hyperDRIVE: Leveraging LDAP to Implement RBAC on the Web

Larry S. Bartz
Internal Revenue Service
U.S. Department of the Treasury
Indianapolis, Indiana 46204
lbartz@infostream.cr.irs.gov
(317) 226-7060

Abstract

hyperDRIVE is the working name of a strategy for implementing authentication, authorization, and access services in an n-tiered internet computing environment. hyperDRIVE employs the Lightweight Directory Access Protocol (LDAP) to store and access hyperDRIVE-defined data objects which are evaluated and manipulated to implement Role Based Access Control (RBAC).

Introduction

The widespread transition to Web-based and associated internet technology computing platforms has provided fertile ground for the germination and cultivation of authorization strategies. The degrees of sophistication and effectiveness of the many currently available approaches vary widely. None has yet proven itself clearly superior. None yet provides for the economical scalability necessary to support integrated internet or intranet computing environments which are composed of many applications, hosted by many servers.

While RBAC is well recognized as a strategy which reduces the cost and complexity of security administration, RBAC which is implemented on a host-by-host basis is still potentially inadequate in a multiple host environment. The core feature of internet computing, its facile interconnectedness, reveals the limited effectiveness of authorization strategies which are implemented on a per-host basis. When the customers of network computing resources shift their computing focus among many multiply-tiered applications (which are hosted by many separate servers) via mere mouse clicks, the potential for incongruity and inconsistency among the many host-based authorization implementations becomes obvious. Who (or what mechanism) can assure that an individual's authorizations don't conflict when the authorization controls are host-based?

Host-based authorization strategies are inherently difficult to audit and manage: How many applications and systems is Suzie User authorized to access? How big is the company? How many systems does it possess? If the authorization scheme is host-based, can one ever be sure of Suzie's potential for mayhem? And in this fragmented host-based authentication and authorization environment, how many lognames and passwords must Suzie remember? Surely she would prefer a single sign-on.

In the context of a large, multiple application, multiple host network computing environment, an integrated, centralized, auditable, manageable authorization database is clearly preferrable to the fragmented, disintegrated host-based alternative.

These considerations provide the motivation for the design of hyperDRIVE. The targeted functionality is a consistent, integrated, auditable, manageable RBAC implementation which is employed by many servers, clients, and applications, scalable to support thousands of clients and hundreds of servers and applications .

Design Considerations

hyperDRIVE's integrated RBAC information base is implemented via the Lightweight Directory Access Protocol (LDAP). LDAP provides a layer of abstraction and standards-based interoperability above the underlying RBAC data implementation. LDAP's widespread adoption among many directory services developers and vendors provides for wide latitude in the implementor's choice. The hyperDRIVE RBAC information base can therefore be defined and hosted by directory and database implementations such as X.500, Novell directory services, Microsoft directory services, relational databases, and "native LDAP" directories such as are available from Sun, Netscape, and the University of Michigan. LDAP client software and client API programming tools are also widely available. The hyperDRIVE proof of concept implementation employs University of Michigan LDAP. A transition to X.500 is intended and forthcoming.

All of hyperDRIVE's custom code is programmed in Java, chosen principally for its "write once, run anywhere" quality. Java is an object oriented language, so the initial and defining design activity was to "find the objects" in the design. The use of LDAP also assisted in this process, as an LDAP directory is composed of objects. A simple compari-

son of LDAP jargon to database jargon holds that LDAP objects possess attributes just as database records possess fields.

The discovery and definition of hyperDRIVE's key objects focuses upon the principal responsibilities of an integrated security function in an intranet computing environment, namely authentication, authorization, and access.

We define the responsibilities of each, as follows:

Access, in the hyperDRIVE context, includes the location, methods, and modes of access to and of network computing resources. For some, the familiar acronym CRUD (create, read, update, delete) is included in this concept. Responsibilitiy for confidentiality and integrity is also encompassed within the functionality implied by the object which defines access. In RBAC terminology, this object points to an operation or privilege. The hyperDRIVE object "operationAccessor" (see Table 1) fulfills this responsibility.

Authorization, in the RBAC context, is the mapping or aggregation of one or more operations or privileges to a defined role. The hyperDRIVE object "role" (see Table 2) fulfills this responsibility.

