OpenHorizons

Issue 20, Q4 2012

The Novell Technology Magazine

Filr: The secure solution for sharing files

GWAVA Reload | *LDAP Authentication* | **Building Social Intranets**
Filr – Next Generation Filesharing | *Shorewall* | **Migrate or Upgrade?**
Case Notes: Millikin University | Community News | **Ask The Experts**

LDAP Authentication Against eDirectory For The Apache Neanderthal

by Lawrence Kearney

In many organisations system administrators are called upon to connect our credential rich, web enabled world. These tasks fall to system administrators by default, because Apache instances run on servers managed by them, or because Apache administrators lack depth in Light Directory Access Protocol (LDAP) and directory service knowledge. This breed of system administrator is what I refer to as an "Apache Neanderthal". They use a few simple pictures and tools to describe and build something that suits their need. Subsequently, this allows the next gatherer in line to build something more elegant. However, if the "credentials carved in the stone" are examined more closely, Neanderthals too are capable of elegance.

Apache "AAA" is easier today, really.

"AAA" in the context of Apache stands for **A**uthentication, **A**uthorisation and **A**ccess Control. Essentially it describes the credentials, usually in the form of user names and passwords, that are required to proceed (authentication). Credentials can be correlated with other information to establish privilege (authorisation), and isolation to particular computers or devices is also an option (access control). Apache AAA services are provided by loading modules and libraries designed to extend Apache for these services. In the past, prior to Apache development efforts levelling somewhat, this was a more cumbersome configuration to deal with. This may well help explain the "you can hear a pin drop in the room" phenomenon when the Apache AAA discussions start.

Fortunately the extension of Apache for AAA is much simpler today. Specifically for versions 2.2 and above. Fewer modules and libraries are required to extend Apache to enable these services. Furthermore, additional functionality and security features have been added as well. The marriage of the "new" AAA knowledge and a little directory service experience can make a more secure and flexible web service infrastructure possible for your complex social group, clan or organisation.

Where to begin

When passing credentials over networks, security should be one of the first planning considerations. This article will focus on securing and optimising communications between clients, Apache instances and directory service user stores. Specifically Apache on Suse Linux Enterprise Server and eDirectory.

The first task should be to understand how many "As" you should implement and where security, using encrypted communication channels, should be used. Often the best way to do this is to diagram the deployment so AAA work flow can be visualised.

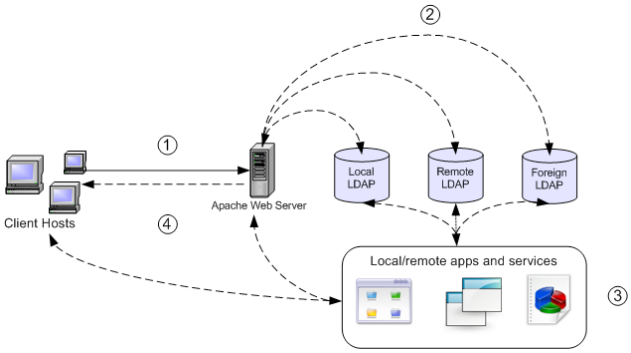

Figure 1: Client and Web Application communication workflow

Consider the data flows in figure 1:

1. HTTP sessions transferring credentials from the clients to the servers should be secure.
2. LDAP communications between the Apache instance and the directory service may occur on the local server, to a remote server on the same network or to a remote server that is off site
3. Participating web applications may be local or remote. Resulting downstream credential operations may also be directed to local, remote or foreign LDAP services.
4. Communications from the participating applications to the clients may be sent from the applications themselves, or from the designated Apache instance(s).

Securing communications between clients and servers using SSL certificates is a well documented process and will not be expanded on beyond reference here. We will however go into some detail on how to implement and

LAWRENCE KEARNEY joined Georgia Health Sciences University in 1998 and is currently a System Support Specialist. He previously worked in the business sector in New York. He has experience of many Novell and SUSE Linux solutions and co-authored the Novell GroupWise 8 Best practice Wiki.

verify secure communications between Apache instances and their LDAP accessible user stores. The technical tutorial will include explanations of the relevant Apache modules and their use

Modules are where it's at

Apache modules can be built into Apache or loaded using the Apache configuration file. The first method, technically compiling modules into the Apache application, offers performance gains that would likely benefit specialised or large scale deployments. The second improves the flexibility of the Apache service but would impose a slight performance tax that could be noticed with larger deployments.

Apache allows us to use several criteria to authenticate and authorise users to web server stuff.

Consider the following options::

Identity credentials
Identity credentials + directory service data
Identity credentials + directory service data + host data

Translating this into Apache modules use we get:

mod_auth_basic
mod_auth_basic + (mod_ldap and mod_authnz_ldap)
mod_auth_basic + (mod_ldap and mod_authnz_ldap) + mod_authz_host

A brief description of these modules and the features they provide:

mod_auth_basic: user lookup service for the Apache instance
mod_ldap: LDAP directive awareness and service optimisation
mod_authnz_ldap: LDAP authentication "and" authorisation services
mod_authz_host: authorisation and access control using network address or host environment info
mod_ssl: provides the ability to use the host server SSL facilities to create secure connections

When configuring Apache or any other application for LDAP access to a directory service there are some key configuration aspects to consider.

