A Closer Look at Authentication and Authorization Mechanisms for Web-based Applications

Conference Paper · January 2012

CITATIONS
6

READS
7,128

2 authors:

Sharil Tumin
Trimensity Tech
139 PUBLICATIONS **152** CITATIONS

SEE PROFILE

Sylvia Encheva
Høgskulen på Vestlandet
350 PUBLICATIONS **870** CITATIONS

SEE PROFILE

A Closer Look at Authentication and Authorization Mechanisms for Web-based Applications

SHARIL TUMIN
University of Bergen
IT Dept
P.O. Box 7800, 5020 Bergen
NORWAY
edpst@it.uib.no

SYLVIA ENCHEVA
Stord/Haugesund University College
Faculty of Technology, Business and Maritime Sciences
Bjørnsonsg. 45, 5528 Haugesund
NORWAY
sbe@hsh.no

Abstract: Authentication is a process by which you provide proofs that you are who you claim to be. Authorization is granting you valid permissions. Everyone is familiar with authentication i.e. login process but not so with authorization. Web-based applications introduced the needs for more understanding of these two processes to both users and implementors. Security data are managed into two related categorizations; authoritative and operational. To provide flexible and efficient administration, accounts, groups and resources data are managed distributively following organizational structures and are based on delegation of rights and responsibilities. To optimize operation, security data are duplicated into directory server to be used by Web-based applications.

Key–Words: Authentication, authorization, access control, security management, security implementations

1 Introduction

We the users need to authenticate ourself every time we initiate interaction with a computer system and are then given permissions dependent on our authorization data, either automatically or by providing some extra authorization credentials depending on the operational security policies governing the protected resources of the organizations and enterprises we belong to.

Basically, an authentication process is a way by which you provide proofs to a system, that you are in fact, who you claimed to be, by presenting a valid credential. At its simplest form this credential is a pair of user identification and password. A user identification ($\mathcal{U}$) is a unique string of characters or digits directly related to a user entity whether it is a human, a thing or a program. A password ($\mathcal{P}$) is a secret phrase known only to the user and not directly known to the system, but directly related to the user identification. Thus, to be authenticated $\mathcal{I}$, a user needs to provide a valid pair of [$\mathcal{U}$,$\mathcal{P}$].

Authentication is not authorization, but to be authorized you need to be authenticated first, since permissions mandate three attributes conditions of 1) *who*, 2) *what* and 3) *which*, within a user's work session in relation to $\mathcal{I}$ with a system. A permission ($\mathcal{Q}$) can be described by a triplet [$\mathcal{I}$, $\mathbb{O}$, $\mathbb{R}$] where $\mathcal{I}$ is the validated identity, $\mathbb{O}$ is a non empty set of allowable operations, and $\mathbb{R}$ is a non empty set of resources.

Permissions are real time attributes for controlling users' access and operations to protected resources. Authorization data is much more involved than the authentication data. To provide real time user's permissions, authorization processes need to be assisted by groups and groups memberships coupled with resource owners and resource groups. Management of authorization data is necessarily delegated to different managers closely related to organizational structure and resource ownership.

Authentication and authorization are explicitly defined in any operating system for example, Linux, Windows, and MacOS. Different DBMS (database management system) manage authentication and authorization differently, some are more involved and complicated than others. Whatever methods being used, the bottom line is; authentication - identifies valid users and authorization - provides correct permissions of access and operations on protected resources, in another word, controlling $\mathcal{Q} = [\mathcal{I}, \mathbb{O}, \mathbb{R}]$ at any one time, where $\bigcup_{i=1\ldots n} = \mathbb{Q}_{\mathcal{U}}$ represents all permissions granted to a particular user $\mathcal{U}$.

More and more enterprise critical applications

ISBN: 978-1-61804-089-3

provided to users are Web-based. Users interact with the underlying services provided by computer systems using Web browsers. The works concerning authentication and authorization are becoming more important than ever before, to both users and developers of Web-based applications. The general security of the underlying and supporting systems for a Web-based application are very much dependent om how the authentication and authorization are implemented on the application level.

