Poznańskie Centrum Superkomputerowo - Sieciowe

PROJEKT LDAP

Analiza i dostosowanie istniejących standardów dotyczących autoryzacji dostępu oraz uwierzytelniania użytkowników:
projekt uniwersalnego interfejsu dla aplikacji typu portal z punktu widzenia technik uwierzytelniania i autoryzacji

Jerzy Chochulski
Sebastian Szuber

Poznań 2002

Spis treści

1 Wstęp

Analiza dotyczy następujących obszarów zastosowania LDAP :

- uwierzytelnianie i autoryzacja użytkowników w aplikacjach typu portal,
- projekt interfejsu API, służący integracji usługi LDAP z wszelkiego rodzaju aplikacjami opartymi o WWW, w których uwierzytelnianie i autoryzacja użytkowników jest wymagana.

Pierwszym krokiem analizy jest zdefiniowanie postaci bazy użytkowników na serwerze LDAP, określeniu, jakie obiekty oraz ich atrybuty mogą być użyte w tym celu. Aplikacje typu portal do uwierzytelniania i autoryzacji przy pomocy usługi LDAP mogą użyć modułów zaimplementowanych w serwerach WWW. W związku z tym postać bazy danych powinna spełniać wymagania już istniejących rozwiązań. Opis bazy danych wraz z przykładem wykorzystywanym przy analizie serwerów WWW znajdują się w następnym rozdziale.

Jak wspomniano powyżej aplikacje typu portal umieszczone na serwerach WWW mogą użyć metod uwierzytelniania z zastosowaniem usługi LDAP zaimplementowanych w samych serwerach. Przegląd popularnych serwerów (Apache, Tomcat oraz iPlanet Web Server) został zawarty w rozdziale trzecim.

Aplikacje typu portal jak i aplikacje innego typu mogą użyć zewnętrznego interfejsu do zapewnienia ochrony dostępu do danych. W rozdziale czwartym przedstawiony jest projekt interfejsu realizującego uwierzytelnianie i autoryzację użytkowników aplikacji.

Rozdział piąty przedstawia wnioski z przeprowadzonej analizy.

2 Projekt bazy danych użytkowników LDAP

Baza danych LDAP musi być odpowiednio skonstruowana w celu zapewnienia odpowiedniej zawartości oraz struktury przechowywanych informacji dla celów uwierzytelniania oraz autoryzacji.

2.1 Użytkownicy

Wszystkie trzy prezentowane w następnym rozdziale serwery WWW umożliwiają autentykację i autoryzację przy pomocy serwera LDAP. Każdy z nich ma pewne wymagania odnośnie bazy użytkowników. Wśród nich iPlanet Web Server definiuje najdokładniej wymagania wobec bazy użytkowników. Dwa pozostałe serwery pozwalają na pewną dowolność odnośnie użycia pewnych atrybutów w obiekcie użytkownika. Tomcat i Apache pozwalają na określenie nazwy atrybutu w DN zawierającego identyfikator użytkownika. Tomcat pozwala także na wybranie atrybutu zawierającego hasło użytkownika. iPlanet Web Server wymaga aby w DN użytkownika był jego identyfikator przedstawiany jako atrybut **uid**, a hasło użytkownika było pamiętane w atrybucie **userPassword**. Powoduje to konieczność zachowania unikalności atrybutu uid. Dla potrzeb prezentacji użytkownika musi on również posiadać atrybut **cn** definiujący jego nazwę.

Aby obiekt użytkownika mógł zawierać powyższe atrybuty musi on należeć do klasy **inetOrgPerson** [RFC2798] lub innej klasy dziedziczącej z klasy inetOrgPerson. Klasa ta wymaga istnienia w obiekcie atrybutów **objectClass** [RFC2256], **cn** (commonName) [RFC2256], **sn** (surname) [RFC2256]. Aby obiekt spełniał wymagania opisane powyżej musi również zawierać atrybuty **uid** (userID) [RFC1274] oraz **userPassword** [RFC2256].

Przykład obiektu użytkownika zawierającego wymagane atrybuty podany jest poniżej:

```
dn:              uid=jank, ou=People, dc=man, dc=poznan, dc=pl
objectClass:     top
objectClass:     person
objectClass:     organizationalPerson
objectClass:     inetOrgPerson
cn:              Jan Kowalski
sn:              Kowalski
uid:             jank
userPassword:    ha123slo
```

Tak zdefiniowany schemat, zawierający identyfikator użytkownika oraz jego hasło spełnia minimalne wymagania dotyczące autentykacji. Oddzielną kwestią jest autentykacja użytkownika na podstawie jego certyfikatu możliwa przy zastosowaniu protokołu SSL lub TLS.

2.2 Grupy

Kwestia autoryzacji może być rozwiązana na dwa sposoby:

- poprzez zdefiniowanie grup użytkowników,
- poprzez przypisanie użytkownikom odpowiednich atrybutów.