In the hyperDRIVE context, roles are assigned or mapped to persons, not vice versa as some RBAC literature describes. A person potentially possesses many roles. Roles do not possess persons.

Authentication and authorization are certainly separate responsibilities and separate activities. Much of the published RBAC literature treats the choice of authentication strategy as inconsequential with respect to the functionality of the RBAC strategy. hyperDRIVE, on the other hand, specifies X.509 certificate-based authentication as an integral and indispensible component.

The X.509 certificate's "subject" attribute (or field), is the Distinguished Name (DN) of the certificate's possessor, which is a person. The DN is a globally unique direct index into the same LDAP directory information base which holds the operationAccessor and role objects. The X.509 "subject" DN binds the authenticated identity (of the person who possesses and presents the certificate at run time) to an object in the directory which describes the person. The hyperDRIVE object "hyperDrivePerson" (see Table 3) fulfills this responsibility.

Figure 1, "LDAP Directory Objects and Relationships" graphically depicts the static relationships of the hyperDRIVE objects as they are represented in the LDAP directory.

The most significant and most obvious software components of the hyperDRIVE implementation include: a Java-capable and SSL-enabled Web client (such as Netscape); an SSL-enabled Web server (such as Apache/Stronghold or Netscape); an LDAP server (such as University of Michigan's or Netscape); an Object Request Broker (such as HORB, JacORB, OrbixWeb, Visigenic); and the hyperDRIVE client Java applet.

The roles, responsibilities, and run time behaviors of these components are described in detail in Table 4, "Operational Scenario". A brief summary follows.

Behaviorial Summary

hyperDRIVE's authentication mechanism is via X.509 certificates, which clients and servers exchange and verify to establish mutually authenticated Secure Socket Layer (SSL) sessions.

hyperDRIVE's authorization model is Role Based Access Control (RBAC). The Lightweight Directory Access Protocol (LDAP) directorycontains objects which describe people (hyperDrivePerson) and objects (operationAccessor) which describe enterprise information resources and the operations which involve those resources. Additional directory objects (role) are aggregations of operationAccessor. One or more role objects are assigned to each person, thus achieving the RBAC objective.

hyperDRIVE provides its customers with a Java applet, which is a GUI navigation guide, or menu. The hyperDRIVE navigation guide is constructed on the fly from the customer's LDAP-hosted RBAC profile.

hyperDRIVE provides servers, applications, and active objects with capabilities and facilities to consult the LDAP-hosted RBAC data. Through this consultation, the entities assure themselves of the appropriateness of customer requests. hyperDRIVE empowers servers to protect their resources, empowers applications and objects to protect themselves.

table 1: operationAccessor Object Class Specification

Description:

The operationAccessor object class is the representation of an operation (referenced by the URL) which requires restricted access authorization and navigational representation in the directory.

Attributes:

- **commonName**: the full, formal name of the operationAccessor.
- **description**: mildly verbose description .
- **objectOwner**: distinguished name(s) of the persons or role(s) which "own" the right to permit or deny authorization to access and manipulate the target object/ operation.
- **organizationalUnitDN**: the distinguished name of the highest-level organizational unit to which the target object/operation belongs.
- **execURL**: the URL through which the target object/operation is accessed, executed, manipulated
- **applicationCertificate:** the X.509 certificate, containing public key and other attributes, which the target object/operation uses to identify itself and its actions.
- **review+ApprovalDN**: the DN of the operationAccessor through which the owner's right and responsibility is exercised. .
- **maintenanceDN:** the DN of the operationAccessor through which maintenance on this object is performed.
- **member:** distinguished name(s) of directory object(s) which are permitted access to the target .
- **hyperDriveObjectType**: a string value which indicates what general type of object is represented. Suggested values include: "production", "information", "administrative", and "review+approval".

Naming Rules:

The commonName attribute must be used for naming.

Structure Rules:

A directory entry for operationAccessor must have an immediately superior entry of organizationalUnit.

Relationships:

organizationalUnit
role

table 2: role Object Class Specification

Description:

The role object is a container for aggregations of operationAccessor objects.