In this paper we concentrate on the problems and solutions to authentication and authorization mechanisms and processes for Web-based applications in relations to users, groups, resources authoritative data management and operational data deployment. In particular we will be presenting a distributive, delegated management model coupled with LDAP (Lightweight Directory Access Protocol) directory services to be used as a centralized authentication and authorization server for all Web-based applications throughout the enterprise.

2 Background

Any centralized security implementation must necessarily reflect the underlining organizational structure of an enterprise. A typical structure would be hierarchical and implemented using a treelike structure. In a hierarchical organization, every entity in the organization, except the top most one, is subordinate to a single other entity. In an educational organization, a university is divided into faculties. Under each faculty are departments and each department is subdivided into units. A hierarchical organization naturally supports distributive management model through delegation of rights and responsibilities at each level of the hierarchy.

Persistence data are stored in two commonly used data storage structures; 1) relational database, and 2) data directory. These structures are optimized for data searching and selecting for different information storage and retrieval operations.

As to the relational database type, data are stored in a collection of relations implemented as data tables. Each data tables are organized as columns of attributes and rows of entry to represent a particular objects. Applications can access data by submitting queries to insert, select, modify and delete rows of data entries using SQL (Structured Query Language). It is necessary for each data row in a data table to be uniquely identifiable. The identification is done by some combination (one or more) of its attribute values. Different data tables in the same database can be joined together under SQL operations [1].

A data directory store data in a DIT (Directory Information Tree) where data are represented in a hierarchical treelike structure uniquely addressable by DNs (Distinguished Names). Each object identified by a DN holds attributes. Each attribute has a name, type and description, and one or more values. The attributes' names and values are defined by a schema [2]. There are three levels of search scope in a DIT; 1) *scope_base* - the object entry only, 2) *scope_onelevel* - its immediate decendents only (excluding the object), and 3) *scope_subtree* - the object and its DIT subtree. Data retrievals can specify a set of attributes to be retrieved, where an empty set means all attributes.

2.1 Central Identity Management

An IdM (Identity Management) system manages users' accounts and groups related data to be used to identify and authorize users across computer networks within an organizational domain and inter-domain collaborations [3]. It behooves the organization to have a central IdM to cater for enterprise wide authentication/authorization mechanisms to all organizational's Web-based applications.

In order to be flexible and responsive, the IdM should be deployed with two main conceptual parts; 1) *authoritative*, and 2) *operational*, as shown in Figure 1. The *authoritative* part implements the administrative functions of the system while the *operational* supports the LDAP-based authentication and authorization mechanisms for the Web-based applications.

Given that the organization structure is hierarchical, management rights and responsibilities can be delegated and distributed at each level of the hierarchy. Let assume that we have a simple hierarchical structure with maximum depth of three and $\mathcal{M}_{000000}$ is the master manager group i.e. members of this group have all managerial rights. Lower manager groups are labeled with subscripts in relation to their position in the organizational tree.

The first two digits signify faculties, the middle two digits signify departments and the last two digits signify units. For examples; 1) $\mathcal{M}_{110000}$ is the manager group for the whole faculty, 2) $\mathcal{M}_{112600}$ is the manager group for a department under a faculty, and 3) $\mathcal{M}_{112614}$ is the manager group for a unit under a department.