Wszystkie serwery WWW reprezentują jednakowe, standardowe podejście do definiowania grup. Grupy reprezentujące role użytkowników są obiektami typu **groupOfUniqueNames** [RFC2256], a nazwa grupy jest określana przez atrybut **cn** (commonName) [RFC2256]. Atrybut wielowartościowy **uniqueMember** [RFC2256] zawiera nazwy obiektów wszystkich członków danej grupy.

```
dn:              cn=administratorzy, ou=People, dc=man, dc=poznan, dc=pl
objectClass:     top
objectClass:     groupOfUniqueNames
cn:              administratorzy
uniqueMember:    uid=jank, ou=People, dc=man, dc=poznan, dc=pl
uniqueMember:    DN użytkownika 2
...
uniqueMember:    DN użytkownika n
```

Podejście polegające na definiowaniu grup użytkowników ma kilka wad, które ujawniają się przy dużej liczbie użytkowników oraz grup. Zaczynają się wówczas pojawiać problemy z administracją dużą ilością obiektów oraz problemy z wydajnością spowodowane dużą liczbą członków grup.

W takich przypadkach rozwiązaniem mogą być grupy dynamiczne oraz autoryzacja na podstawie atrybutów obiektu użytkownika a nie poprzez sprawdzanie przynależności do grupy.

Grupy dynamiczne są na przykład zaimplementowane w serwerze iPlanet Directory Server wersja 4 i 5. Obiekt reprezentujący grupy dynamiczne dziedziczy z klas **groupOfUniqueNames** [RFC2256] oraz **groupOfURLs** [IDSADM, IDSSCH]. Uczestnictwo w grupie dynamicznej jest definiowane poprzez wielowartościowy atrybut **memberURL** [IDSADM, IDSSCH], który może zawierać URL LDAP. Za członków grupy są wówczas przyjmowani użytkownicy spełniający warunki zawarte w filtrze zdefiniowanym przez URL [RFC2254, RFC2255].

Niestety obsługa grup dynamicznych nie jest obecnie objęta żadnym standardem. Mechanizm ten nie jest również obecnie zaimplementowany w projekcie OpenLDAP - darmowej implementacji serwera LDAP [OPENLDAP], nie jest też uwzględniona w najbliższych planach tego projektu.

```
dn:              cn=pokój22, ou=People, dc=man, dc=poznan, dc=pl
objectClass:     top
objectClass:     groupOfUniqueNames
objectClass:     groupOfURLs
cn:              pokój22
memberUrl:  ldap:///ou=People,dc=man,dc=poznan,dc=pl??sub?(roomNumber=22)
```

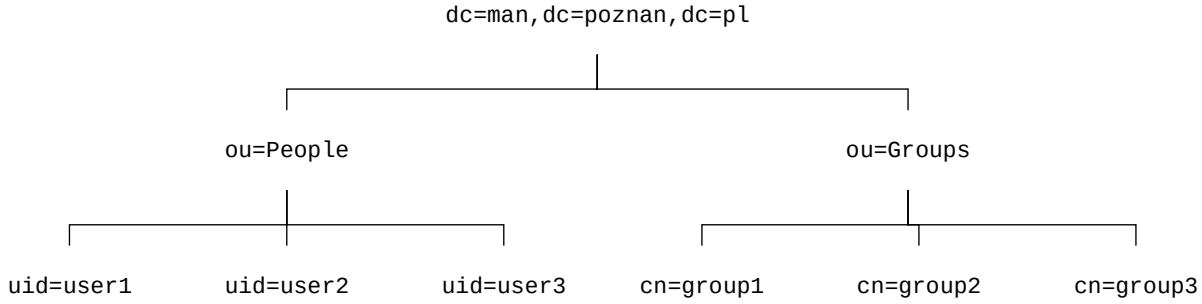
Aby móc oprzeć autoryzację użytkownika o atrybuty zapisane w jego obiekcie obiekt ten musi zapewniać odpowiednią ilość informacji. W ramach inicjatywy EDUCAUSE/Internet2 [EDUCSE, INT2] powstała definicja klasy **eduPerson** [EDUPER] i atrybutów, które pozwalają lepiej opisać osobę związaną ze środowiskiem wyższej edukacji w USA. W Europie propozycja podjęcia takiej pracy [DEEP] została poddana w grupie roboczej TF-LSD [TF-LSD] działającej w ramach organizacji TERENA jednak nie uzyskała do tej pory akceptacji. Odpowiednia informacja zapisana w atrybutach użytkownika może również służyć do autoryzacji użytkowników pomiędzy różnymi organizacjami. Budową odpowiedniej infrastruktury dla takiej autoryzacji zajmuje się w ramach inicjatywy Internet2 projekt Shibboleth [SHIB].

Propozycja odpowiedniej definicji dla polskiego środowiska naukowego będzie przedmiotem jednego z kolejnych zadań w tym projekcie.