Attributes:

- **commonName**: the full, formal name of the role
- **description:** mildly verbose
- **roleOwner**: distinguished name(s) of the persons, or roles which "own" the right to permit or deny assignment of the role to persons.
- **organizationUnitDN**: the distinguished name of the highest-level organizational unit to which the role object belongs.
- **review+ApprovalDN**: the DN of the operationAccessor object through which the owner's right and responsibility is exercised.
- **maintenanceDN**: the DN of the operationAccessor object through which maintenance on this object is performed.
- **includedRole:** a list of one or more distinguished names (DN) of role object. This attribute supports the implementation of role hierarchies.
- **conflictingRole**: a list of one or more distinguished names (DN) of role object. This attribute supports the implementation of mutually exclusive role policies.
- **operations**: a list of one or more distinguished names (DN) of operationAccessor object.

Naming Rules:

The commonName attribute must be used for naming.

Structure Rules:

A directory entry for operationAccessor must have an immediately superior entry of organizationalUnit.

Relationships:

organizationalUnit
operationAccessor
hyperDrivePerson

table 3: hyperDrivePerson Object Class Specification

Description:

The hyperDrivePerson object is an extension of the directory's person object, and is a container for aggregations of role objects.

Attributes:

- **objectOwner**: distinguished name(s) of the person(s), or role(s) which "owns" the right to permit or deny authorization to access and manipulate this object.
- **review+ApprovalDN**: the DN of the operationAccessor through which the owner's right and responsibility is exercised.
- **maintenanceDN**: the DN of the operationAccessor through which maintenance on this object is performed.
- **roles:** a list of one or more distinguished names (DN) of role object. This attribute accomplishes the assignment of role objects to the person.

Naming Rules:

The commonName attribute must be used for naming.

Structure Rules:

hyperDrivePerson is a non-structural object class which extends another person-descriptive object class, such as newPilotPerson.

Relationships:

organizationalUnit
role

table 4: hyperDRIVE: Operational Scenario

A graphical representations of the following scenario is provided in Figure 2.

1. Web client establishes SSL [9] session with web server, attaining mutual authentication through exchange of X.509 [12] certificates. Client requests web page which contains the hyperDRIVE Guide Java applet. The customer's authenticated identity, as supported by the person's X.509 certificate, is their distinguished name (DN) reference to the LDAP directory object which describes the customer.

2. Web server returns hyperDRIVE Guide applet to client, securely passing the customer's authenticated identity as an applet execution parameter. The hyperDRIVE Guide applet is invoked within the Java Virtual Machine (JVM) of the web client suite.

3. hyperDRIVE Guide applet establishes distributed object communication with an object request broker (ORB) server which resides, according to the Java applet "sandbox" policy [5], on the same host from which the applet was served. The client applet requests role objects and operationAccessor objects which apply to the customer's DN. We refer to distributed object communication, by the name of the Object Management Group's Common Object Request Broker Architecture (OMG/CORBA) [11] Internet Inter-ORB protocol, or IIOP. The hyperDRIVE proof-of-concept implementation uses a freeware ORB, known as HORB [10]. The development roadmap for hyperDRIVE portends a transition to CORBA compliance and IIOP.

4. The ORB contacts the well-known LDAP server, requesting role and operationAccessor objects which apply to the customer's DN.

5. The LDAP server provides the requested role and operationAccessor objects.

6. The ORB provides the requested directory objects. The hyperDRIVE Guide applet can now act as a navigation tool for the customer, displaying names, descriptions, and links which describe operations for which the customer is authorized .

7. Through the hyperDRIVE Guide applet, the customer invokes an operation. The customer's web client suite invokes the URL against the target, while simultaneously attaining a mutually authenticated SSL session with the web server which hosts the targeted operation.

8. The web server asks an ORB-supported object to confirm or deny the customer's authorization to perform the requested operation A purpose-written Java program is invoked by a lightweight Common Gateway Interface (CGI) script to accomplish the task of communicating with the ORB. By this mechanism (steps 8-11), the web server protects the resource from unauthorized access.

