入門LDAP認証（２）

― 検索と認証 ―

平　野　　靖

Ⅰ．はじめに

１．前回の補足

今回の本題を説明する前に，まず前回の補足をしておく。前回は，“ OpenLDAPがほとんど唯一のフリーのLDAPサーバである ”と説明したが，Apache Projects[1]の下部のApache Directory Project[2]で，Pure JavaベースのLDAPサーバが開発されているようである。このLDAPサーバは，2003年10月30日にApache Projectsの一部として採用され，それまでLDAPdと呼ばれていたものをEveと改名したようである。EveはLDAPとX.500の実験用として設計されているとのことであるが，今後，実用段階まで開発が進むことを期待している。

２．お試しサーバのおさらい

前回はLDAPとはどういうものであるのかを説明した。今回は，JavaやPHPなどのプログラミング言語を用いてエントリを検索したり，認証に用いたりする方法を説明する。今回も，接続するLDAPサーバは情報連携基盤センターのお試しサーバである。

まずはお試しサーバのおさらいをしておく。詳細は2005年２月発行のセンターニュースを参照してほしい。

- ホスト名：pub-ldap.itc.nagoya-u.ac.jp
- ポート番号：1024
- 100人分の一般ユーザのエントリと２つの管理者用エントリを格納
- 匿名アクセス（anonymous access）ではuserPassword属性に対してはcompareのみ，その他の属性値に対してはreadとsearchが可能
- cn=LdapStaff,ou=Staff,o=LDAP-TESTからのアクセスではo=LDAP-TEST 全体に対してすべての属性値のread，search，compareが可能
- cn=DptStaff,ou=Staff,o=LDAP-TESTからのアクセスではou=Edo,o=LDAP-TEST以下のエントリだけはuserPassword属性も含めてすべての属性値のread，search，compareが可能，その他のou以下のエントリに対しては匿名アクセスと同等の権限
- 一般ユーザ用エントリのuserPasswordは，cnの先頭にpをつけたもの

1　http://www.apache.org/　WebサーバのApacheやServlet ContainerのTomcatなどで有名である。
2　http://incubator.apache.org/directory/

- cn=LdapStaff,ou=Staff,o=LDAP-TESTとcn=DptStaff,ou=Staff,o=LDAP-TESTのuserPasswordは，それぞれps00001とps00002

Ⅱ．検索と認証の基本

検索や認証などのためには，iPlanet Directory Server（以下，iDSと略す）やOpenLDAPなどに付属するldapsearchや，さまざまな言語のためのSDKを用いることができる。このようなSDKはJava，C，Perl，PHP，及びRubyなど一般的な言語に対して作成されているので，アプリケーション（LDAPクライアント）を作成する上でそれほどの困難はないだろう。

iDSやOpenLDAPなどに付属するコマンドやさまざまな言語のAPIを使って検索や認証を行う場合でも，手順はほとんど同じである。もちろん，iDSやOpenLDAPなどに付属するコマンドでは，細かな手続きが適切な順序で行われるように作られているが，APIを使って自分でLDAPクライアントを作成する場合にはこのような手続きに対応するAPIを適切に並べる必要がある。一般的な手順は下記のとおりである。

・セッションハンドルの取得（接続）
・バインド（bind）（認証）
・データ操作（属性の比較，検索など）
・アンバインド（unbind）（切断）

このうちバインドとは，どのエントリ（dn）の権限でデータ操作を行うかをLDAPサーバに通知することである。バインドには簡易認証とSASL（Simple Authentication and Security Layer）［1］［2］の２種類がある。簡易認証ではLDAPサーバにはdnと平文パスワードが送信される。一方，SASLを用いるとエントリ名とパスワードが暗号化されるため，セキュアなバインドを行うことができる。ただし，SASLを用いても暗号化されるのはバインドの過程だけで，バインドが終わったあとの通信（データの操作）は暗号化されない。

ここでは簡単のために，簡易認証について述べる。簡易認証には，バインド時にLDAPサーバに送信する情報の違いによって，下記の３種類に分けられる。いずれの場合もバインドが成功した段階で，バインドのために送信したエントリ名（Bind DN）に対応するACI（Access Control Instruction）が適用される。

匿名認証（Anonymous bind）：エントリ名もパスワードも送信しない。
非認証：エントリ名だけ送信する。結果的には匿名認証と同一となる。
エントリ名／パスワード認証：エントリ名とパスワードを送信する。

次節からは，実際のコードを示しつつ，認証や検索の方法を説明する。

Ⅲ．エントリの検索

LDAPサーバにはID（エントリ名）とパスワード以外の情報も格納することができる。ここではそれらの情報を検索する方法を紹介する。

１．ldapsearchの使い方

ldapsearchとはエントリの属性値を表示するためのコマンドであり，LDAPサーバに付属している。実際のアプリケーションを書く際にはほとんど役に立たないが，検索の可否を確認するという目的では有用である。LDAPサーバを提供するベンダーによってldapsearchの挙動やオプションが異なる。ここでは，OpenLDAPとiDSのldapsearchを例に説明をする。

以下，UNIXのコマンドプロンプトを“％”であらわす。また，“＼”はコマンドライン中で，つぎの行に続くことを示している。

OpenLDAPのldapsearch

OpenLDAPに付属するldapsearchの最も基本的な使い方は下記のとおりである。

```
% ldapsearch -x -w ps00001 -h pub-ldap.itc.nagoya-u.ac.jp -p 1024 \
  -b o=LDAP-TEST -D cn=LdapStaff,ou=Staff,o=LDAP-TEST "(objectclass=*)"
```