2.3 Drzewo katalogu

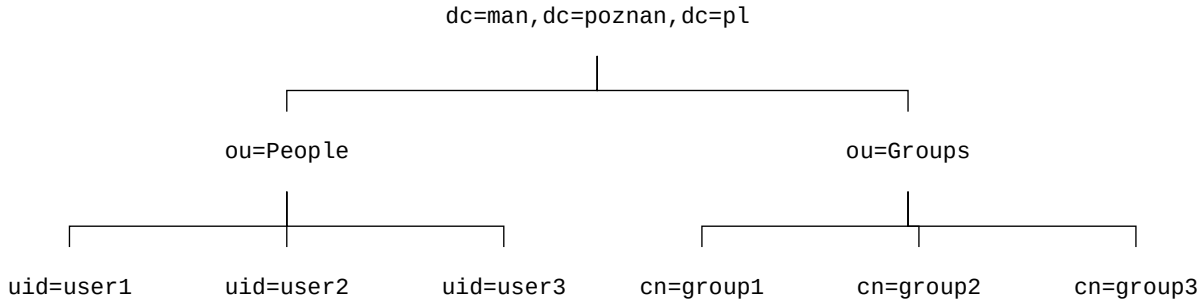
W tradycyjnym modelu bazującym na standardzie X.500 uważa się, że należy drzewo katalogu konstruować tak aby odwzorowywało strukturę organizacji. Taka konstrukcja powoduje dodatkowy nakład pracy administracyjnej związanej z aktualizacją zawartości drzewa zwłaszcza w organizacjach o rozbudowanej strukturze administracyjnej i dużej częstości zmian. Przykładem tego typu organizacji są wyższe uczelnie [LRECIPE]. Dlatego aby uprościć administrację informacja zawartą w drzewie katalogu proponujemy aby w

każdej organizacji utworzyć dwie gałęzie: pierwszą nazwaną **ou=People** i zawierającą obiekty użytkowników oraz drugą nazwaną **ou=Groups** i zawierająca obiekty gruo statycznych. Przykład takiej budowy drzewa znajduje się na rysunku poniżej.



2.4 Testowa baza danych

Do przeprowadzenia testów modułów autentykacji i autoryzacji w różnych serwerach została przygotowana testowa baza danych. Struktura tej bazy danych jest przedstawiona poniżej.



Korzeniem katalogu jest obiekt **dc=man, dc=poznan, dc=pl**. Poniżej są umieszczone dwie gałęzie:

- podkatalog użytkowników **ou=People, dc=man, dc=poznan, dc=pl**,
- podkatalog grup **ou=Groups, dc=man, dc=poznan, dc=pl**.

```
# Define an organizational unit 'People'
dn: ou=People, dc=man,dc=poznan,dc=pl
objectClass: top
objectClass: organizationalunit
ou: People

# Define an organizational unit 'Groups'
dn: ou=Groups, dc=man,dc=poznan,dc=pl
objectClass: top
objectClass: organizationalunit
ou: Groups
```

W katalogu ou=People zdefiniowano obiekty dla trzech użytkowników : user1, user2 i user3.

```
# Define a user named 'user1'
dn: uid=user1,ou=People, dc=man,dc=poznan,dc=pl
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: inetorgperson
givenName: user
cn: user 1
uid: user1
sn: 1
userPassword: user1

# Define a user named 'user2'
dn: uid=user2,ou=People, dc=man,dc=poznan,dc=pl
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: inetorgperson
givenName: user
cn: user 2
uid: user2
sn: 2
userPassword: user2

# Define a user named 'user3'
dn: uid=user3,ou=People, dc=man,dc=poznan,dc=pl
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: inetorgperson
givenName: user
cn: user 3
uid: user3
sn: 3
userPassword: user3
```

W katalogu ou=Groups zdefiniowano obiekty dla trzech grup statycznych:

- grupa **group1** zawierająca wszystkich trzech użytkowników,
- grupa **group2** zawierająca użytkownika user2 i user3,
- grupa **group3** zawierająca tylko użytkownika user3.

```
# Define all members of the 'group1' role
dn: cn=group1,ou=Groups, dc=man,dc=poznan,dc=pl
description: Group 1
objectClass: top
objectClass: groupofuniquenames
cn: group1
uniqueMember: uid=user1,ou=People, dc=man,dc=poznan,dc=pl
uniqueMember: uid=user2,ou=People, dc=man,dc=poznan,dc=pl
uniqueMember: uid=user3,ou=People, dc=man,dc=poznan,dc=pl

# Define all members of the 'group2' role
dn: cn=group2,ou=Groups, dc=man,dc=poznan,dc=pl
description: Group 2
```

```
objectClass: top
objectClass: groupofuniquenames
cn: group2
uniqueMember: uid=user2,ou=People, dc=man,dc=poznan,dc=pl
uniqueMember: uid=user3,ou=People, dc=man,dc=poznan,dc=pl

# Define all members of the 'group3' role
dn: cn=group3,ou=Groups, dc=man,dc=poznan,dc=pl
description: Group 3
objectClass: top
objectClass: groupofuniquenames
cn: group3
uniqueMember: uid=user3,ou=People, dc=man,dc=poznan,dc=pl
```

3 Serwery WWW

Przedstawione w tym rozdziale serwery WWW zostały wybrane jako najbardziej popularne aplikacje w swojej grupie, które już posiadają możliwość połączenia się z serwerem LDAP i użycia przechowywanej tam informacji o użytkownikach w procesie autentykacji i autoryzacji.

3.1 Apache

Do serwera WWW Apache [AHTTPD] można dodawać moduły realizujące dodatkowe funkcje, nie obsługiwane przez sam serwer. Moduły te mogą między innymi dokonywać autentykacji użytkowników oraz w połączeniu z plikiem konfiguracyjnym samego serwera dokonywać ich autoryzacji.

Jako pierwszy przedstawimy moduł **LDAP Authentication module for Apache** [ALDAPAUTH].