9. The ORB contacts the well-known LDAP server, requesting role objects which apply to the customer's DN.

10. The LDAP server provides the requested directory objects.

11. The ORB-hosted object compares customer's authenticated identity (DN) and currently assigned roles, and (for this scenario) finds an affirmative match. The ORB returns a confirmation message to the web server, via the CGI/IIOP process.

12. The web server provides the requested resource to the customer's web client suite. In this scenario, the requested resource is a business object interface Java applet ("bizApplet", for short), which is invoked within the Java Virtual Machine (JVM) of the web client suite. An alternative scenario would have the web server return a static web object, or possibly a CGI entry point to executable content.

13. The bizApplet has (in this scenario) been programmed with an active capability to further protect itself against unauthorized operation, in the manner of [4]. The bizApplet contacts an ORB-supported object, seeking the specific LDAP objects which confirm the customer's authorization to perform the operation.

14. The ORB contacts the well-known LDAP server, requesting role objects which apply to the customer's authenticated identity.

15. The LDAP server provides the requested directory objects.

16. The ORB's response message to the bizApplet object includes (in this scenario) an affirmation of the customer's authorization. Since the bizApplet is certain of the customer's identity, and since the customer's identity is the customer's distinguished name from the enterprise LDAP directory, the bizApplet also has ready access to other customer attributes which might be pertinent within the context of the business transaction.

17. Satisfied of authorization concerns, and equipped with whatever other runtime tailoring information is necessary, the bizApplet object establishes an IIOP session with an ORB which resides on the same host (applet sandbox policy again) from which it was served.

18. The ORB's middleware functionality establishes and maintains communication with the host(s) which serve as the repositories of the enterprise business data. This channel of communication can be accomplished via many and various techniques, including CORBA, TPM, MOM, RPC, etc.

19. The return channel: business data, status messages, etc.

20. The ORB marshalls the business data into objects, which then exchange messages with the bizApplet objects.

Session Continuity: Steps 17-20 loop or cycle to maintain the IIOP session's statefulness. **Session Concurrency**: Steps 7-20 can be executed (with various targeted operations) while other steps 7-20 are in progress.

