

Jesua Gonzalez

Ethical Hacking

Lightweight Directory Access Protocol

Sat Oct 11

## Executive Summary

This report covers my hands-on work in the Labtainers LDAP lab, where I explored how the Lightweight Directory Access Protocol(LDAP) centralizes user and group authentication for Linux systems. In a controlled lab environment, I will connect a client to the shared LDAP repository, enumerate and inspect directory entries, create or modify test user records as allowed, and verify that multiple hosts can authenticate against the same credential set. All while taking the necessary screenshots and keeping vital output results to demonstrate my findings and provide context to my learning experience. My goals are to showcase the flow of authentication, observe and analyze encrypted versus unencrypted LDAP traffic, and produce clear, reproducible steps and remediation advice based on my findings.

Throughout the lab, I used Wireshark to capture and inspect network traffic, allowing me to observe LDAP authentication exchanges and determine whether credentials were transmitted securely. I used SSH to perform encrypted logins from the client and to demonstrate what a secure session looks like when authentication is backed by the directory. I also modified LDAP entries to see how changes to user and group records affected authentication. While performing these actions, I saved packet captures, console output, and screenshots so each step can be documented and audited.

## Lab LDAP:

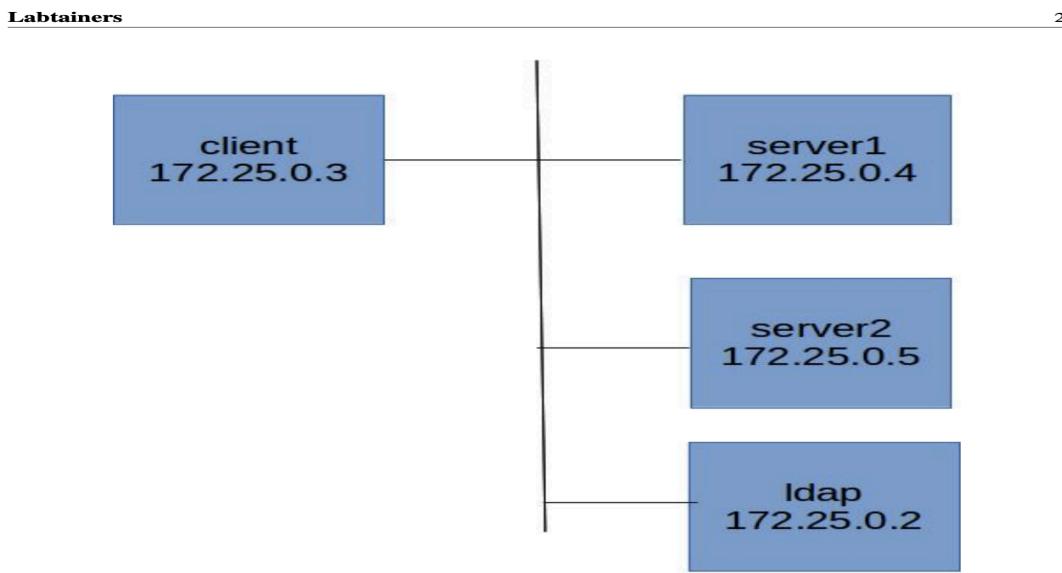


Figure 1: Network topology for the LDAP lab

- The lab provides a client, two servers, and an LDAP server, as shown in the following topology. I used the supplied client and LDAP terminal windows to run commands to authenticate Mike and other users to the LDAP-connected servers.

```

student@LabtainerVMware: ~
admin@ldap: ~

cn: admin
description: LDAP administrator

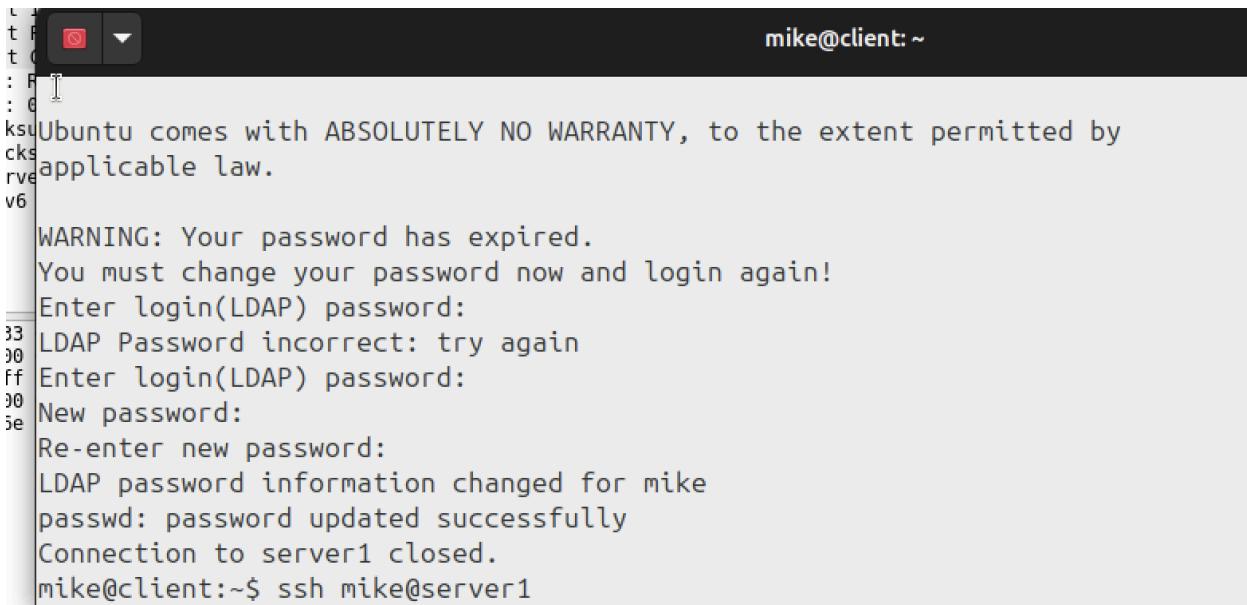
# users, example.com
dn: ou=users,dc=example,dc=com
objectClass: organizationalUnit
ou: users

# groups, example.com
dn: ou=groups,dc=example,dc=com
objectClass: organizationalUnit
ou: groups

# projx, groups, example.com
dn: cn=projx,ou=groups,dc=example,dc=com
objectClass: top
objectClass: posixGroup
gidNumber: 1500
cn: projx

# mike, users, example.com
dn: uid=mike,ou=users,dc=example,dc=com
objectClass: top
objectClass: account
objectClass: posixAccount
objectClass: shadowAccount
cn: mike
uid: mike
uidNumber: 1501
gidNumber: 1500
homeDirectory: /home/mike
: 
  