Opis integracji modułu z Apach'em w systemie Linux/Unix jest dostępny na stronie WWW modułu. W systemie Windows z zainstalowanym Apach'em można użyć gotowego DLL-a. W tym celu należy :

1. skopiować plik mod_auth_ldap.dll do katalogu *Apache*/modules,
2. dokonać wpisu do pliku konfiguracyjnego *Apache*\conf\httpd.conf:

```
# LDAP (Windows 9x/NT/2000)
LoadModule ldap_auth_module modules/mod_auth_ldap.dll
AddModule mod_auth_ldap.c
```

Moduł definiuje następujące dyrektywy używane w celu autentykacji i autoryzacji z wykorzystaniem LDAP :

- **AuthLDAPAuthoritative** – domyślnie ustawiona na 'yes'; W przypadku gdy Apache jest skompilowany razem z modułem (wtedy Apache zawsze próbuje dokonać autentykacji użytkowników z serwera LDAP) a autentykacja użytkowników ma być dokonana z innego źródła niż serwer LDAP, dyrektywa powinna być ustawiona na 'no';
- **LDAP_Server** – nazwa serwera LDAP (np. ldap.man.poznan.pl); możliwe jest zdefiniowanie więcej niż jednego serwera LDAP, np. **LDAP_Server** "ldap.man.poznan.pl ldap2.man.poznan.pl"
- **LDAP_Port** – port serwera LDAP (standardowo 389)
- **Base_DN** – bazowy DN (Distinguished Name) dla operacji przeszukiwania
- **Bind_DN** – DN (Distinguished Name) użytkownika dla połączenia z serwerem LDAP
- **Bind_Pass** – hasło użytkownika dla połączenia z serwerem LDAP
- **UID_Attr** – atrybut, w którym zapisana jest nazwa użytkownika; używany przy operacji przeszukiwania bazy użytkowników na serwerze w gałęzi Base_DN; standardowo przyjęty jest atrybut 'uid'.
- **require** – dyrektywa ta występuje w czterech formach :

  require user *nazwa_użytkownika1 nazwa użytkownika2* ...
  np. require user user2 user3

  require *nazwa_atrybutu wartość_atrybutu*
  np. require roomnumber "123"

  require filter zdefiniowany_filtr
  np. require filter "(&(telephonenumber=123456789)(roomnumber=12)"

  require group *atrybut=nazwa_grupy,nazwa_katalogu_grup*
  np. require group cn=group2,ou=Groups

Dyrektywy używamy do określania dostępu w pliku konfiguracyjnym httpd.conf. Przykładowy wpis do pliku httpd.conf – chronione są wszystkie zasoby katalogu htdocs/ldap – dostęp do nich mają tylko użytkownicy z grupy group2 – user2 i user3 (przykład odnosi się do przykładowej bazy użytkowników przedstawionej w drugim rozdziale; atrybutem przechowującym nazwę użytkownika w tym przykładzie jest ‘cn'). Dostęp do serwera LDAP realizowany jest w tym przypadku przez użytkownika ‘anonymous’.

```
<Directory "D:/Apache/htdocs/ldap">
Options Indexes FollowSymLinks
AllowOverride None
Order allow,deny
Allow from all
AuthName "Authorization Test"
AuthType Basic
LDAP_Server azalea
LDAP_Port 389
Base_DN "dc=man,dc=poznan,dc=pl"
UID_Attr uid
Require group cn=group2,ou=Groups
</Directory>
```

Moduł z góry zakłada, iż hasła są przechowywane w atrybucie ‘userPassword’ przy czym metoda ich kodowania w serwerze LDAP nie ma wpływu na wynik autentykacji.

Zalety modułu :

- moduł podobnie jak i serwer są darmowe,
- komunikaty generowane przez moduł pomagają w śledzeniu procesu uwierzytelniania i autoryzacji i usuwaniu ewentualnych błędów.

Wady modułu :

- hasło przechowywane w pliku konfiguracyjnym nie jest kodowane,
- brak możliwości połączenia Apache’a z serwerem LDAP przy pomocy protokołu SSL.

3.2 Java Servlets i Tomcat

Tomcat jest rozwijaną przez konsorcjum Apache i wspomaganą przez firmę SUN referencyjną implementacją technologii **servletów** [SERVLET], czyli aplikacji WWW napisanych w języku Java. Testy zostały przeprowadzone na serwerze Tomcat w wersji 4.0.1. Ta wersja Tomcat'a implementuje Java Servlet w wersji 2.3 oraz Java Server Pages w wersji 1.2 [SERVLET23].

3.2.1 Java Servlet API 2.3

Specyfikacja Java Servlet API w wersji 2.3 definiuje bezpieczeństwo aplikacji WWW obejmujące sposób autentykacji i autoryzacji na dwa sposoby:

- bezpieczeństwo deklaratywne,
- bezpieczeństwo programistyczne.

Bezpieczeństwo deklaratywne polega na określeniu zasad bezpieczeństwa obejmującego autentykację i autoryzację w sposób niezależny i zewnętrzny w stosunku do chronionej aplikacji. Miejscem, w którym znajduje się taka deklaracja bezpieczeństwa dotycząca aplikacji jest plik opisujący aplikację (ang. deployment descriptor). W pliku tym jest zapisana polityka bezpieczeństwa dotycząca konkretnego środowiska uruchomieniowego. W czasie działania aplikacji kontener servletów (ang. servlet container) używa tej polityki bezpieczeństwa do realizacji odpowiedniej autentykacji i autoryzacji.