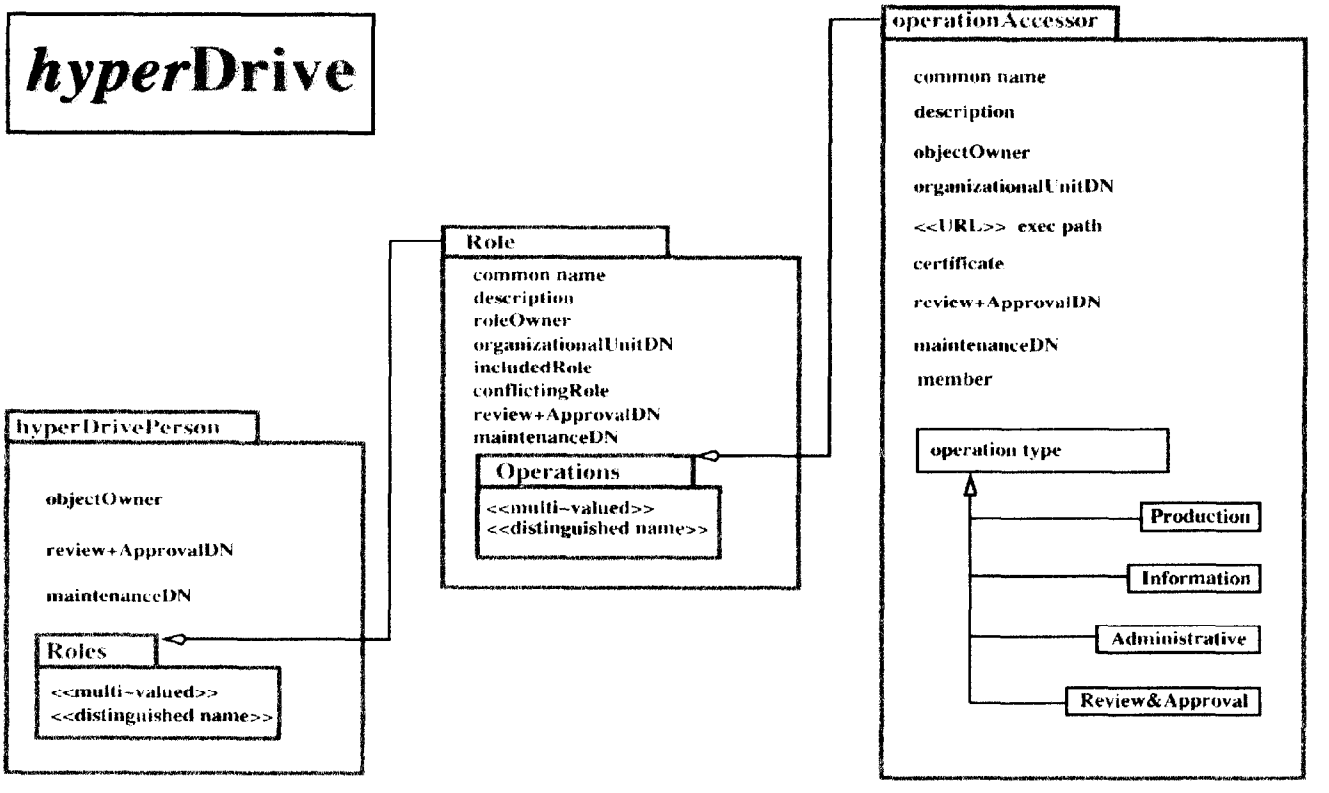

Figure 1: LDAP Directory Objects & Relationships

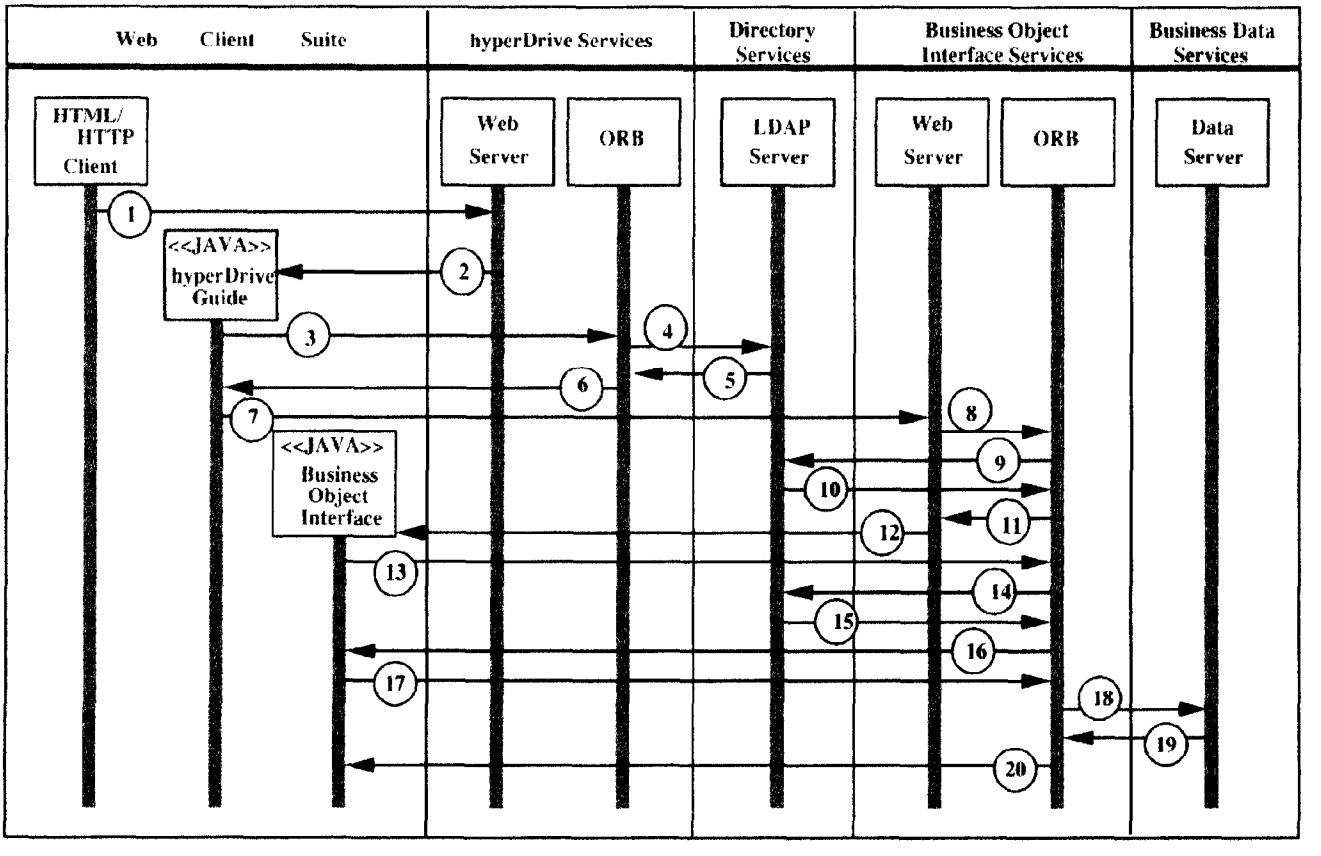

Figure 2: Activity/Sequencing

References

[1] David F. Ferraiolo and Richard Kuhn, "Role Based Access Control", Proceedings of the 15th NIST-NSA National Computer Security Conference, Baltimore, MD, 13-16 October 1992.
URL:http://hissa.ncsl.nist.gov/rbac/paper/rbac1.html

[2] John Barkley, "Implementing Role Based Access Control Using Object Technology", First ACM/NIST Workshop on Role-Based Access Control, Gaithersburg, MD, USA, Nov. 1995.
URL:http://hissa.ncsl.nist.gov/rbac/rbacot/titlewrkshp.html

[3] David F. Ferraiolo, Janet A. Cugini, D. Richard Kuhn, "Role-Based Access Control (RBAC): Features and Motivations", National Institute of Standards and Technology, U.S. Department of Commerce, Gaithersburg, MD 20899, 11th Annual Computer Security Applications Proceedings 1995.
URL:http://hissa.ncsl.nist.gov/rbac/newpaper/rbac.html

[4] Roy H. Campbell, Tin Qian, Willy Liao, Zhaoyu Liu, "Active Capability: A Unified security Model for Supporting Mobile, Dynamic and Application Specific Delegation", Security White Paper, System Software Research Group, Department of Computer Science, University of Illinois at Urbana-Champaign, February 16, 1996.
URL:http://choices.cs.uiuc.edu/Security/Papers/delegate.ps