このコマンドを実行することにより，cn=LdapStaff,ou=Staff,o=LDAP-TESTが読むことができるエントリの属性が LDIF形式で出力される。どのエントリのどの属性を読むことができるかはACIで定義される。ldapsearchの主なオプションの意味を表１に示す。詳細は，man[3]やWebページ[4]を参照していただきたい。

`"(objectclass=*)"`はfilterであり，`objectclass`が“＊”に一致するエントリを検索対象にすることを示す。“＊”はメタキャラクタであり，すべての文字列に一致する。また，すべてのエントリは，なんらかの`objectclass`を持つ。したがって，filterを`"(objectclass=*)"`，BaseDNを`o=LDAP-TEST`と指定すると，o=LDAP-TEST以下のすべてのエントリが検索対象となる。なお，“"”はシェル（bsh，tcsh，zshなど）によって“＊”が展開されないようにするためである。パスワードやBaseDNなどに“＊”，“？”“＼”などの特殊文字が含まれる場合にも，“"”でくくる必要がある。

紙面の都合で実行結果は示さないが，上記のコマンドを実行すると，一般ユーザをあらわすエントリ以外に組織名（o）や部署名（ou）をあらわすエントリも表示されることが分かる。また，cn=LdapStaff,ou=Staff,o=LDAP-TESTはすべての情報を読み込むことができる特権的なエントリであるので，任意のエントリのパスワード（userPassword）も検索することができる。

`cn=LdapStaff,_ou=Staff,o=LDAP-TEST`のように，Base DNやBind DNの途中にスペースを入れるとエラーになるので，スペースを入れないで記述するか，Base DNやBind DNの全体を“"”でくくる。`-w`オプションでパスワードを指定すると，パスワードがそのままディスプレイに表示されてしまう。これが好ましくない状況では，-Wオプションを指定し，パスワード入力プロンプト（Enter LDAP Password:）を出力するようにする。なお，-wオプションも-Wオプションも指定しないと，-DオプションでBind DNを指定したとしても，匿名アクセスとして扱われる。

3　UNIXのコマンドラインで`man ldapsearch`と入力する

4　文献［3］の筆者のページ（http://www5f.biglobe.ne.jp/~inachi/openldap/man/man1/ldapsearch.1.html）や，OpenLDAPの本家のページ（http://www.openldap.org/software/man.cgi）など

匿名認証を行うには下記のコマンドを実行する。

```
% ldapsearch -x -h pub-ldap.itc.nagoya-u.ac.jp -p 1024 \
  -b o=LDAP-TEST "(objectclass=*)"
```

表１　OpenLDAP付属のldapsearchコマンドの主なオプション

オプション	意　味
-b basedn	指定したノード（Base DN）以下を検索対象とする
-LL	コメントなしのLDIF形式で結果を出力する
-LLL	コメントとバージョン番号なしのLDIF形式で結果を出力する
-S attr	“ attr ”で指定された属性でソートして出力する
-D binddn	バインドするエントリ（Bind DN）
-h host	LDAPサーバ名
-p port	ポート番号
-v	冗長モード（詳細モード）
-w passwd	バインドパスワード
-W	バインドパスワードの入力を促すプロンプトを出力する
-x	簡易認証（Simple authentication）で接続する。このオプションをつけないとSASL認証を試みる
-y file	“ file ”で指定されたファイルからパスワードを読み取る
-d level	“ level ”で指定されたdebugging level（０～255の数字で指定する）で実行する。debugging levelを２進数で表現したときにどの桁に１があるかによって表示される情報が異なる。デフォルトは０

Sun iPlanet Directory Server(iDS)のldapsearch

iDSのldapsearchでも，OpenLDAPの基本的なオプションの多くは共通である。以下，相違点を挙げる。

- -xオプションはなく，デフォルトで簡易認証を行う。
- -LLオプションは存在しない。-LLオプションを指定してもエラーにはならないが，出力される結果は-Lオプションを指定した際と同じである。
- -Wオプションは存在しない。
- -wオプションを指定しなくても，-Dオプションで指定されたBind DNでバインドを試みる。デフォルトでパスワード入力プロンプトを出力する。
- デフォルトでは属性型と属性値の区切り文字は“ = ”であるので，出力結果はLDIF形式ではない。LDIF形式で結果を得るには，-Lオプションを指定する必要がある。

また，-Lオプションや-Bオプションを指定しないと，漢字やひらがななど半角アルファベットではない文字列は“ NOT ASCII ”と出力される。UTF-8をBase64でエンコードして出力するためには-Lオプションを指定し，UTF-8のまま出力するには-Bオプションを指定する。ただし，-Bオプションをした場合には，UTF-8をサポートしない環境（UNIXのコンソールなど）では文字化けしたように表示される。

２．サンプルプログラム

ここでは，JavaとPHPによる検索プログラムを示す。プログラムの作成には，主に文献［３］やJLDAPのパッケージ中のサンプルプログラムを参考にした。本連載に掲載するほとんどすべてのプログラムなどはWebサーバに置いておくので，自由に使って欲しい。WebサーバのURLはhttp://pub-auth-web.itc.nagoya-u.ac.jpである。

Java（JNDI）

JNDIとは，Java Naming and Directory Interfaceの略でJavaが標準で提供するネーミングサービスのためのAPIである[5]。このAPIはLDAPを含めて，DNS，NIS（YP），及びX.500などの既存のネーミングサービスに対して統一されたインタフェースを提供する。