ISBN: 978-1-61804-089-3

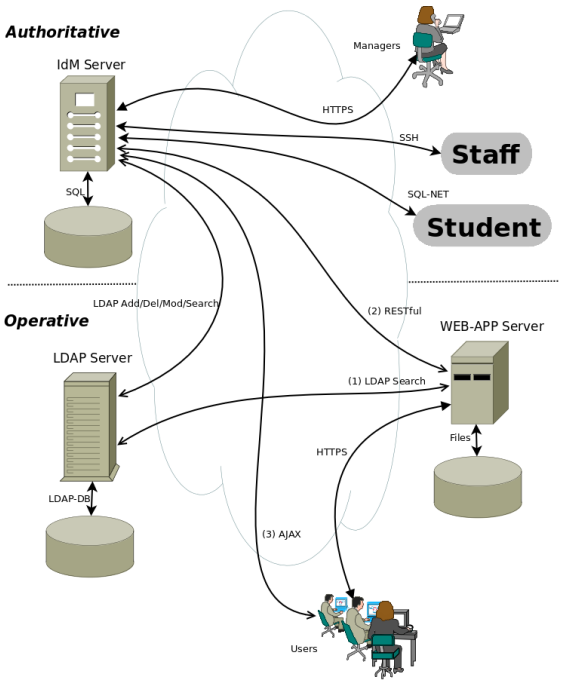

Figure 1: A Typical IdM Architecture

In fact, the subscripts scheme discussed here can be used to label any type of groups, for example; 1) $\mathcal{U}_{112600}$ – users belonging to a department, 2) $\mathcal{R}_{112614}$ – resources belonging to a work unit. It is natural to label groups that reflect the organizational structure. Authoritative data are stored in a relational database containing data tables that map these relations. Operational data stored in a data directory DIT will naturally follow the organizational tree for data objects that implement organizational security policies.

2.2 Accounts, Groups, Resources and Controls

Web-based applications provide services to users. Users are those registered to the IdM by having users' accounts. A person having an account will at it simplest form know a pair of security credential namely, user-identifier ($\mathcal{U}$) and user password ($\mathcal{P}$). An account is basically an identity of a person or a process coupled with credential data which provide proof of authenticity. Providing a valid credential will validate a person thereby granting access to protected applications and permits operations on resources depending on user's roles.

The password is only known to the person owning the account. Users' passwords must never be stored in clear text by the system. Passwords are normally stored by the system in cryptographic hash [4] values as exemplified in Table 1.

Table 1: Cryptographic Hashes Values

Functions	Values (Password = 123abc#!)
Crypt	SA7QICgO9uVOQ
MD5	cyV9tkiFa6dKZ6h+NRgbvA==
SHA1	iXr77gK7q4bdCRX4zfXqsWeb/rU=
	Values (Password = 123abc#!Z)
Crypt	SA7QICgO9uVOQ
MD5	UMIjRuU9uCzfKRCy/aAotQ==
SHA1	VIdFcOQS0VErsMXi/RxX5Y2yMPA=

There are three basic mechanism of access control [5]:

DAC – Discretionary Access Control
MAC – Mandatory Access Control
RBAC – Role Based Access Control

A *DAC* is a credential based access control. A user is identified by a validation process of user-identification and user password. Group members can also be validated using group's password. A user is the owner to resources for example all files under the user's home directory and sub-directories. The user, at her own discretion can change permissions of users including her own self for all resources owned by her. The user can transfer ownership of a resource or group of resources to another user. In an operating system, a special super user practically owns all system resources. This super user is called *root* user in a *Unix* system. A person authenticated as *root* has all the permissions on all resources.

A *MAC* is often used to control highly sensitive resources. It is centrally managed, enforcing organizational security policies in which all resources belong to the organization with no possibility of individual ownership. Each protected resource is assigned a security tag depending on what security level the resource was assigned to. A resource with a high security level can be downgraded after a period of time. Users with a specific security level can access resources labeled with the same or lower level. The management of security levels for both resources and users will follow strict procedural protocols and centrally defined policies.

A *RBAC* is an access control based on an individual roles of a particular user. Roles are defined

in relation to persons' work responsibilities and organizational security policies. Access control based on groups and roles could be applied successfully within organizations with clear hierarchy of authorities and separation of duties, for example; 1) hospitals, 2) banks, and 3) universities. Conceptually, RBAC is defined by named sets containing users, resources and operations as shown in Figure 2. Groups and Resources pools are managed centrally and could be delegated by strict sign-in and sign-offs and procedures in relation to the organizational hierarchical structure.