Directory access: multiple LDAP service addresses can be used. Alternatively LDAP targets can be load balanced or made highly available. SSL security should be used for remote LDAP servers.

Object and attribute rights: authenticated LDAP proxy user configurations can be used to force Apache to use credentials other than "anonymous" for directory searches. This is useful if searches are performed on attributes and values intended to be protected from anonymous access. In eDirectory environments anonymous access employs rights of the [Public] pseudo-object. Of particular importance are the rights to the "CN" attribute. For secure proxy user configurations, assign rights to both protected attributes and the attributes available using anonymous access to LDAP proxy users.

Optimising for performance and security: directory servers can, and should, index attributes that are searched frequently to improve performance. Apache LDAP searches can also be limited in scope using LDAP filters for additional performance gains. As an added security measure consider limiting Apache LDAP search result cache "Time To Live" (TTL) values so that changes to objects in the directory are more quickly reflected into the Apache search cache.

Still discussing modules

Verifying the modules built into Apache can be accomplished using the "**httpd2 -l**" command as displayed here.

```
darkvixen163:/home/admin # httpd2 -l
Compiled in modules:
  core.c
  prefork.c
  http_core.c
  mod_so.c
```

Figure 2: Displaying modules that are built into Apache

Next we assess modules loaded by Apache. An easy way to add, remove and display these modules is to use the **a2enmod** and **a2dismod** commands (these commands are specific to the SUSE Linux distributions). Use the "**a2enmod -l**" command to list the ones currently in use.

```
darkvixen163:/home/admin # a2enmod -l
actions alias auth_basic authn_file authz_
groupfile authz_default authz_user
authn_dbm autoindex cgi dir env expires
include log_config mime negotiation
setenvif ssl suexec userdir php5 rewrite
```

Figure 3: Displaying modules loaded by Apache

If any of the required modules are not listed, enable them using the following commands:

a2enmod mod_ldap
a2enmod mod_authnz_ldap
a2enmod authz_host

	EXAMPLE 1	EXAMPLE 3
Directives	AuthType Basic (**mod_auth_basic**) AuthBasicProvider ldap AuthName "DarkVixen protected content" AuthzLDAPAuthoritative On (**mod_authnz_ldap**) AuthLDAPUrl "ldaps://192.168.2.160/o=dvc?cn?sub" Require valid-user (**core**)	AuthType Basic (**mod_auth_basic**) AuthBasicProvider ldap AuthName "More DarkVixen protected content" AuthzLDAPAuthoritative On (**mod_authnz_ldap**) AuthLDAPUrl "ldaps://192.168.2.160/o=dvc?cn?sub?(|(objectClass=inetOrgPerson)(objectClass=groupOfNames))" Require ldap-group cn=IS_G,ou=IS,ou=INFOTECH,o=DVC
Expanation	• Configures a credential based LDAP user service for Apache to use • Prevents other authentication and authorisation methods being used if the LDAP method fails • Implements a secure LDAP server connection • Performs a search from the "dvc" organisation down for the user CN attribute • Authenticate and authorise the user if it exists in the LDAP directory	• Configures a credential based LDAP user service for Apache to use • Prevents other authentication and authorisation methods being used if the LDAP method fails • Implements a secure LDAP server connection • Limits LDAP searches to user and group object classes, increasing directory search efficiency • Performs a search from the "dvc" organisation down for the user CN attribute • Requires the user to be a member of a specific group to authorise the user
What Happens	• Apache authenticates (binds) anonymously to the LDAP server to search for the user • If the user CN is found, the user is authorised to attempt authentication • If the user authenticates successfully the HTTP content is served	• Apache authenticates (binds) anonymously to the LDAP server to search for the user using only user and group object classes • Apache performs a compare operation to verify the CN used is a member of the specified group, if the compare is successful the user is authorised to attempt authentication • If the user authenticates successfully the HTTP content is served
	EXAMPLE 2	EXAMPLE 4
Directives	AuthType Basic (**mod_auth_basic**) AuthBasicProvider ldap AuthName "DarkVixen protected content" AuthzLDAPAuthoritative On (**mod_authnz_ldap**) AuthLDAPUrl "ldaps://192.168.2.159/o=dvc?cn?sub" "ldaps://192.168.2.160/o=dvc?cn?sub" Require ldap-attribute objectClass=inetOrgperson	AuthType Basic (**mod_auth_basic**) AuthBasicProvider ldap AuthName "Even more DarkVixen protected content" AuthzLDAPAuthoritative On (**mod_authnz_ldap**) AuthLDAPUrl "ldaps://192.168.2.160/o=dvc?cn?sub?(|(objectClass=inetOrgPerson)(objectClass=groupOfNames))" AuthLDAPBindDN "cn=APACHE,ou=PROXIES,o=CORP" AuthLDAPBindPassword "novell" Require ldap-filter &(groupMembership=cn=FSA_G,ou=MCG,o=DVC)(employeeStatus=Active)
Explanation	• Configures a credential based LDAP user service for Apache to use • Prevents other authentication and authorisation methods being used if the LDAP method fails • Implements multiple secure LDAP servers in a fail over configuration • Performs a search from the "dvc" organisation down for the user CN attribute • Authenticate and authorise the user if it exists in the LDAP directory, but uses fewer cross-module directives. Ideally more efficient and more directory service friendly.	• Configures a credential based LDAP user service for Apache to use • Prevents other authentication and authorisation methods being used if the LDAP method fails • Implements a secure LDAP server connection • Limits LDAP searches to user and group object classes, increasing directory search efficiency • Performs a search from the "dvc" organisation down for the user CN attribute • Configures a LDAP proxy user for Apache to use to read protected attributes • Requires the user to be a member of a specific group "AND" have a value of "Active" for their "employeeStatus attribute to authorise the user
What Happens	• Apache authenticates (binds) anonymously to the LDAP server to search for the user • If the user CN is found, the user is authorised to attempt authentication • If the user authenticates successfully the HTTP content is served	• Apache authenticates (binds) as the specified user to the LDAP server to search for the user using only user and group object classes • Apache performs a compare operation to verify the CN used is a member of the specified group and has the correct value for the " employeeStatus" attribute, if both compare operations are successful the user is authorised to attempt authentication • If the user authenticates successfully the HTTP content is served