JNDIを用いて検索を行うためのサンプルプログラムを以下に示す。このプログラムでは，InitialDirContext(env)で接続とバインドを行い，ctx.close()でアンバインドを行っている。また，外側のwhile文はエントリが，内側のwhile文はそのエントリの属性があり続ける限り繰り返される。

―― ファイル名：ldapSearch.java ――

```
import javax.naming.*;
import javax.naming.directory.*;

import java.util.Hashtable;
import java.util.Enumeration;

public class ldapSearch {

    public static void main(String[] args) {

        if (args.length != 6 ){
            System.out.println("usage: ldapSearch host port BaseDN BindDN BindPW filter");
            System.exit(0);
        }

        String host   = args[0];
        String port   = args[1];
        String BaseDN = args[2];
        String BindDN = args[3];
        String BindPW = args[4];
        String Filter = args[5];
```

5　http://java.sun.com/j2ee/ja/jndi/

```
// 初期コンテキストのプロパティ設定
Hashtable env = new Hashtable();
env.put(Context.INITIAL_CONTEXT_FACTORY, "com.sun.jndi.ldap.LdapCtxFactory");
env.put(Context.PROVIDER_URL, "ldap://" + host + ":" + port );
env.put(javax.naming.Context.SECURITY_AUTHENTICATION, "simple");
env.put(javax.naming.Context.SECURITY_PRINCIPAL, BindDN );
env.put(javax.naming.Context.SECURITY_CREDENTIALS, BindPW);

try {
    DirContext ctx = new InitialDirContext(env); //初期コンテキスト作成

    // 検索スコープの指定
    SearchControls cons = new SearchControls();
    cons.setSearchScope(SearchControls.SUBTREE_SCOPE);

    NamingEnumeration res = ctx.search( BaseDN, Filter, cons);  // 検索の実行

    // 検索結果の読出し
    while (res.hasMore()) {
        SearchResult Entry = (SearchResult)res.next();
        System.out.println("dn: " + Entry.getName() + "," + BaseDN); // dnの表示

        // 属性集合の取り出し
        Attributes Attrs = Entry.getAttributes();
        if (Attrs == null) break;

        // 個々の属性の取り出し
        NamingEnumeration EnumAttrs = Attrs.getAll();
        while (EnumAttrs.hasMore()) {
            Attribute anAttr = (Attribute)EnumAttrs.next();

            Enumeration Vals = anAttr.getAll();  // 属性の取り出し
            while (Vals.hasMoreElements()) {
                System.out.println(anAttr.getID() + ": " //属性型の取り出し
                                   + Vals.nextElement()); //属性値の取り出し
            }
        }
        System.out.println();
    }
    ctx.close();
} catch (NamingException e) {
    System.out.println("JNDI Error: "+ e.toString());
}
```

```
    }
}
```

―― * ―― * ―― * ――

このプログラムを実行するには，上記のプログラムをldapSearch.javaというファイル名で保存した後，

```
% javac ldapSearch.java
```

と入力してプログラムをコンパイルした後，

```
% java ldapSearch pub-ldap.itc.nagoya-u.ac.jp 1024 \
  o=LDAP-TEST cn=LdapStaff,ou=Staff,o=LDAP-TEST ps00001 "cn=*"
```

と入力する。なお，ユーザの環境によっては，CLASSPATHやPATHなどの環境変数を変更する必要があるかもしれない。

Java（JLDAP）

つぎに，JLDAP（LDAP Class Libraries for Java）によるサンプルプログラムを示す。JLDAPとは，IETF（Internet Engineering Task Force）[6]が標準化を進めているAPIである。JNDIは各種のネーミングサービスのための汎用APIであるのに対し，JLDAPはLDAPに特化している。JLDAPをインストールするには，http://developer.novell.com/ndk/jldap.htmから使用するOSに適したLDAP Class Libraryをダウンロードする[7]か，下記のコマンドでOpenLDAPのCVSリポジトリから入手する。

```
% setenv CVSROOT :pserver:anonymous@cvs.OpenLDAP.org:/repo/OpenLDAP
% cvs login
Logging in to :pserver:anonymous@cvs.openldap.org:2401/repo/OpenLDAP
CVS password:（「OpenLDAP」と入力する）
% cvs -z3 checkout -P jldap
% cvs logout
```

これらのコマンドを入力することにより，カレントディレクトリにjldapというディレクトリが作成され，その下にソースファイルが展開される。

JLDAPでは属性値が漢字やバイナリ形式であるかを判断したり，Base64エンコード・デコードしたりするためのクラス（com.novell.ldap.util.Base64）が用意されている[8]ので，このクラスを使うことにより，漢字やひらがななどのアルファベットではない文字やuserPassword属性の値など

6　インターネットに関する仕様等の標準化活動を行っている組織。http://www.ietf.org/

7　http://developer.novell.com/ndk/downloadaz.htmからはC＃言語やC用のLDAPのLibraryのほか，NetWareやNovell eDirectoryのためのさまざまなLibraryを入手することができる。また，http://developer.novell.com/ndk/doc/samplecode/jldap_sample/からはさまざまなサンプルプログラムをダウンロードすることができる。

8　このほかにorg.apache.xerces.impl.dv.util.Base64というクラスもあるようである。

をBase64でエンコードしてから表示することができる。

JNDIを用いたプログラムとほぼ同様のものを下記に示す。JLDAPではLDAPサーバとの通信の各ステップが一つ一つのメソッドに対応するので下記のプログラムを理解するのは容易であろう。