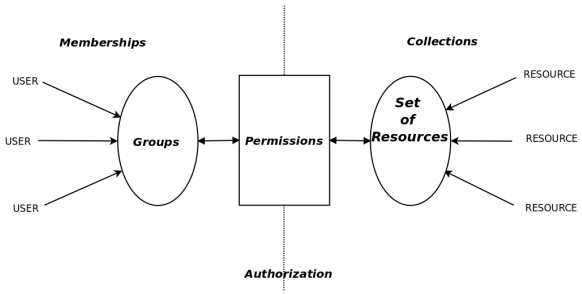

Figure 2: RBAC: Memberships, Permissions and Authorizations

2.3 LDAP

LDAP (Lightweight Directory Access Protocol) is an application protocol for accessing information stored in a simpler version of X.500 directory server implementation. LDAP only supports a subset of the comprehensive X.500 directory services specifications. While X.500 supports a multi-servers distributive directory configuration, LDAP is only supporting a single client-server operation.

A directory server implementing LDAP supports these data model and core services:

i. *Hierarchical Names* – Data components are defined in a hierarchical directory with hierarchical names in contras to a flat directory like phone-books. A data component such as uid=edpst,ou=people,dc=uib,dc=no represent a user with user-identifier *edpst* defined under an organizational components *people* belong to organization *uib* within a country *no*.

ii. *Typed Name Components* – Named (i.e. addressable) data components are typed, for example; UID (User Identification), CN (Common Name), O (Organization), OU (Oragnizational Unit), C (Country), DC (Domain Component). This in contrast to a typeless name schemes such as the domain name for example siam.uib.no. Some components type are leaf (external) nodes, e.g. uid=edpst,ou=people,dc=uib,dc=no while others are internal nodes, e.g. ou=people,dc=uib,dc=no. The name components are case insensitive i.e. UID=STU001,OU=People,Dc=uib,dc=NO is treated as equal to uid=stu001,ou=people,dc=uib,dc=no.

iii. *Typed Objects* – Data components are specific collections of data objects. Data objects have types defined by object classes, for example People, Organizations, and Computers.

iv. *Typed Attributes* – Data objects are set of data attributes with specific names and values, for example, *set of names*, *telephone number*, and *given name*. The attributes values are usually encoded as UTF-8 (8-bit Unicode Transformation Format) strings. Some attributes can contain value with binary data such as JPEG (Joint Photographic Experts Group) compressed digital pictures format.

v. *Directory Operations* – These operations are; 1) read, 2) compare, 3) search, 4) add, 5) delete, and 6) modify.

vi. *Security Models* – Individual data component can be protected using internal ACL (Access Control List). Permissions are given to a specific user after a successful authentication process call LDAP bind. Public access are given all with anonymous bind i.e. binding without user identification and password.

3 Implimentation

Web browsers, i.e. application clients communicate to Web-based applications servers using HTTP (Hypertext Transfer Protocol). Massages, controls and data are send in HTML (HyperText Markup Language), XML (Extensible Markup Language), CSS (Cascading Style Sheets), JavaScript and binary images.

Most Web-based applications provides LDAP authentication as a standard configuration, however none provide a standard authorization mechanism using LDAP. This is not surprising since authorization is very much dependent on applications, resources and local security policies.

As LDAP directory services are modeled after a client-server architectural model; firstly, a centralized IdM for enterprise user accounts management is a prerequisite condition and secondly, all Web-based applications will need to establish working connection with the directory server. There are three ways by which a Web-based application acting as direc-

ISBN: 978-1-61804-089-3

tory service client can get information from a centralize directory server; 1) directly, by using supporting LDAP client libraries, 2) indirectly via REST (Representational State Transfer) services from a HTTP server, and 3) indirectly using AJAX (Asynchronous JavaScript and XML) on a Web browser from a HTTP server that supports AJAX, refer to Figure 1.

3.1 Authentication

There are two complimentary parts in data management which concern users accounts, credentials, roles and permission, namely 1) *authoritative*, and 2) *operative* as shown in Figure 1. Authentication is about providing a correct credential during the validation process of login or sign on to a Web-based application. While user account creation is nothing more than registering a person with an user identification and user password. There are many details in the process of account creation which effect matters concerning; 1) efficiency, and 2) security of an enterprise.