```

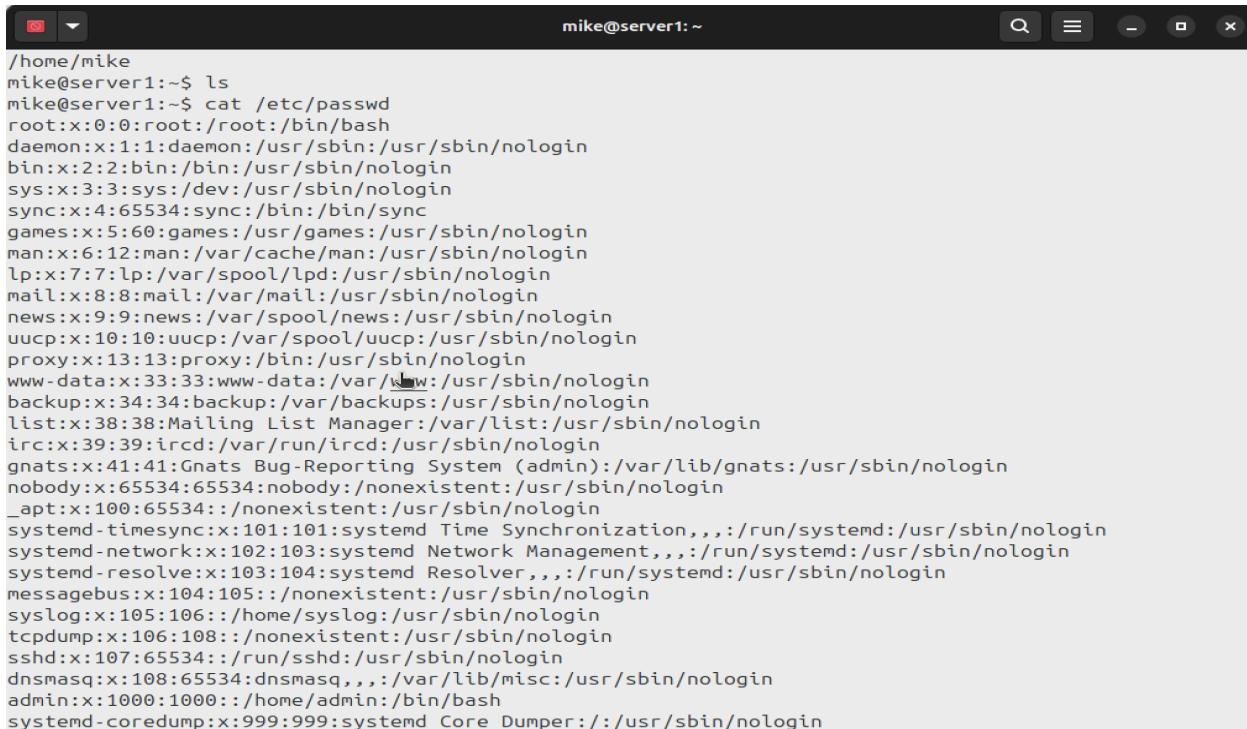
- Starting this lab, I first checked the current directory contents using the command '**ldapsearch -x | less**'. This allowed me to observe the already configured users/groups on the LDAP server, in this case, 'Mike' and 'Projx'.



```
mike@client:~$ ksu
Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

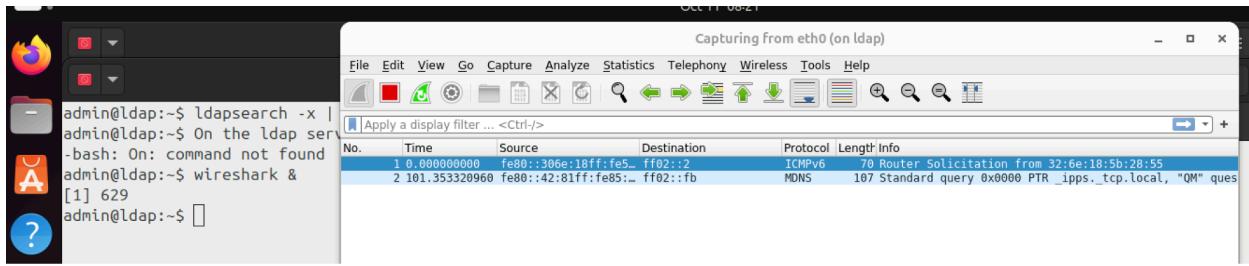
v6
WARNING: Your password has expired.
You must change your password now and login again!
Enter login(LDAP) password:
33 LDAP Password incorrect: try again
ff Enter login(LDAP) password:
30 New password:
3e Re-enter new password:
LDAP password information changed for mike
passwd: password updated successfully
Connection to server1 closed.
mike@client:~$ ssh mike@server1
```

- Using the SSH server login as Mike, I was prompted to change my password: ‘password123’. Usually, this occurs in the real world due to rules such as a password policy, which keeps things in rotation and up to date, to minimize attack surfaces.