Bezpieczeństwo programistyczne polega na udostępnieniu aplikacji (servletowi) metod objętych API, które pozwalają pobrać informacje o aktualnym stanie autentykacji i autoryzacji użytkownika wywołującego aplikację. W API zostały umieszczone trzy takie metody:

- getRemoteUser,
- isUserInRole,
- getUserPrincipal.

Metoda **getRemoteUser** zwraca nazwę użytkownika, która została użyta do autentykacji lub wartość null, gdy użytkownik nie podlegał autentykacji. Metoda **isUserInRole** sprawdza, czy użytkownik jest w roli (należy do grupy) o nazwie podanej jako argument wywołania. Nazwy ról podawane w aplikacji mogą być w pliku opisującym aplikację mapowane do innych nazw grup, co pozwala na dostosowanie wymagań aplikacji do istniejących grup bez potrzeby jej modyfikacji. Metoda **getUserPrincipal** zwraca informację o tożsamości autentykowanego użytkownika w formie obiektu klasy java.security.Principal.

Metody to pozwalają aplikacji pobrać informacje o bieżącym stanie autentykacji i autoryzacji i stosownie do swoich wymagań aplikacja może wówczas podjąć realizację żądanej akcji lub odmówić tej realizacji.

Autentykacja w systemach WWW może się odbywać na kilka sposobów:

- HTTP Basic,
- HTTP Digest,
- HTTPS,
- bazująca na formularzu.

Autentykacja **HTTP Basic** zdefiniowana w protokole HTTP/1.0 polega na przesłaniu przez przeglądarkę do serwera WWW nazwy użytkownika i hasła. Dane te są przesyłane otwartym tekstem, więc nie gwarantuje to zbyt wysokiego poziomu bezpieczeństwa - hasło może być podsłuchane. Aby podnieść ten poziom bezpieczeństwa można użyć w niższej warstwie protokołu SSL/TLS lub innej metody zabezpieczającej przesyłane informacje (np. IPSEC lub VPN). W metodzie tej serwer WWW nie podlega autentykacji.

Autentykacja **HTTP Digest** również bazuje na nazwie użytkownika i haśle, jednak dane te są przesyłane do serwera w formie zaszyfrowanej. Dzięki temu poziom bezpieczeństwa

przesyłanych danych jest znacznie wyższy niż w metodzie HTTP Basic. Niestety autentykacja HTTP Digest nie jest szeroko rozpowszechniona. W metodzie tej serwer WWW nie podlega autentykacji.

Autentykacja **HTTPS** jest silnym mechanizmem autentykacji, który może być wykorzystany przy użyciu protokołu HTTP po SSL/TLS. Mechanizm ten bazuje na użyciu certyfikatów kluczy publicznych infrastruktury PKI. Przy jego użyciu autentykowany jest również serwer WWW.

Autentykacja **bazująca na formularzu** pozwala przede wszystkim na elastyczne zaprojektowanie wyglądu formularza, poprzez który użytkownik wprowadza dane. Dane te są jednak przesyłane do serwera otwartym tekstem. Dlatego zaleca się tu również wykorzystanie protokołu HTTPS zapewniającego szyfrowanie przesyłanych danych oraz autentykację serwera.

3.2.2 Tomcat 4.0.1

Opisywana wersja Tomcat'a oferuje możliwość przeprowadzenia autentykacji i autoryzacji z wykorzystaniem serwera LDAP-owego. W tym celu należy dokonać zmiany w pliku konfiguracyjnym serwera, polegającej na wstawieniu elementu typu **<Realm>** o ponizej zdefiniowanej postaci do jednego z elementów <Engine>, <Host> lub <Context>. Tomcat wykorzystuje interfejs JNDI zawarty w języku Java jako sposób dostępu do serwera LDAP.

```
      <Realm      className="org.apache.catalina.realm.JNDIRealm"
                  debug =
                  connectionName =
                  connectionPassword =
                  connectionURL =
                  roleBase =
                  roleName =
                  roleSearch =
                  roleSubtree =
                  userPassword =
                  userPattern =
      />
```

Znaczenie parametrów:

- **debug** – poziom wypisywania komunikatów: 0 – brak, 99 - pełny,
- **connectionName** – nazwa użytkownika dla połączenia z serwerem LDAP,
- **connectionPassword** – hasło użytkownika dla połączenia z serwerem LDAP,
- **connectionURL** – adres URL do serwera LDAP,
- **roleBase** – bazowy DN (Distinguished Name) katalogu grup dla operacji przeszukiwania,
- **roleName** – nazwa atrybutu użytego do zapamiętania nazwy grupy,
- **roleSearch** – wzorzec przeszukiwania grup w LDAP,
- **roleSubtree** – ustawiany na 'true' gdy przeszukiwanie ma objąć poziomy poniżej zdefiniowanego w **roleBase**,
- **userPassword** – nazwa atrybutu użytego do zapamiętania hasła,
- **userPattern** – wzorzec przeszukiwania użytkowników w LDAP.