With efficiency in mind, these concerns amongst other matters were implemented:

i.Web-based account application, registration, approval and creation.

ii. Web-based accounts management based on distributive rights and responsibility through delegation.

iii. Self-service management of editable personal data and password change.

iv. Disallow use of special local alphabetic characters in password, e.g. Å, å, Æ, æ, Ø, ø which will eventually introduce login problems due to different type of keyboards.

Security policies implementation is the major undertaking, a few important deployments are:

i. One person one account policy. No two or more persons sharing an account. A person only owns one user credential.

ii. Automatic user identifiers will ensure uniqueness and protection against trivial mapping of user name to user identification, e.g. the system will assign "Peter White" an identifier "pew028" instead of "peter".

iii. Enforce a strong password policies, e.g. more than 10 characters long, mix upper case, lower case, digits and non-alphanumeric.

iv. Enforce password aging and password recycle policies.

v. Implement automatic account expiration and deletion based on publicly accessible documented policies.

The *authoritative* data are stored in an Oracle RDBMS (relational database management system) use by the Web-based applications implementing the self-service and operators management functions. Authentication data are synchronized to the *operative* data storage, where each active account will have a data entry component in the LDAP server.

Normally user credentials are stored in *posixAccount* ldap's *ObjectClass* where user identity is stored in the 'required' *uid* attribute and user password is stored in the 'may have'*userPassword* attribute. To provide authentication both attributes must be used. The content of the *userPassword* can be a clear text password, but this should never be used, instead password is stored in Crypt or base64 encoded hash values as shown in Table 1. The SHA1 hash function is better then MD5 and take note that Crypt uses only the first 8 characters. There is no reason not to use the SHA1 for storing users passwords in LDAP server.

A login process is a mapping $[\mathcal{U},\mathcal{P}]_{valid} \mapsto \mathcal{I}_{session}$, i.e. as long as $\mathcal{I}_{session}$ is valid, its holder will have the rights of $\mathcal{U}$. For Web-based application the $\mathcal{I}_{session}$ is stored as one of the data in session cookies of the Web browser, $\mathcal{I}_{session} \in \mathbb{S}$.

3.2 Authorization

An authorization process is granting a set of permissions $\mathbb{Q}_{\mathcal{U}}$, where each $Q_i \in \mathbb{Q}_{\mathcal{U}}$ can be fully described by a triplet $[\mathcal{I}, \mathbb{O}_i, \mathbb{R}_i]$, for examples,

Q_1:[$\mathcal{I}$, {read},{http://form.uib.no}]
Q_2:[$\mathcal{I}$, {read, write},{file://store.uib.no/data}]
Q_3:[$\mathcal{I}$, {manage},{dit:ou=it,dc=uib, dc=no}]

remember that $\mathcal{I}$ for a specific session holds the reference to the validated user and consequently groups and roles the user $\mathcal{U}$ is a member of. While $\mathcal{I}$ changes from session to session, the $\mathcal{U}$ referred to by a particular $\mathcal{I}_{session}$ will always identify the same enterprise user. In another word, to be authorized, a user must be authenticated first.

HTTP is a stateless protocol [6]. In order for a web server to know in what state a web client (web bowser) is currently in, the web client needs to send state data back to the web server each time a new request is made by the web client. The state data are stored in web cookies. A web cookie has name and value pair and these attributes; 1) domain, 2) path, 3) expiration time, 4) secure flag, and 5) httponly flag. A web cookie and its attributes must not be longer than 4095 bytes [7].

All authenticated to a Web-based application users have session data stored in cookies. There are

two ways to store user's session data; 1) C_b, all data are stored in web cookies by the web browser, and 2) C_s, only session identifier $\mathcal{I}$ is stored in a web cookie, while session data are stored in a database on the web server. C_s provides the most flexible solutions.