── ファイル名：ldapSearch.java ──

```
import com.novell.ldap.LDAPAttribute;
import com.novell.ldap.LDAPAttributeSet;
import com.novell.ldap.LDAPConnection;
import com.novell.ldap.LDAPEntry;
import com.novell.ldap.LDAPException;
import com.novell.ldap.LDAPSearchResults;
import com.novell.ldap.util.Base64;
import java.util.Enumeration;
import java.util.Iterator;
import java.io.UnsupportedEncodingException;

public class ldapSearch
{
   public static void main(String[] args)
   {

      if (args.length != 6 ){
         System.out.println("usage: ldapSearch host port BaseDN BindDN BindPW filter");
         System.exit(0);
      }

      String host   = args[0];
      int    port   = Integer.parseInt(args[1]);
      String BaseDN = args[2];
      String BindDN = args[3];
      String BindPW = args[4];
      String Filter = args[5];
      LDAPConnection ld = new LDAPConnection();

      try {
         ld.connect(host, port);  // LDAPサーバへ接続
         ld.bind(LDAPConnection.LDAP_V3, BindDN, BindPW.getBytes("UTF8"));  // バインド

         LDAPSearchResults searchResults =
            ld.search(BaseDN, LDAPConnection.SCOPE_SUB, Filter, null, false);  // 検索

         while (searchResults.hasMore()) {
```

```
            LDAPEntry Entry = null;
            try {
                Entry = searchResults.next();
            }
            catch(LDAPException e) {
                System.out.println("Error: " + e.toString());
                continue;
            }

            System.out.println("dn: " + Entry.getDN()); // dnの表示
            LDAPAttributeSet attrSet = Entry.getAttributeSet(); // 属性集合の取り出し
            Iterator allAttr = attrSet.iterator();

            while(allAttr.hasNext()) {
                LDAPAttribute attr = (LDAPAttribute)allAttr.next();  // 属性の取り出し
                String attrName = attr.getName();  // 属性型の取り出し
                Enumeration allValues = attr.getStringValues();  // 属性値の取り出し

                if( allValues != null) {
                    while(allValues.hasMoreElements()) {
                        String Value = (String) allValues.nextElement();
                        // 表示可能な場合
                        if (Base64.isLDIFSafe(Value)) {
                            System.out.println(attrName + ": " + Value);
                        }
                        // 表示不可能な場合（Base64でエンコードしてから表示）
                        else {
                            Value = Base64.encode(Value.getBytes());
                            System.out.println(attrName + ": " + Value);
                        }
                    }
                }
            }
            System.out.println("");
        }
        ld.disconnect();
    }
    catch( LDAPException e ) {
        System.out.println( "Error: " + e.toString() );
    }
    catch( UnsupportedEncodingException e ) {
        System.out.println( "Error: " + e.toString() );
    }
```

```
        System.exit(0);
    }
}
```

―― * ―― * ―― * ――

このプログラムもJNDIの項で説明した方法によってコンパイル，及び実行することができる。

PHP

Webアプリケーションを作る際にはPHPを用いることが少なくない。そこで，つぎにPHPでLDAPサーバにアクセスする方法を説明する。用いたPHPのバージョンは4.2.2である。PHPでLDAPサーバにアクセスするには，まず，OpenLDAPかNetscapeのSDKを入手し，コンパイルする。つぎに設定オプションとして-with-ldap=[LDAPクライアントライブラリがあるディレクトリ]を指定してPHPをコンパイルする。

下記に示すのはPHPスクリプトにBind DNやBase DNなどのパラメータを渡すためのhtmlファイルである。Webブラウザでhttp://pub-auth-web.itc.nagoya-u.ac.jp/php/ldapSearch.htmlにアクセスすることにより，図１の検索画面を表示することができる。

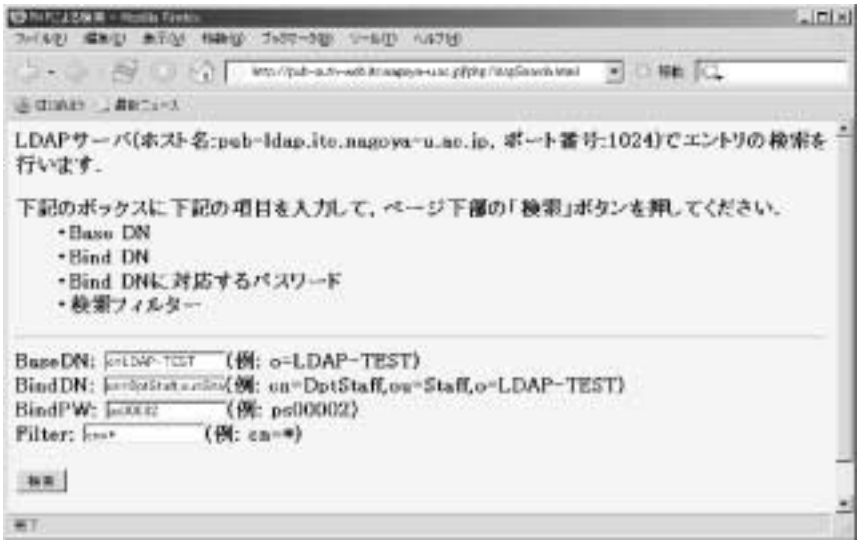

図１　PHPによる検索の例

―― ファイル名：ldapSearch.html ――

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=EUC-JP">
<title>PHPによる検索</title>
</head>
```