Przykładowy wpis w pliku konfiguracyjnym *serwera, dotyczący* bazy użytkowników opisanej w drugim rozdziale jest przedstawiony poniżej. Przykład ten przedstawia objęcie autentykacją i autoryzacją pojedynczej aplikacji rozmieszczonej w katalogu examples.

```
<!—Tomcat Examples Context -->
<Context
      path="/examples"
      docBase="examples"
      debug="99"
```

```
      reloadable="true">
<Realm  className="org.apache.catalina.realm.JNDIRealm"
            debug="99"
            connectionName="cn=Directory Manager"
            connectionPassword="secret"
            connectionURL="ldap://localhost:389"
            roleBase="dc=man,dc=poznan,dc=pl"
            roleName="cn"
            roleSearch="(uniqueMember={0})"
            roleSubtree="true"
            userPassword="userPassword"
            userPattern="uid={0},ou=People,dc=man,dc=poznan,dc=pl"
    />
 .
 .
 .
 </Context>
```

Dostęp poszczególnych grup użytkowników do zasobów aplikacji może być zdefiniowany w pliku opisu aplikacji (web.xml). Poniżej przedstawiony jest przykład definiowania dostępu do zasobów w katalogu: jsp\security\protected dla grupy group2 (user2 i user3).

```
<security-constraint>
    <display-name>Example Security Constraint</display-name>
    <web-resource-collection>
       <web-resource-name>Protected Area</web-resource-name>
        <!-- Define the context-relative URL(s) to be protected -->
        <url-pattern>/jsp/security/protected/*</url-pattern>
        <!-- If you list http methods, only those methods are protected -->
        <http-method>DELETE</http-method>
        <http-method>GET</http-method>
        <http-method>POST</http-method>
        <http-method>PUT</http-method>
    </web-resource-collection>
    <auth-constraint>
        <!-- Anyone with one of the listed roles may access this area -->
        <role-name>group2</role-name>
    </auth-constraint>
 </security-constraint>

 <!—Default login configuration uses form-based authentication -->
 <login-config>
    <auth-method>FORM</auth-method>
    <realm-name>Example Form-Based Authentication Area</realm-name>
    <form-login-config>
       <form-login-page>/jsp/security/protected/login.jsp</form-login-page>
       <form-error-page>/jsp/security/protected/error.jsp</form-error-page>
    </form-login-config>
 </login-config>
```

Aby autentykacja przebiegła prawidłowo należy pamiętać o zapewnieniu odpowiednich uprawnień dla użytkownika, którego nazwa i hasło jest wykorzystywana do połączenia się z serwerem LDAP. Brak możliwości wyszukania i odczytania właściwych obiektów i ich atrybutów z serwera LDAP uniemożliwia dokonanie autentykacji i autoryzacji.

Zalety rozwiązania:

- serwer jest darmowy,

- komunikaty generowane przez serwer w plikach *.log pomagają w śledzeniu procesu uwierzytelniania i autoryzacji,

Wady rozwiązania:
- hasło przechowywane w pliku konfiguracyjnym nie jest zakodowane.

3.3 iPlanet Web Server

Serwer iPlanet jest najbardziej zaawansowany ze wszystkich trzech omawianych serwerów, jeśli chodzi o współpracę z serwerem LDAP. Jednocześnie jednak ten serwer jest jedynym komercyjnym serwerem pośród testowanych. Wersja 6.0 oferuje, oprócz serwera WWW, serwer administracyjny (iPlanet Web Server Administration Server), który pozwala na konfigurację serwera WWW używając przeglądarki stron internetowych. Po zalogowaniu się do serwera administracyjnego użytkownik widzi szereg zakładek, wśród których dwie dotyczą tematyki LDAP. Zakładka „Global Settings" zawiera opcję „Configure Directory Service" w pasku po prawej stronie. Jej wybranie pozwala ustawić następujące parametry:

- Host Name – adres lub nazwa serwera LDAP,
- Port – numer portu serwera LDAP,
- Use SSL for connections – pozwala wybrać protokół SSL dla połączeń z serwerem LDAP,
- Base DN – bazowy DN dla operacji na serwerze LDAP,
- Bind DN – DN użytkownika dla operacji logowania na serwerze LDAP,
- Bind Password – hasło użytkownika (przechowywane w pliku konfiguracyjnym jest kodowane).

Po ustawieniu parametrów połączenia z serwem LDAP należy wybrać opcję „Save changes" (dane są zapisywane w plikach konfiguracyjnych), a następnie przeładować serwer WWW. Odtąd przy wszelkich poleceniach autentykacji i autoryzacji, serwer WWW będzie przeszukiwał bazę użytkowników na serwerze LDAP.

Zakładka „Users and Groups" udostępnia interfejs graficzny pozwalający na utworzenie całego drzewa użytkowników i grup na serwerze LDAP.

Zabezpieczenie zasobów serwera WWW jest realizowane przez tworzenie elementów ACL (Access Control List) w plikach konfiguracyjnych znajdujących się w katalogu httpacl. Znajdują się tam pliki o nazwach (po 2 dla każdego serwera WWW):

- generated.https-*nazwa_serwera*.acl
- genwork.https-*nazwa_serwera*.acl

W przypadku dokonywania zmian bezpośrednio w plikach należy się upewnić, że plik genwork.https-*nazwa_serwera*.acl jest kopią pliku generated.https-*nazwa_serwera*.acl. Poniżej przedstawiony jest przykład elementu ACL umieszczanego w pliku konfiguracyjnym.

```
acl "path=d:/iPlanet/WebSrv/docs/examples/*";
authenticate (user, group) {
  prompt = "Security Test Pages";
  database = "default";
  method = "basic";
};
deny (all) user = "anyone";
allow (all) group = "group2";
```

Przykład przedstawia zabezpieczenie zasobów katalogu **examples/**. Dostęp do zasobów mają jedynie użytkownicy należący do grupy o nazwie **group2** (są to użytkownicy **user2** i **user3**).

