About Mark

- Who Am I
- What I do

Goal

- To understand the roles LDAP can play in regards to Apache

What We'll Cover

- What are directory services
- Why LDAP
- LDAP Basics
- Why Apache and LDAP
- Apache, Perl and LDAP
- Useful things today
- Future/Questions

Why Directory Services

- Compliment DBMS
- Provide Quick Read Access
- Provide Distributed Data Management
- Provide Replicated Data Management
- Hierarchical Data Relationships

Why LDAP

- Open, Standard Protocol
- Runs over IP (e.g. it's Internet ready)
- Provides an access model
- Provides a security model
- Provides a data model
- Provides a naming model

LDAP Basics

- Access Model
  - client/server using TCP/IP for transport
  - query and modification support
- Security Model
  - default no access
  - waterfall ACL
  - support for SSL/TLS, SASL and clear-text

LDAP Basics cont.

- Data Model
  - object-like
  - flexible & structured & standardized & extensible
    - RDBMS/ODBMS not standardized
  - distributed
  - Replication
  - Ex: uid,cn,mail,userpassword

LDAP Basics cont.

- LDAP Naming Model
  - Follows X.500
  - Each entry is unique
  - Comprised of attributes in the entry
    - uid=mewilcox,ou=people,dc=unt,dc=edu

Why Apache and LDAP

- Authentication
  - Who Are You
- Authorization
  - What Can You Do
- Configuration
  - How do I manage a server farm

Apache, Perl and LDAP

- Apache is successful for its API
- mod_perl makes it easy to use Apache API
- Net::LDAP and PerLDAP make it easy to use LDAP from Perl
- Net::LDAP is “pure” Perl
- PerLDAP is Perl wrapper for Netscape’s C API

Useful Things Today

- Authentication
  - Use LDAP to authenticate users
    - userid and password
    - SSL
- Authorization
  - Role based access using groups
- Active Directory

Authentication Example

- How authentication works in LDAP
- BASIC

Authentication Configuration

In httpd.conf

```
<Directory “/opt/apache/htdocs”>
#only set the next two if you need to bind as a user for searching
 #PerlSetVar BindDN "uid=user1,ou=people,o=acme.com" #optional
 #PerlSetVar BindPWD "password" #optional
 PerlSetVar BaseDN "ou=people,o=acme.com"
 PerlSetVar LDAPServer ldap.acme.com
 PerlSetVar LDAPPort 389
 PerlSetVar UIDAttr uid
 PerlAuthenHandler Apache::AuthNetLDAP
AuthName "LDAP Test Auth"
 AuthType Basic
 require valid-user
</Directory>
```

Authenticate Code

```
my ($result, $password) = $r->get_basic_auth_pw;
      return $result if $result;
      my $username = $r->connection->user;
…
my $ldap = new Net::LDAP($ldapserver, port => $ldapport);

      #initial bind as user in Apache config
      my $mesg = $ldap->bind($binddn, password=>$bindpwd);
…
my $attrs = ['dn'];
     $mesg = $ldap->search(
                      base => $basedn,
                      scope => 'sub',
                      filter => "($uidattr=$username)",
                      attrs => $attrs
                     );
```

Authenticate Code cont.

```
unless ($mesg->count())
    {
         $r->note_basic_auth_failure;
         $r->log_reason("user $username: user entry not
 found for filter: $uidattr=$username",$r->uri);
         return AUTH_REQUIRED;
    }


   my $entry = $mesg->shift_entry;
    $mesg = $ldap->bind($entry->dn(),password=>$password);
```

Authenticate Final

- Compare
  - Less resource intensive
  - Quicker
  - Server may not enforce password policies
- compare(dn,”userpassword”, encrypted_password)

Authorization Example

- Groups
  - No standard
  - GroupOfNames (member)
  - GroupOfUniquenames (uniquemember)
  - Dynamic Groups (groupOfUrls)
- LDAP URLs
  - ldap://pandora.acs.unt.edu/o=unt.edu??sub?sn=wilcox

Authorization config

```
use Apache::AuthzNetLDAP;
  PerlSetVar BindDN "cn=Directory Manager"
  PerlSetVar BindPWD "password"
  PerlSetVar BaseDN "ou=people,o=unt.edu”
...
require valid-user
require user mewilcox
require user mewilcox@venus.acs.unt.edu
require group "cn=Peoplebrowsers1,ou=UNTGroups,ou=People,
     o=unt.edu"
require ldap-url ldap://pandora.acs.unt.edu/o=unt.edu??sub?sn=wilcox
```

Authorization Code

- Similar to AuthNetLDAP
- First need user's DN
- Check which "require" matches
- Call proper subroutine
- `_getIsMember`
- `_checkURL`

Authorization cont.

- _getIsMember()

```
my $mesg =
$ldap->compare($groupDN,attr=>"uniquemember",value=>$userDN);

my $mesg =
$ldap->compare($groupDN,attr=>"member",value=>$userDN);

If result (mesg->code()) is 6 then it's a match. Else we
   continue.

Do a recursive search if any of the member values are group
   objects
```

Authorization cont.

- LDAP URLs
  - _getIsMember()
  - _checkURL
- If user entry matches LDAP query
- ldap://host:port/base??scope?filter

Authorization cont.

- Why LDAP URLs?
  - Easier to manage
    - Large multi-valued attributes difficult to handle
  - More scalable
- How I use them at UNT
  - Course memberships
  - Entry pointer, Group pointer
  - Emulates many to many relationship

Authorization cont.

- Get filter from URL. Chunk the rest

```
use URI;
my $uri = new URI ($urlval);
   my $filter = $uri->filter();
my @attrs = $uri->attributes();
   my $mesg = $ldap->search(
                base => $userDN,
                scope => "base",
                filter => $filter,
                attrs => \@attrs
                );
```

- If search returns an entry, it matches

Module Futures

- To compare or not to compare
- To cache or not to cache
- DECLINE on not found (this one done just needs to be posted to CPAN)
- Only test on original request?
- Clean up code
- Document and Test

Configuration Example

- <Perl>
- LDAP schema and hierarchy
- UDP or TCP?
- Security
- Make it easier to manage server farms

Questions

- Visit me on openldap-general or perl-ldap mailing list
- Or send me a message at mark@mjwilcox.com