IBM Security Access Manager for Web: Importing Users from LDAP

White Paper

May 2013

Ori Pomerantz
(orip@us.ibm.com)

Security Intelligence.
Think Integrated.

Smarter security solutions from IBM

Table of contents

Introduction

IBM® Security Access Manager for Web (ISAM) often shares an LDAP user repository with other applications. In those situations, it is useful to automatically import a list of the existing LDAP users into ISAM.

Audience

Administrators and implementers who implement ISAM to existing environments.

Acknowledgments

I would like to thank Windayani Achmad Zaenullah for identifying the need for this white paper and Lance Clinton for reviewing it. Any remaining mistakes are my fault.

1 Background

IBM’s security products are often implemented as ***brown field development***. This term means that there are already applications in the environment, possibly with thousands of users.

It is common to implement IBM Security Access Manager for Web (ISAM) into an environment that already has a directory server. When that happens, the same directory server user entries are typically used for IBM Security Access Manager users and some of the backend applications.

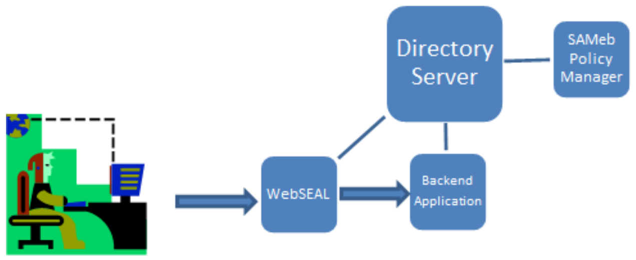

Figure 1: Directory server used by ISAM components and backend applications

The ISAM command line user interface, **pdadmin**, contains a command to import a user from LDAP. In this white paper, you learn a script that finds all the new users under a branch in the LDAP tree and then imports them automatically.

2 Script

This section explains the script. If you are not interested in the details and just want to use the script, read only Section 2.2, Settings, to learn how to modify the settings.

Because the script relies on the IBM Security Access Manager **pdadmin** command, it must run on a computer with the IBM Security Access Manager runtime environment.

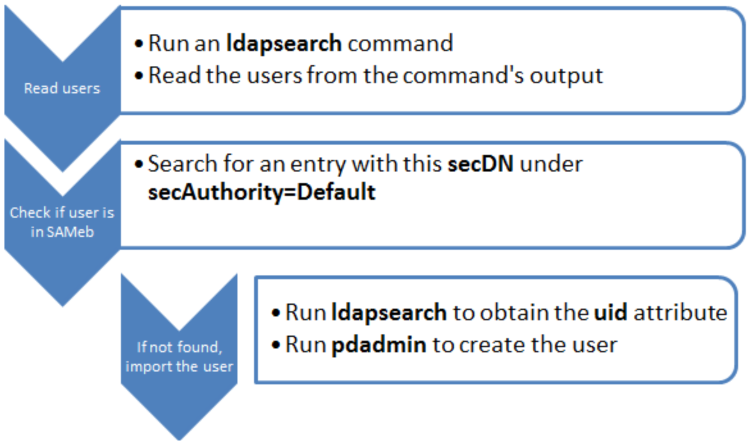
Read users
• Run an **ldapsearch** command
• Read the users from the command's output
Check if user is in SAMeb
• Search for an entry with this **secDN** under **secAuthority=Default**
If not found, import the user
• Run **ldapsearch** to obtain the **uid** attribute
• Run **pdadmin** to create the user

Figure 2: The major steps of the script

2.1 Python

Python is a common scripting language. It is a standard part of most Linux distributions. If your installation of Security Access Manager is running on Windows, you can download Python from this web address:

http://python.org/download/

There are many resources for you to learn Python. One example is Udacity, at the following web address:

https://www.udacity.com/course/cs101

2.2 Settings

The first part of the script includes the configuration.

```
#! /usr/bin/python
```