Serwer WWW zakłada, iż hasła są przechowywane w atrybucie ‘userPassword’ obiektu użytkownika. Metoda kodowania haseł w serwerze LDAP nie ma wpływu na wynik autentykacji.

Zalety rozwiązania:

- hasło (Bind Password) przechowywane w pliku konfiguracyjnym jest kodowane,
- możliwości połączenia iPlanet Web Serwer z serwerem LDAP przy pomocy protokołu SSL,
- wysoki stopień integracji serwera WWW z serwerem LDAP,
- możliwość tworzenia drzewa użytkowników i grup na serwerze LDAP bezpośrednio z poziomu serwera administracyjnego.

Wady rozwiązania:

- brakuje możliwości śledzenia procesu autentykacji i autoryzacji, przez co konfiguracja zabezpieczenia zasobów serwera sprowadza się do serii prób i błędów, aż do uzyskania poprawnego rezultatu,
- dość wysoka cena serwera WWW.

4 Interfejs dla uwierzytelniania i autoryzacji

Z przeprowadzonej analizy istniejących rozwiązań dotyczących autentykacji i autoryzacji użytkowników LDAP w aplikacjach bazujących na WWW wynika, że właściwie każdy liczący się na rynku produkt posiada lub może dołączyć moduły współpracujące z serwerem LDAP. Do rozwiązania pozostaje kwestia odpowiedniego zaprojektowania drzewa informacji LDAP tak, aby istniejąca baza użytkowników i grup mogła być wykorzystana przez różne serwery. Propozycja struktury drzewa informacji LDAP zaproponowana w rozdziale drugim spełnia te wymagania i może służyć jak wzór dla budowy bazy danych użytkowników w celu autoryzacji.

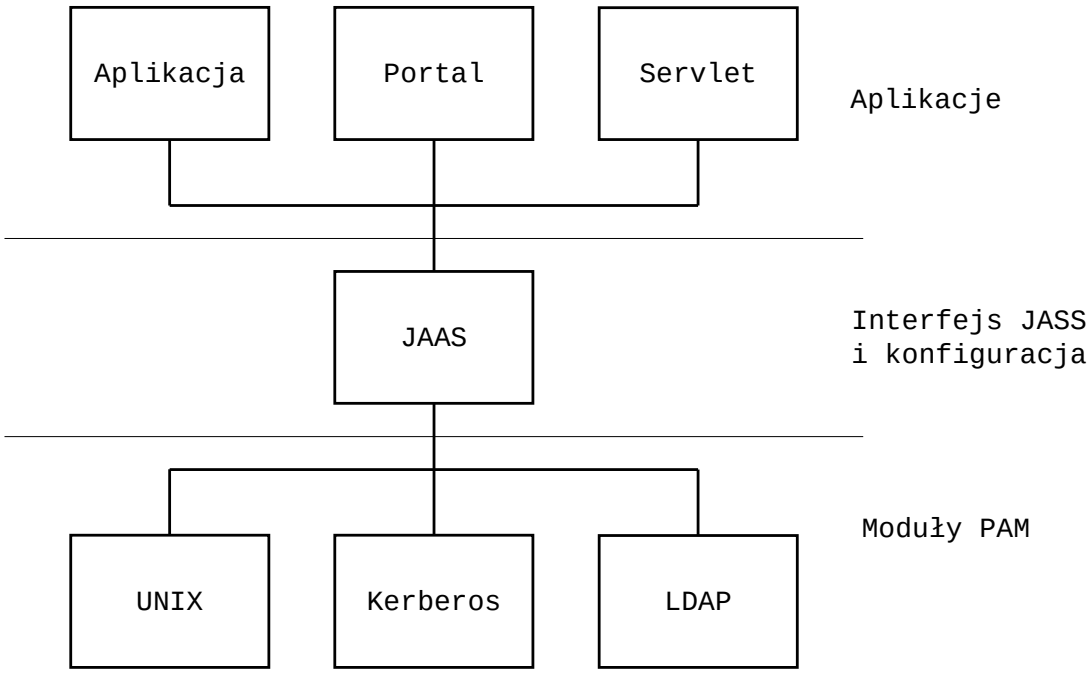
Drugą kwestią jest odpowiedni interfejs programistyczny dla dostępu do usług autoryzacji i autentykacji z programów, które dopiero będą tworzone lub modyfikowane. Dotyczy to w szczególności aplikacji bazujących na WWW (np. portali), ale również innych aplikacji przeprowadzających interakcje z użytkownikiem i wymagających z tego powodu identyfikacji i kontroli uprawnień użytkowników.

Zdecydowaliśmy się ograniczyć proponowane rozwiązanie do języka programowania Java. Do takiej decyzji skłoniła nas przede wszystkim rosnąca popularność tego języka w zastosowaniach WWW. Dla aplikacji pisanych w językach C/C++ i innych istnieje możliwość wykorzystania istniejących, darmowych interfejsów dostępu bezpośrednio do serwera LDAP (np. OpenLDAP, iPlanet) lub wykorzystanie innych interfejsów służących do autentykacji i autoryzacji (np. GSS-API, SASL), które mogą pośredniczyć pomiędzy aplikacją a serwerem LDAP.