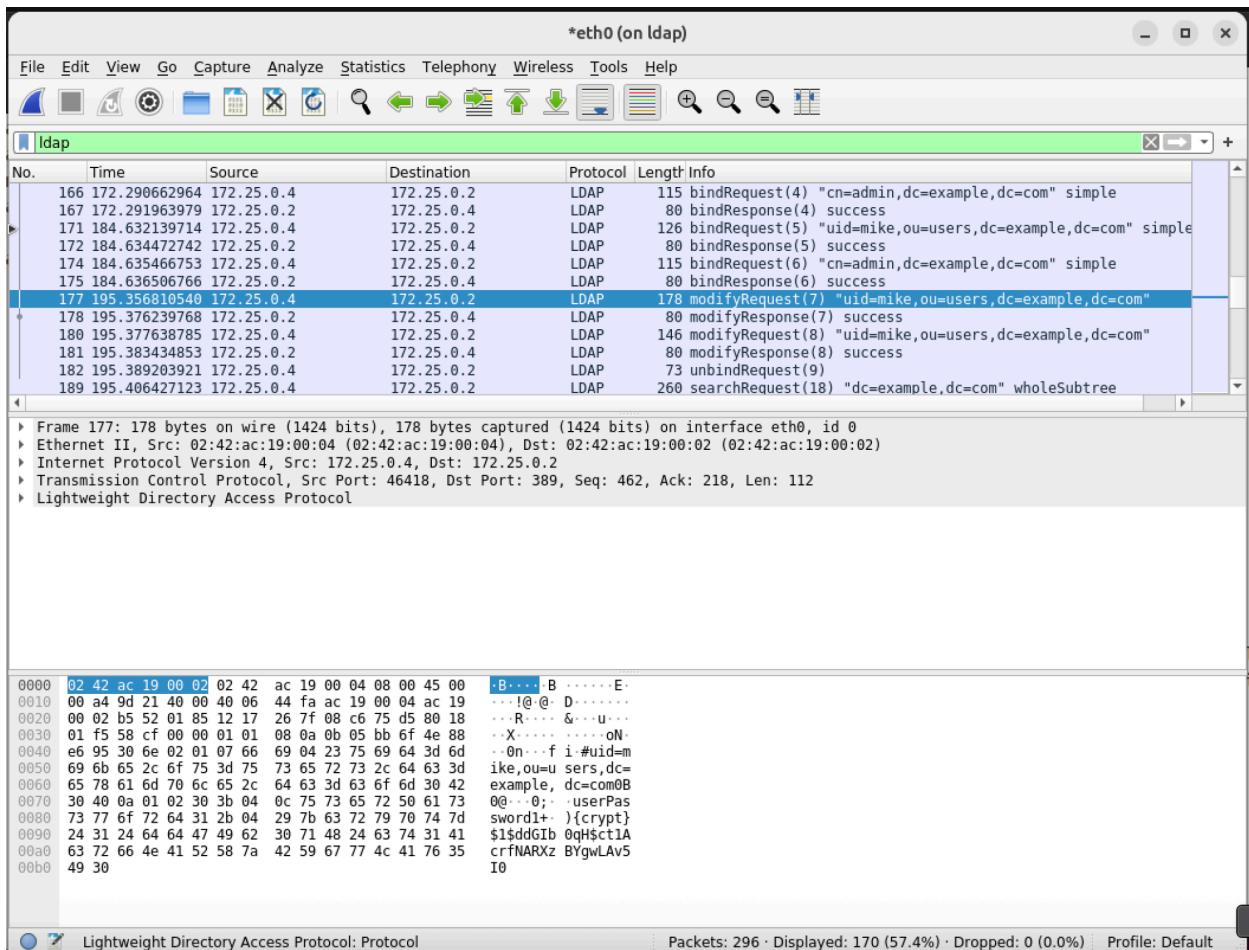


```
mike@server1:~$ ls
mike@server1:~$ cat /etc/passwd
root:x:0:0:root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin/nologin
bin:x:2:2:bin:/bin/nologin
sys:x:3:3:sys:/dev/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
_apt:x:100:65534::/nonexistent:/usr/sbin/nologin
systemd-timesync:x:101:101:systemd Time Synchronization,,,:/run/systemd:/usr/sbin/nologin
systemd-network:x:102:103:systemd Network Management,,,:/run/systemd:/usr/sbin/nologin
systemd-resolve:x:103:104:systemd Resolver,,,:/run/systemd:/usr/sbin/nologin
messagebus:x:104:105::/nonexistent:/usr/sbin/nologin
syslog:x:105:106::/home/syslog:/usr/sbin/nologin
tcpdump:x:106:108::/nonexistent:/usr/sbin/nologin
sshd:x:107:65534:/run/sshd:/usr/sbin/nologin
dnsmasq:x:108:65534:dnsmasq,,,:/var/lib/misc:/usr/sbin/nologin
admin:x:1000:1000::/home/admin:/bin/bash
systemd-coredump:x:999:999:systemd Core Dumper:/:/usr/sbin/nologin
```

- After successfully logging in as Mike with a new password, I was then able to use the ‘ls’ command. This allowed me to display files, and I was able to read files such as ‘/etc/passwd’. I was not able to locate any files with the names tagged under user ‘Mike’ or group ‘Projx’, so I side-noted this and needed to further research file topics.



- On the other hand, while I tried logging into Mike through the SSH server login, I had used ‘**wireshark &**’ to listen to protocol traffic flow on the eth0 device.