Normally, Python lines that start with a number sign (#) are comments. This line, however, is a special case. The UNIX convention is that files that start with a number sign followed by an exclamation point (#!) use the rest of the line as the name of the interpreter that runs the file. In this case, this code is the path for the Python interpreter. Modify it as needed.

```
# Authentication
ldap_dn = "cn=root"
ldap_pwd = "object00"
ldap_server = "localhost"
```

This code is the information (log on, password, and host name of server) for the LDAP server. Modify as needed.

```
sameb_uid = "sec_master"
sameb_pwd = "object00"
```

This code is the information (log on and password) for IBM Security Access Manager. Modify as needed.

```
# The part of the LDAP tree where you expect users
ldap_base = "ou=people,o=xyz"
```

Figure 3 and the preceding code show the location of the users in the LDAP tree. If there are multiple locations, either modify the script or use multiple copies, one for each location.

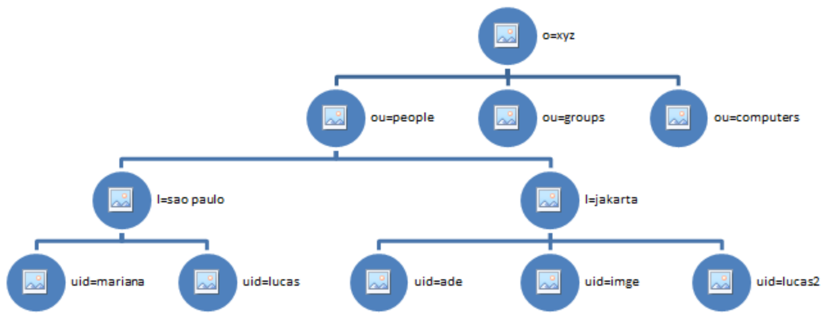

Figure 3: LDAP tree with users under ou=people,o=xyz

```
# The part of the LDAP tree where the SAMeb
# configuration is located
ldap_sameb_base = "secAuthority=Default"
```

In addition to locating users, the script must limit itself to those users who are *not* already in IBM Security Access Manager. To do so, it must know the location in the LDAP tree of the ISAM configuration. The default is **secAuthority=Default**, but you can change it when you first configure the policy server.

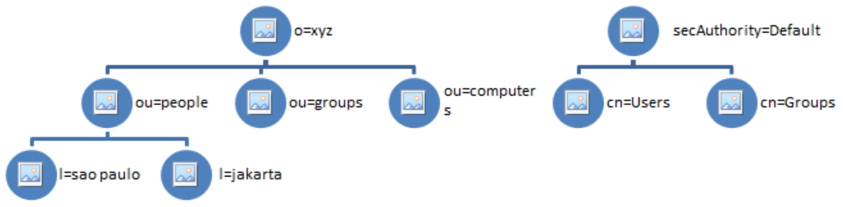

Figure 4: LDAP tree with secAuthority=Default

```
# Internal configuration
ldap_user_file = "ldap_users"
uid_file = "uid_file"
```

These are the files used as intermediate storage for the output of the commands used to obtain user information. Unless you have a reason to run multiple instances of this script at the same time, there is no reason to change these file names.

2.3 Obtaining the user list

The first actual step in the process is to obtain the user list from LDAP.

```
from subprocess import call
```

The script uses the **subprocess.call** function to run shell commands. This line imports the function so that it is available.

```
call("ldapsearch -LLL -D " + ldap_dn + " -w " + ldap_pwd
      + " -b " + ldap_base + " -h " + ldap_server
      " objectclass=inetOrgPerson dn >" + ldap_user_file,
          shell=True)
```

This code is the first call. With the default configuration, it runs this shell command:

```
ldapsearch -LLL -D cn=root -w object00 -b ou=people,o=xyz
-h localhost objectclass=inetOrgPerson dn > ldapusers
```

In other words, the first call searches for entities of type **inetOrgPerson**, which is used to represent people. It reads only their distinguished name and sends the output to a file called **ldapusers**.

```
ldap_users = open(ldap_user_file, "r")

for line in ldap_users:
```

These lines open the **ldapusers** file and read it, line by line. Python handles blocks by indentation; so everything that is indented at least once from this point is done once per user.

```
if len(line) > 1:
    dn = line[4:-1]
```

Half the lines in **ldapusers** are blank. The other half starts with **dn:**. These lines ignore the blank lines and discard the first four characters of the other lines.