```
<body bgcolor=#a0ffff>
<h3>
LDAPサーバ（ホスト名：pub-ldap.itc.nagoya-u.ac.jp，ポート番号：1024）でエントリの検索を行います。
<br>
<dl>
<dt>下記のボックスに下記の項目を入力して，ページ下部の「検索」ボタンを押してください。<br>
<dd>・Base DN
<dd>・Bind DN
<dd>・Bind DNに対応するパスワード
<dd>・検索フィルタ
</dl>

<hr>
<form action="ldapSearch.php" method="post">
BaseDN: <input type="text" name="BaseDN">(例: o=LDAP-TEST)</div><br>
BindDN: <input type="text" name="BindDN">(例: cn=DptStaff,ou=Staff,o=LDAP-TEST)</div><br>
BindPW: <input type="text" name="BindPW">(例: ps00002)</div><br>
Filter: <input type="text" name="Filter">(例: cn=*)</div><br>
<br>
<input type="submit" name="submit" value="検索">
</form>
</h2>
<hr>
<a href=/>Top Pageに戻る</a>
</body>
```

―――＊―――＊―――＊―――

つぎに，上記のhtmlファイルから値を渡され，実際に検索を行うPHPスクリプトを示す。検索結果がUTF-8で出力されるので，PHPスクリプト自体もUTF-8で記述する必要がある。検索結果を図２に示す。

```
<?php
print "<html>";
print "<head>";
print "<meta http-equiv=\"Content-Type\" content=\"text/html; charset=UTF-8\">";
print "<title>検索結果</title>";
print "</head>";
print "<body bgcolor=#a0a0ff>";

$host = "pub-ldap.itc.nagoya-u.ac.jp";
$port = 1024;
$basedn = $_POST['BaseDN'];
$binddn = $_POST['BindDN'];
```

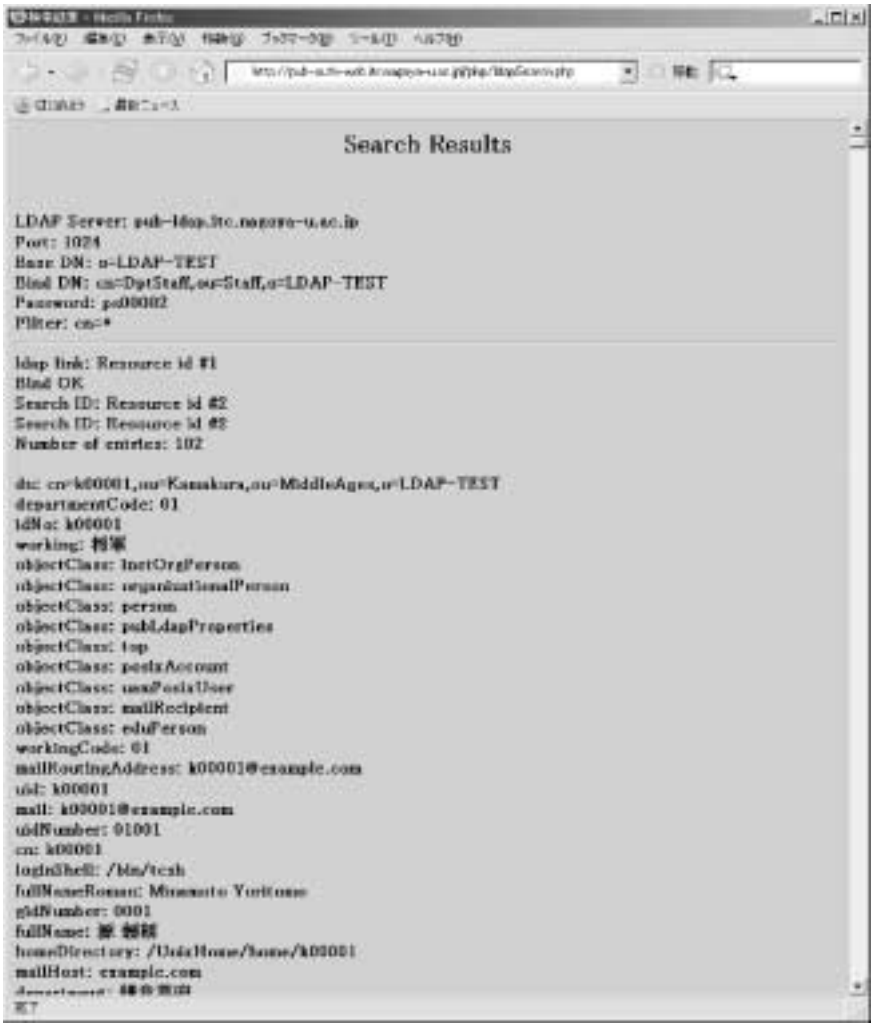

図２　検索結果の例

―― ファイル名：ldapSearch.php ――

```
$bindpw = $_POST['BindPW'];
$filter = $_POST['Filter'];

print "<h2>";
print "<center>Search Results</center><br>";
print "<h4>";
print "LDAP Server: ".$host."<br>";
print "Port: ".$port."<br>";
print "Base DN: ".$basedn."<br>";
print "Bind DN: ".$binddn."<br>";
print "Password: ".$bindpw."<br>";
```

```
print "Filter: ".$filter."<br>";
print "<hr>";

$ld = ldap_connect($host, $port);
print "ldap link: ".$ld."<br>";

ldap_set_option($ld, LDAP_OPT_PROTOCOL_VERSION, 3);