- After logging in to server 1 through the client window, I stopped my packet sniffing and used the packet display filter for LDAP results only. This displayed the connections using LDAP on port 389. I was then able to track communication between the LDAP server and server1 through IP addresses 172.25.0.4 ↔ 172.25.0.2. Packet 177 shows a request to modify. In this case, it was the time when I was prompted to change my default password, and I was able to save this packet in a file called ‘**password.pcapng**’ for future reference.

```
mike@client:~$ find 'mike' /etc/passwd
find: 'mike': No such file or directory
/metc/passwd
mike@server1:~$ cd /etc/passwd
-bash: cd: /etc/passwd: Not a directory
mike@server1:~$ exit
logout
Connection to server1 closed.
mike@client:~$ ssh mike@server2
The authenticity of host 'server2 (172.25.0.5)' can't be established.
ECDSA key fingerprint is SHA256:ZtE8xi5Y50aUktZ/XtgjIs1c5jxYQB84Vq5ofmlgGng.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'server2,172.25.0.5' (ECDSA) to the list of known hosts.
mike@server2's password:
Creating directory '/home/mike'.
Welcome to Ubuntu 20.04.2 LTS (GNU/Linux 6.14.0-29-generic x86_64)

 * Documentation: https://help.ubuntu.com
 * Management: https://landscape.canonical.com
 * Support: https://ubuntu.com/advantage

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/*copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

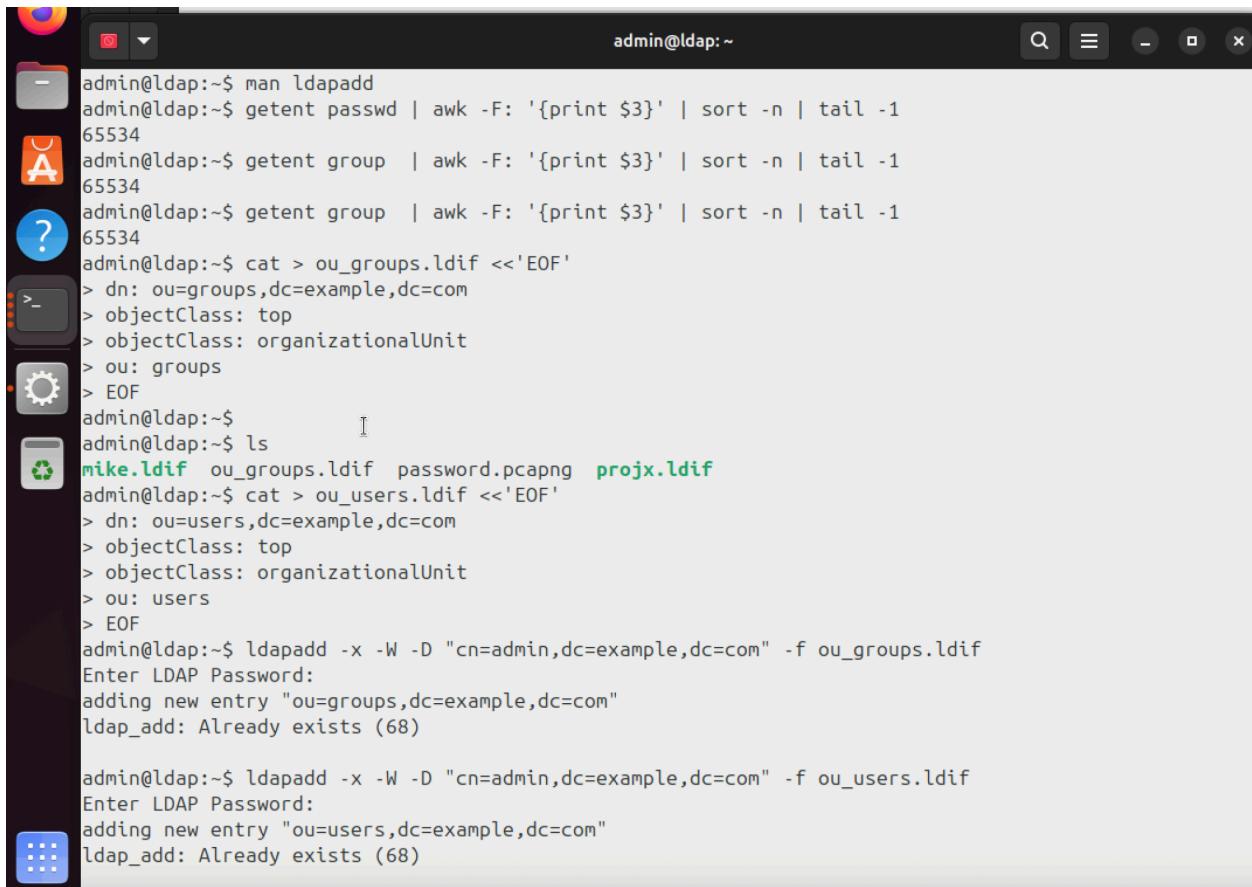
mike@server2:~$ exit
logout
Connection to server2 closed.
mike@client:~$
```