Because Python blocks are designated by indentation, the rest of the script runs only for lines that are not blank.

2.4 Checking if users have ISAM accounts

ISAM user entries have a **secDN** attribute with the value of the original distinguished name. This code segment uses **ldapsearch** to look for an entry with the appropriate **secDN** value.

```
ret_val = call("ldapsearch -D " + ldap_dn +
    " -w " + ldap_pwd +
    " -h " + ldap_server +
    " -b " + ldap_sameb_base + " secDN=" + dn +
    " | grep numEntries > /dev/null",
    shell=True)
if (ret_val == 1):
    print "Need to import " + dn
```

If the return value is 1, **grep** failed. This failure happens when no LDAP entry matches the filter. In that case, the script must import the user.

2.5 Obtaining the user identifier

To create a new user, you must specify the user's UID, the identifier that they use to log on. These lines retrieve that value.

```
call("ldapsearch -D " + ldap_dn +
     " -w " + ldap_pwd +
     " -h " + ldap_server +
     " -b " + dn +
     " objectClass=inetOrgPerson uid " +
     "| grep uid: >" + uid_file,
     shell=True)
```

These lines show an **ldapsearch** for the attribute value.

```
uid_f = open(uid_file, "r")
uid = uid_f.read()
uid_f.close()
```

This line reads the first line of the file.

```
uid = uid[5:-1]
```

The line specifies to discard the first five characters (**uid**:); the remainder is the actual attribute value.

2.6 Creating accounts

Finally, these lines create the actual IBM Security Access Manager account using **pdadmin**.

```
call("pdadmin -a " + sameb_uid + " -p " +
     sameb_pwd + " user import " + uid +
     " " + dn,
     shell=True);
call("pdadmin -a " + sameb_uid + " -p " +
     sameb_pwd + " user modify " + uid +
     "account-valid yes ", shell=True)
```

It is necessary to issue the second command because users created by **pdadmin** are inactive by default.

Appendix A Full source code

This code is the full source of the script.

```
#! /usr/bin/python

# Authentication
ldap_dn = "cn=root"
ldap_pwd = "object00"
ldap_server = "localhost"
sameb_uid = "sec_master"
sameb_pwd = "object00"

# The part of the LDAP tree where you expect users
ldap_base = "o=xyz"

# The part of the LDAP tree where the SAMeb
# configuration is located
ldap_sameb_base = "secAuthority=Default"

# Internal configuration
ldap_user_file = "ldap_users"
uid_file = "uid_file"

from subprocess import call

call("ldapsearch -LLL -D " + ldap_dn + " -w " + ldap_pwd
      + " -b " + ldap_base + " -h " + ldap_server
      " objectclass=inetOrgPerson dn >" + ldap_user_file,
           shell=True)

ldap_users = open(ldap_user_file, "r")

for line in ldap_users:
      if len(line) > 1:
           dn = line[4:-1]
           ret_val = call("ldapsearch -D " + ldap_dn +
                " -w " + ldap_pwd + " -h " + ldap_server +
                " -b " + ldap_sameb_base + " secDN=" + dn +
                " | grep numEntries > /dev/null",
                shell=True)
           if (ret_val == 1):
                print "Need to import " + dn
                call("ldapsearch -D " + ldap_dn +
                     " -w " + ldap_pwd +
                     " -h " + ldap_server + " -b " + dn +
                     " objectClass=inetOrgPerson uid " +
                     "| grep uid: >" + uid_file,
                     shell=True)
```

```
uid_f = open(uid_file, "r")
uid = uid_f.read()
uid_f.close()
uid = uid[5:-1]
call("pdadmin -a " + sameb_uid + " -p " +
     sameb_pwd + " user import " + uid +
     " " + dn,
     shell=True);
call("pdadmin -a " + sameb_uid + " -p " +
     sameb_pwd + " user modify " + uid +
     "account-valid yes ",
     shell=True);
```