Authorization data can be obtained from the LDAP server; 1) all at once, $\mathbb{Q}_{\mathcal{U}}$ and store the set in C_s, as a part of authentication process, or 2) on demand, Q_i depending on which $\mathcal{R}_i \in \mathbb{R}_i$ and store it in C_b. Permissions on $\mathcal{R}_i$ can be checked and operation $O_i \in \mathbb{O}_i$ can be granted based on data store in C_b or C_s. Since ad hoc management of access controls data are done, C_b provides the most up-to-date and accurate authorization data, while C_s needs user $\mathcal{U}$ relogin.

3.3 Distributive Management

Authentication data are stored under a flat *uid* branch of a DIT, normally under the ou=people,dc=uib,dc=no, for example uid=edpst,ou=people,dc=uib,dc=no.

To manage access controls data, three management interfaces on a Web-based application are provided; 1) roles, 2) resources, and 3) permissions. The DIT is arranged to reflect the organizational tree where each organization unit is identified by a 6 digits number. Each organization unit is placed under the ou=roles,dc=uib,dc=no, for example ou=221000,ou=roles,dc=uib,dc=no.

A group for example zoo under 221000 will be named as ou=zoo,ou=221000,ou=roles,dc=uib,dc=no. Roles will be placed under each group, for example:

cn=admin,ou=zoo,ou=221000,ou=roles,dc=uib,dc=no
cn=user,ou=zoo,ou=221000,ou=roles,dc=uib,dc=no

where each role will holds a group of *uid* names. The role cn=admin,ou=... is special. This *admin* role has the rights to manage all DIT entries under the ou=... i.e. members of the role can add, delete, and modify entries in the branches below.

All resources owned by 221000 will be placed under ou=221000,ou=res,dc=uib,dc=no, for example cn=sebra,ou=221000,ou=res,dc=uib,dc=no in which a groups of resources are defined. Permissions are defined under a resource data entry for example:

cn=read:opr,cn=sebra,ou=221000,ou=res,dc=uib,dc=no.
cn=add:opr,cn=sebra,ou=221000,ou=res,dc=uib,dc=no.

where each permission will hold a group of *uid* names. Members of the group have the permission to perform operation X defined in the cn=X:opr... on the group of resources defined in its parent.

The distribution of management responsibilities and rights for the whole DIT are effectuated by the cn=admin,ou=... data entries at any position of the DIT. Managers of an organizational unit will be defined as members of cn=admin,ou=... for that unit, for example:

cn=admin,ou=221000,ou=roles,dc=uib,dc=no
cn=admin,ou=221000,ou=res,dc=uib,dc=no

Any ou=... can have another ou=... which defined an internal node. Any ou=... can have a cn=... which defined an external node. Any amount of distributive model can be implemented in the DIT.

4 Conclusions

We have described in details how authentication and authorization for Web-based applications can be modelled and implemented using LDAP. Working habits of modern people are changing. The current and most probably the future trend is mobility. Touch screen internet tablets and smartphone would be the future tools of choice. These small and light devices will have theirs programs, application and data store elsewhere in the Internet clouds. In this respect, the Web plays a major role in providing mobile users with critical business data, applications and services. The understanding of how authentication and authorization can be managed and deployed securely and effectively is a key factor for continuing success of the Web.

References:

[1] Momjian, Bruce: PostgreSQL: Intrduction and Concepts. Addison-Wesley, ISBN 0-201-70331, 2001.

[2] Abrech, Henry: Directory Service Integration and Deployment Guide. Oracle, Release 2 (9.2), Part No. A96579-01, 2002.

[3] Buecker, Axel: Identity Management Advanced Design IBM, Redbooks series, SG24-7242-00, First Edition, 2006.

[4] Ferguson, Niels; Schneier, Bruce: Practical Cryptography Addison-Wesley, ISBN 978-0471223573, 2003.

[5] Endler, David et al.: A Guide to Building Secure Web Application OWASP, 2002.

[6] Fielding, R et al.: RFC 2616 Hypertext Transfer Protocol HTTP/1.1 The Internet Society, 1999.

[7] Kristol, D; Montulli, L.: RFC 2109 HTTP State Management Mechanism Network Working Group, 2010.