$bi = ldap_bind($ld, $binddn, $bindpw);
if($bi==FALSE){
  print "<font color=green><blink>Invalid credentials</blink><blink></font><br><hr>";
  print "<a href=ldapSearch.html>Back to Search page</a>";
  exit(0);
}
print "Bind OK";
if($bindpw==""){
  print "<font color=green><blink> (Anonymous Bind)</blink><blink></font>";
}
else{
   print "(Bind by dn: ".$binddn.")";
}
print "<br>";

$res = ldap_search($ld, $basedn, $filter);
if($res==FALSE){
  print "<font color=green><blink>Invalid Base DN</blink></font><br><hr>";
  print "<a href=ldapSearch.html>Back to Search page</a>";
  exit(0);
}
print "Search ID: ".$res."<br>";

$num = ldap_count_entries($ld, $res);
print "Number of entries: ".$num."<br>";
print "<hr>";

for ($entry = ldap_first_entry($ld, $res); $entry;
     $entry = ldap_next_entry($ld, $entry)) {

  $dn = ldap_get_dn($ld, $entry);
  echo "dn: ".$dn."<br>";

  for ($attr = ldap_first_attribute($ld, $entry, $ber);
       $attr;
```

```
      $attr = ldap_next_attribute($ld, $entry, $ber)) {

    $values = ldap_get_values($ld, $entry, $attr);
    for ($i = 0; $values[$i]; $i++) {
      echo $attr.": ".$values[$i]."<br>";
    }
  }
  echo "<br>";
}
ldap_unbind($ld);
echo "<hr>";
print "<a href=ldapSearch.html>Back to Search page</a>";
?>
```

―― * ―― * ―― * ――

その他の言語

サンプルコードを示した言語以外にもさまざまな言語用のSDKが用意されている。これらについてはSDKを入手するためのURLのみを示す。下記に示したもの以外にもさまざまな言語のためのSDKが見つかるかもしれない。

- C言語: http://www.openldap.org/software/download/（OpenLDAPをインストールすることにより，IETFのInternet-Draftの内容にしたがったC LDAP APIを利用できる）
- Ruby/LDAP: http://ruby-ldap.sourceforge.net/
- Perl: Perl-LDAP…http://ldap.perl.org/，PADL…http://www.padl.com/
  （http://search.cpan.org/でPerlのモジュールを検索することができる）

Ⅳ．認証

１．認証の方法

認証とは，あるユーザが本人であるかを確認する作業であり，通常はIDとパスワードの組が認証システムにあらかじめ登録されているものと一致するか否かを確かめることによって行われる。

以下，３つの認証方法について考える。いずれもユーザがcnとパスワードを入力し，LDAPサーバに格納された情報を基にユーザの認証を行うことを想定する。なお，以下では，情報サービスプロバイダとは部局・全学の学生・職員に対して情報サービスを提供する主体，管理者とは情報サービスプロバイダの管理者，及びユーザとはその情報サービスを提供される者，という意味で使う。また，方法２，３は，まずクライアント（１台１台のホストではなく，例えば，計算機室の１台の管理マシンなど）ごとに管理者用の特別エントリを作成しておく。

方法１：匿名認証による方法

１．匿名認証でバインドし，cnをキーとしてそのcnを含むdnを検索。

２．見つかったdnと入力されたパスワードを使って再度バインド。

3．バインドが成功したら，ユーザは認証されたとする。

方法２：管理者用のエントリでバインドする方法（１）

1．管理者用のエントリでバインドし，cnをキーとしてそのcnを含むdnを検索。

2．見つかったdnと入力されたパスワードで再度バインド。

3．バインドが成功したら，ユーザは認証されたとする。

方法３：管理者用のエントリでバインドする方法（２）

1．管理者用のエントリでバインドし，cnをキーとしてそのcnを含むdnを検索。

2．管理者用のエントリの権限で見つかったdnの暗号化されたパスワードをLDAPサーバから取得。

3．クライアント側で入力されたパスワードを暗号化し，LDAPサーバから取得したパスワードと比較。

4．２つのパスワードが一致したら，ユーザは認証されたとする。

前回の説明のように，一般的にはcnが同一である複数のdnが存在する可能性がある。したがって，この場合，上記の手順で複数のdnが見つかったときにはもう少し考慮の必要がある。しかし，全学IDでは同一のcnを持つdnが複数存在することはない。

それぞれの方法の長所と短所を見ていく。

方法１：匿名認証による方法

特徴：各ユーザがLDAPサーバにバインドする。パスワードの比較はLDAPサーバで行う。

長所：管理者用のエントリを用意するなどの設定が不要。一般的な設定方法。

短所：匿名認証を許可しているので，誰がアクセスしてくるか分からない。少なくとも，不特定の人にdnが漏洩する可能性があり，ブルートフォースアタックやネットワークの盗聴などにより不正アクセスされる可能性がある。これにより，LDAPサーバを利用しているクライアントがクラッキング[9]される危険性がある。ただし，iDSやOpenLDAP+TCP Wrappersなどを使えば，特定のホストからのみの匿名認証を許可することができるため，比較的安全である。しかし，SASLやldaps，VPNなどを使わないと，一般ユーザの平文パスワードがネットワークに流れる。

方法２：管理者用のエントリでバインドする方法（１）

特徴：各ユーザはLDAPサーバにバインドしない。パスワードの比較はLDAPサーバで行う。

長所：不特定のホストからアクセスされ，dnが漏洩する可能性が少ない。

短所：管理者，及び一般ユーザの平文パスワードがネットワークに流れてしまう。

方法３：管理者用のエントリでバインドする方法（２）

特徴：各ユーザはLDAPサーバにバインドしない。パスワードの比較はクライアントで行う。

長所：一般ユーザの平文パスワードはネットワークに流れない。管理者用のエントリの平文パ

9　一般にはハッキングと言われるが，ハッキングとは「プログラムやシステムを解析すること」であって，必ずしも悪とは限らない。一方，クラッキングは「システムに侵入して情報を盗んだり，システムそのものを破壊したりすること」である。

スワードはネットワークに流れるが，そのエントリに対応するクライアント以外のマシンからのアクセスは異常なので，攻撃を発見しやすい。iDSでは，個々のエントリに対してアクセス元のホストを指定できるので，管理者用のエントリの平文パスワードが漏洩しても他のホストからのアクセスは不可能である。

短所：一般的ではない。管理者用のエントリの平文パスワードが流れる。

iDSを使えば，方法３でも安全ではあるが，SSL/StartTLS，ldaps，VPN，あるいはSSH port forwardingを用いて通信路を暗号化すればさらに安全である。

２．パスワードの暗号化方式

多くのLDAPサーバではuserPassword属性の値を平文のまま格納するのではなく，なんらかの方法で暗号化している。暗号化方式としては，LDAPサーバソフトによって異なるが，CRYPT，MD5，SMD5，SHA，SSHA，及び暗号化なしなどが選べる。情報連携基盤センターが運用するLDAPサーバでは，これらの暗号化方式のうち，もっとも安全であるといわれるSSHA（Salted Secure Hash Algorithm）で暗号化（正確にはハッシュ）している[10]。SSHAとは，一方向ハッシュ関数を用いた変換方式の一種であり，変換された結果から元の情報（パスワードなど）を復元することは不可能である。さらに，SSHAでは情報の最後にsaltと呼ばれる任意のビット列を付加することにより，辞書攻撃に対する耐性を高めている。そのため，正しいパスワードを入力してもsaltが異なればSSHAで変換した結果は異なる。

３．サンプルプログラム

ここでは，方法３（管理者用のエントリでバインドする方法（２））でユーザを認証するプログラムのサンプルを示す。サンプルプログラムではLDAPサーバから，SSHAで暗号化され，さらにBase64でエンコードされたuserPasswordが送信されてくると仮定した。

SSHAによる暗号化では，平文パスワードの末尾に８バイトのsaltを付加した後，SHA-1で暗号化し，その結果得られた20バイトのビット列の末尾にsaltを付加する[11]。したがって，認証においては，LDAPサーバから読み出してきたuserPasswordからsaltを取り出し，これを用いてユーザが入力したuserPasswordにSSHAを適用しなければならない。サンプルプログラムの流れを以下に示す。

１．LDAPサーバから暗号化されたパスワード（PasswordInLDAP）を読み込む
２．PasswordInLDAPをBase64デコードする（decodedPassword）
３．decodedPasswordの後ろの８バイトを切り出す。これがsaltである
４．ユーザが入力したパスワード（TargetPW）の最後にsaltを連結する
５．encyptedPasswordにSHA-1でハッシュをかける

10　一部にSHAで暗号化されているエントリも存在する
11　http://www.cafesoft.com/CSDigest/showDigest.doに分かりやすい説明がある

6．得られたものの最後にsaltを連結する

7．Base64エンコードする

8．PasswordInLDAPと７で得られたものを比較し，一致すれば認証できたとする

上記の手順で認証するためのサンプルプログラムを下記に示す。

── ファイル名：auth.java ──