Table 1

Finally, check you work ensuring the modules are now listed and restart Apache.

```
darkvixen163:/home/admin # a2enmod -l
actions alias auth_basic authn_file
authz_groupfile authz_default authz_user
authn_dbm autoindex cgi dir env expires
include log_config mime negotiation
setenvif ssl suexec userdir php5 rewrite
mod_ldap mod_authnz_ldap
authz_host
```

Figure 4: Verifying the required modules are configured to load

Modules can also be added or removed from Apache configurations on SLES servers by modifying the "APACHE_MODULES=" section of the "/etc/sysconfig/apache2" file.

Making the connections

Moving forward with the configuration bits, assuming client HTTP connections have been secured, the first task is to secure the communications between the Apache instance and the LDAP service. If your LDAP instance is local, meaning on the same host as Apache, SSL connectivity may not be required.

You will require the SSL certificate from your LDAP server(s) in the format required by the LDAP SDK in use on your Apache server. SLES servers use the OpenLDAP SDK requiring a PEM formatted certificate, but Novell, Netscape or some other SDK could be in use on your platform and may require a different certificate format.

This configuration should apply globally, so the "/etc/apache2/default-server.conf" file is modified with the following LDAP directives provided by the mod_ldap modules.

In the examples listed here and in Table 1 the directives will be grouped together by the module providing them. The module name is in bold following the first directive of a group, and is not intended to be used in actual configurations.

LDAPTrustedGlobalCert CA_BASE64 /etc/apache2/certs/darkvixen160.crt **(mod_ldap)**
LDAPTrustedMode SSL
LDAPOpCacheTTL 300

These three directives:

- Specifies the certificate type and file to use for LDAP SSL/TLS connections
- Specifies the mode (and port) to use for secure LDAP connections
- Limits LDAP operations cache TTL for search results to 5 minutes

Be sure to check the Apache "error_log" file for any information that may indicate certificate format issues or any other LDAP service connectivity errors.

The configuration examples in Table 1 (previous page) could apply globally to Apache configurations, to discreet virtual hosts or even to specific URLs. Place the required directives in the appropriate configuration files.

Use the Apache or LDAP server logs to verify your service connections are occurring using SSL or TSL. If the LDAP server is an OES server the iMonitor and the DS Trace utilities are very useful here.

LDAP error codes can be researched using SDK documentation, or using a popular LDAP Wiki maintained by community LDAP gurus at "http://ldapwiki.willeke.com".

Finally

Several useful Apache AAA configurations have been presented. However some useful authorisation directives have been omitted for brevity, such as the "Allow", "Deny" and "Satisfy" directives provided by the **mod_authz_host** module.

Allow and Deny directives operate in a hierarchy fashion and manage the authorisation of clients using network address and even local environment criteria. Network address values could include individual hosts, network ranges or domain name criteria. Local environment criteria can include browser versions or local operating system environment settings as examples.

The Satisfy directive can be used to group authorisation logic to make even more complex decisions when more than one authorisation method is employed. Apache requires Satisfy directives evaluate as 'true" before authorisation is successful. The "Any" value for the directive authorises the user if any of the methods evaluated succeed. Conversely, the "All" value requires all methods to succeed.

With Apache versions 2.3 and above the Satisfy directive can be used to authorise users without this unanimity of result.

Additional information on Apache AAA implementations can be found on the authors web site at :
www.lawrencekearney.com/files/Apache_AAA_against_eDir-LDAP.pdf