[5] J. Steven Fritzinger, Marianne Mueller, "Java Security", (c) 1996, Sun Microsystems, Inc.
URL:http://www.javasoft.com/security/whitepaper.txt

[6] W. Yeong, T. Howes, S. Kille, "Lightweight Directory Access Protocol. Request for Comments (RFC) 1777", March, 1995.
URL:ftp://ds.internic.net/rfc/rfc1777.txt

[7] T. Howes, S. Kille, W. Yeong, C. Robbins, "The String Representation of Standard Attribute Syntaxes. RFC 1778", March, 1995.
URL:ftp://ds.internic.net/rfc/rfc1778.txt

[8] Rational Software Corporation, "UML Notation Guide, Version 1.0", January 13, 1997.
URL:http://www.rational.com/uml/start/notation_guide.html

[9] Netscape Communications Corporation, "The SSL Protocol, Version 3.0", March 1996.
URL:http://home.netscape.com/eng/ssl3/ssl-toc.html

[10] Satoshi, Hirano, "HORB: Distributed Execution of Java Programs", Electrotechnical Laboratory and RingServer Project, 1-1-4 Umezono Tsukuba, 305 Japan, March 1997.
URL:http://ring.etl.go.jp/~hirano/horb_wwwca97.ps

[11] Object Management Group, "OMA Executive Overview", February 3, 1997.
URL:http://www.omg.org/omg00/omaov.htm

[12] R. Housley, W. Ford, W. Polk, D. Solo, "Internet Public Key Infrastructure, Part I: X.509 Certificate and CRL Profile", Internet Draft,March 26, 1996
URL:ftp://ietf.org/internet-drafts/draft-ietf-pkix-ipki-part1-04.txt