```
import com.novell.ldap.LDAPAttribute;
import com.novell.ldap.LDAPAttributeSet;
import com.novell.ldap.LDAPConnection;
import com.novell.ldap.LDAPEntry;
import com.novell.ldap.LDAPException;
import com.novell.ldap.LDAPSearchResults;
import com.novell.ldap.util.Base64;
import com.novell.ldap.LDAPExtendedOperation;
import com.novell.ldap.LDAPExtendedResponse;
import java.io.UnsupportedEncodingException;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
import java.util.*;

public class auth
{
   public static void main( String[] args )
   {

      if (args.length != 5 ){
         System.out.println("usage: auth BaseDN BindDN BindPW TargetCN TargetPW");
         System.exit(0);
      }

      String host   = "pub-ldap.itc.nagoya-u.ac.jp";
      int    port   = 1024;

      String BaseDN   = args[0];
      String BindDN   = args[1];
      String BindPW   = args[2];
      String TargetCN = args[3];
      String TargetPW = args[4];

      LDAPConnection ld = new LDAPConnection();

      try {
```

```
ld.connect(host, port);            //connect to the server
ld.bind(LDAPConnection.LDAP_V3, BindDN, BindPW.getBytes("UTF8"));
LDAPSearchResults searchResults =
   ld.search(BaseDN, LDAPConnection.SCOPE_SUB, TargetCN, null, false);

while (searchResults.hasMore()) {
   LDAPEntry nextEntry = null;
   try {
      nextEntry = searchResults.next();
   }
   catch(LDAPException e) {
      System.out.println("Error: " + e.toString());
      continue;
   }

   System.out.println("dn: " + nextEntry.getDN());

   LDAPAttributeSet attributeSet = nextEntry.getAttributeSet();
   Iterator allAttributes = attributeSet.iterator();

   LDAPAttribute fullNameRoman = attributeSet.getAttribute("fullNameRoman");

