell and Mark Weiser. The future of integrated design of ubiquitous computing in combined real & virtual worlds. In *Communications of the ACM*, page 275. ACM, 1998.

[20] Ravi S. Sandhu, Edward J. Coyne, Hal L. Feinstein, and Charles E. Youman. Role based access control models. In *IEEE Computer*, volume 2, February 1996.

[21] M. Satyanarayanan. Integrating security in a large distributed system. *ACM Transactions on Computer Systems*, 7:247–280, August 1989.

[22] Roy Want, Andy Hopper, Veronica Falcao, and Jon Gibbons. The active badge location system. Technical report, Olivetti Research Ltd. (ORL), 1992.

[23] Mark Weiser. The computer for the 21st century. Scientific American, September 1991.