4.1 Java Authentication and Authorization Service

Firma SUN opracowała dla języka Java interfejs "Java Authentication and Authorization Service" (JAAS), dostarczający usługi autentykacji i autoryzacji. Zdecydowaliśmy się wykorzystać istniejące rozwiązanie i ewentualnie opisać je dla potrzeb usług LDAP.

Architektura JAAS oparta jest o koncepcję modułów dostarczających usługi autentykacji PAM (ang. Pluggable Authentication Module). Dzięki temu możliwa jest zmiana sposobu autentykacji i autoryzacji poprzez zmianę wykorzystywanego modułu PAM bez potrzeby dokonywania zmian w aplikacji. Architektura jest przedstawiona na poniższym rysunku.



Firma SUN wraz oprogramowaniem JAAS dostarczają również moduły autentykacji i autoryzacji użytkowników korzystające z następujących metod:

- SUN Solaris,
- MS Windows NT,
- JNDI.

Moduł wykorzystujący JNDI może być użyty do podłączenia serwera LDAP.

5 Podsumowanie

Po przeprowadzeniu szeregu analiz należy stwierdzić, że większość dostępnych produktów zarówno darmowych jak i komercyjnych posiada odpowiednie moduły pozwalające na autentykację i autoryzację użytkowników poprzez serwer LDAP. W szczególności dotyczy to trzech przetestowanych i przedstawionych tu serwerów Apache, Tomcat i iPlanet. Dostępne dla tych serwerów rozwiązania dotyczące autentykacji i autoryzacji są częściowo zdeterminowane poprzez istniejące standardy protokołów HTTP, SSL czy interfejsu Servlet API. Niektóre kwestie nie ustalone w tych standardach są traktowane poprzez poszczególne moduły indywidualnie i wymagają najczęściej odpowiedniej konfiguracji.

Dla aplikacji, które nie posiadają dostępu do już istniejących modułów autoryzacji polecamy użycie interfejsu "Java Authentication and Authorization Service", który wraz z modułem JNDI pozwala na zapewnienie tych usług bazujących na serwerze LDAP.

Głównym obszarem, który wymagał sprecyzowania jest sposób budowy drzewa informacji LDAP oraz wymagane dla obiektów użytkowników klasy i atrybuty. Proponowane rozwiązanie zostało przedstawione w rozdziale drugim i powinno być uwzględnione w końcowych zaleceniach projektu dotyczących budowy bazy danych LDAP.

Bibliografia

[AHTTPD] "Apache HTTP Server Project", http://httpd.apache.org/

[ALDAPAUTH] Muhammad A. Muquit: "LDAP Authentication module for Apache", April 2001, http://www.muquit.com/muquit/software/mod_auth_ldap/mod_auth_ldap.html

[DEEP] P. Gietz, "Technical Project Proposal. Definition of an European Educational Person (DEEP)", May 2001, http://www.terena.nl/task-forces/tf-lsd/projects/DEEP-Projekt-proposal.rtf

[EDUCSE] "EDUCAUSE", http://www.educause.edu/about.html

[EDUPER] "eduPerson 1.0 Specification", February 2001, http://www.educause.edu/netatedu/groups/pki/eduperson/spec.pdf

[IDSADM] "iPlanet Directory Server Administrator's Guide", December 2001, http://docs.iplanet.com/docs/manuals/directory/51/pdf/iDS51admin.pdf

[IDSSCH] "iPlanet Directory Server Schema Reference", December 2001, http://docs.iplanet.com/docs/manuals/directory/51/pdf/iDS51schema.pdf

[INT2] "Internet2", http://www.internet2.edu/html/about.html

[LRECIPE] Michael R Gettes: "A Recipe for Configuring and Operating LDAP Directories", May 2000, http://www.georgetown.edu/giia/internet2/ldap-recipe/

[OPENLDAP] "OpenLDAP", http://www.openldap.org/

[RFC1274] P. Barker, S. Kille: "The COSINE and Internet X.500 Schema", November 1991, http://www.ietf.org/rfc/rfc1274.txt

[RFC2254] T. Howes: "The String Representation of LDAP Search Filters", December 1997, http://www.ietf.org/rfc/rfc2254.txt

[RFC2255] T. Howes, M. Smith: "The LDAP URL Format", December 1997, http://www.ietf.org/rfc/rfc2255.txt

[RFC2256] M. Wahl: "A Summary of the X.500(96) User Schema for use with LDAPv3", December 1997, http://www.ietf.org/rfc/rfc2256.txt

[RFC2798] M. Smith: "Definition of the inetOrgPerson LDAP Object Class", April 2000, http://www.ietf.org/rfc/rfc2798.txt

[SERVLET] "Java Servlet Technology", http://java.sun.com/products/servlet/index.html

[SERVLET23] "Java™ Servlet Specification Version 2.3", August 2001, http://www.jcp.org/aboutJava/communityprocess/final/jsr053/

[SHIB] "Shibboleth", http://middleware.internet2.edu/shibboleth/

[TF-LSD] "LDAP Services Deployment", http://www.terena.nl/task-forces/tf-lsd/