   LDAPAttribute userPassword = attributeSet.getAttribute("userPassword");
   String PasswordInLDAP = userPassword.getStringValue();
   PasswordInLDAP = PasswordInLDAP.substring(6, 46);

   byte[] decodedPassword = Base64.decode(PasswordInLDAP);
   byte[] salt = new byte[8];
   System.arraycopy(decodedPassword, 20, salt, 0, 8);

   byte[] TargetPWbin = new byte[TargetPW.length() + salt.length];
   System.arraycopy(TargetPW.getBytes(), 0, TargetPWbin, 0, TargetPW.length());
   System.arraycopy(salt, 0, TargetPWbin, TargetPW.length(), salt.length);

   try {
      MessageDigest md=MessageDigest.getInstance("SHA");
      md.update(TargetPWbin);
      byte[] encyptedPassword = md.digest();
      byte[] PWandSalt = new byte[encyptedPassword.length + salt.length];

      System.arraycopy(encyptedPassword, 0, PWandSalt, 0, encyptedPassword.length);
      System.arraycopy(salt, 0, PWandSalt, encyptedPassword.length, salt.length);
      String PasswordToCheck = Base64.encode(PWandSalt);
```

```
                if (PasswordInLDAP.equals(PasswordToCheck)) {
                    System.out.println("Password for " + fullNameRoman.getStringValue()
                                    + " is valid.");
                }
                else {
                    System.out.println("Password for " + fullNameRoman.getStringValue()
                                    + " is invalid.");
                }
                System.out.println("{SSHA}" + PasswordInLDAP  + "   [LDAP]");
                System.out.println("{SSHA}" + PasswordToCheck + "   [USER]");

            }
            catch (NoSuchAlgorithmException e) {
                e.printStackTrace();
            }
         }
         ld.disconnect();
      }
      catch (LDAPException e) {
         System.out.println( "Error: " + e. toString() );
      }
      catch (UnsupportedEncodingException e) {
         System.out.println("Error: " + e.toString());
      }
      System.exit(0);
   }
}
```

―― * ―― * ―― * ――

このプログラムでは，対象とするDITを第１引数で指定し，一般ユーザのuserPasswordを読み取れる権限を持つエントリのdnとそれに対応するuserPasswordをそれぞれ第２，第３引数に，認証の対象となるエントリのdnとそれに対応するuserPasswordを第４，第５引数に指定する。

正しいuserPasswordを入力すると，下記のような出力が得られ，認証が成功していることが分かる。

```
% java auth o=ldap-test cn=ldapstaff,ou=staff,o=ldap-test ps00001 \
  cn=e00001 pe00001
dn: cn=e00001,ou=Edo,o=LDAP-TEST
Password for Tokugawa Ieyasu is valid.
{SSHA}KyTPouHDohrf6NSxhT3z8F7dsyDSTwlhJSfRfg==   [LDAP]
```

```
{SSHA}KyTPouHDohrf6NSxhT3z8F7dsyDSTwlhJSfRfg==   [USER]
```

一方，不正なuserPasswordを入力した場合には，下記の出力を得る。

```
% java auth o=ldap-test cn=ldapstaff,ou=staff,o=ldap-test ps00001 \
  cn=e=00001 pe00002
dn: cn=e00001,ou=Edo,o=LDAP-TEST
Password for Tokugawa Ieyasu is invalid.
{SSHA}KyTPouHDohrf6NSxhT3z8F7dsyDSTwlhJSfRfg==   [LDAP]
{SSHA}kqncQYkBKTs5FznmmSfIoaA05V/STwlhJSfRfg==   [USER]
```

Ⅴ．おわりに

今回は簡易認証によるエントリの検索と認証の方法を説明した。次回は通信路の暗号化による安全なアクセス方法の説明をする。

なお，情報連携基盤センターでは，Webベースの情報サービスに対してCAS[12]による認証方法を提供することを考えている。CASを用いることにより，ユーザは一度CASに対するログインを行うと，CASに対応している情報サービスに再度ログインする必要がなくなる。これまでも名古屋大学ポータル[13]でSingle Sign-On(SSO)を実現していくつかのサービスを提供してきたが，これはあくまで名古屋大学ポータルを通じてサービスを提供する場合に限られていた。部局ごと，あるいは全学的に提供される情報サービスがCASに対応することにより，各情報サービスプロバイダは他のプロバイダを意識することなく，SSOを実現することができる。現在，CASの利用は教務システムにおける成績投入（2005年２月）・履修申請（2005年３月），及び名古屋大学ポータルでのユーザ認証に用いられている。Webベースのアプリケーションで全学IDを使う場合にはCASの方が便利であるし，安全性も高いため，情報連携基盤センターとしては，Webベースの情報サービスに対してはCASを推奨していく。機会があればCASの解説記事も掲載する予定である。

参考文献

［１］http://www.ipa.go.jp/security/rfc/RFC2222EN.html

［２］http://www.ipa.go.jp/security/rfc/RFC2222JA.html（文献［１］の和訳）

［３］稲地　稔：“ OpenLDAP入門－オープンソースではじめるディレクトリサービス－ ”，技術評論社，東京，2003

（ひらの　やすし：名古屋大学情報連携基盤センター大規模計算支援環境研究部門）

12　http://tp.its.yale.edu/tiki/tiki-index.php?page=CentralAuthenticationService.　ただし，情報連携基盤センターで用いているCASはYale大学で作られたものを改良し，より密接にLDAPサーバと連携できるようにしてある。

13　https://mynu.jp