- After exiting the first session with server1 using the ‘Exit’ command, I then switched to logging into server2. What I noticed was that I was able to enter server2 using the new credentials I changed after I first tried server1. This leads me to confirm that LDAP stores credentials to be able to match at later logins to authenticate users in a network.

```
admin@ldap:~$ ls
mike.ldif  password.pcapng  projx.ldif
admin@ldap:~$ cat mike.ldif
dn: uid=mike,ou=users,dc=example,dc=com
objectClass: top
objectClass: account
objectClass: posixAccount
objectClass: shadowAccount
cn: mike
uid: mike
uidNumber: 1501
gidNumber: 1500
homeDirectory: /home/mike
loginShell: /bin/bash
gecos: mike
userPassword: {crypt}x
shadowLastChange: 0
shadowMax: 0
shadowWarning: 0
admin@ldap:~$
```

```
admin@ldap:~$ cat projx.ldif
dn: cn=projx,ou=groups,dc=example,dc=com
objectClass: top
objectClass: posixGroup
gidNumber: 1500
admin@ldap:~$
```

- Back to the LDAP terminal, I viewed the files of the group and users using a simple ‘cat’ command to get a better understanding of what components went into defining these accounts.



The screenshot shows a terminal window titled "admin@ldap:~". The terminal contains the following commands and output:

```
admin@ldap:~$ man ldapadd
admin@ldap:~$ getent passwd | awk -F: '{print $3}' | sort -n | tail -1
65534
admin@ldap:~$ getent group | awk -F: '{print $3}' | sort -n | tail -1
65534
admin@ldap:~$ getent group | awk -F: '{print $3}' | sort -n | tail -1
65534
admin@ldap:~$ cat > ou_groups.ldif <<'EOF'
> dn: ou=groups,dc=example,dc=com
> objectClass: top
> objectClass: organizationalUnit
> ou: groups
> EOF
admin@ldap:~$ ls
mike.ldif  ou_groups.ldif  password.pcapng  projx.ldif
admin@ldap:~$ cat > ou_users.ldif <<'EOF'
> dn: ou=users,dc=example,dc=com
> objectClass: top
> objectClass: organizationalUnit
> ou: users
> EOF
admin@ldap:~$ ldapadd -x -W -D "cn=admin,dc=example,dc=com" -f ou_groups.ldif
Enter LDAP Password:
adding new entry "ou=groups,dc=example,dc=com"
ldap_add: Already exists (68)

admin@ldap:~$ ldapadd -x -W -D "cn=admin,dc=example,dc=com" -f ou_users.ldif
Enter LDAP Password:
adding new entry "ou=users,dc=example,dc=com"
ldap_add: Already exists (68)
```

- I now needed to create two new profiles, one for a user named ‘Mary’ and one for a group named ‘qa’. But first, parent files needed to be created for LDAP to add any new users or groups to a coherent state. I was able to figure out more information using the AWS Directory Service: Create an LDIF guide.

```

admin@ldap:~$ cat > qa.ldif <<'EOF'
> dn: cn=qa,ou=groups,dc=example,dc=com
> objectClass: top
> objectClass: posixGroup
> cn: qa
> gidNumber: 1001
> memberUid: mary
> EOF
admin@ldap:~$ cat > mary.ldif <<'EOF'
> dn: uid=mary,ou=users,dc=example,dc=com
> objectClass: top
> objectClass: person
> objectClass: inetOrgPerson
> objectClass: posixAccount
> objectClass: shadowAccount
> cn: Mary
> sn: Mary
> uid: mary
> uidNumber: 1001
> gidNumber: 1001
> homeDirectory: /home/mary
> loginShell: /bin/bash
> EOF
admin@ldap:~$ ldapadd -x -W -D "cn=admin,dc=example,dc=com" -f qa.ldif
Enter LDAP Password:
adding new entry "cn=qa,ou=groups,dc=example,dc=com"

admin@ldap:~$ ldapadd -x -W -D "cn=admin,dc=example,dc=com" -f mary.ldif
Enter LDAP Password:
adding new entry "uid=mary,ou=users,dc=example,dc=com"

admin@ldap:~$ 

```

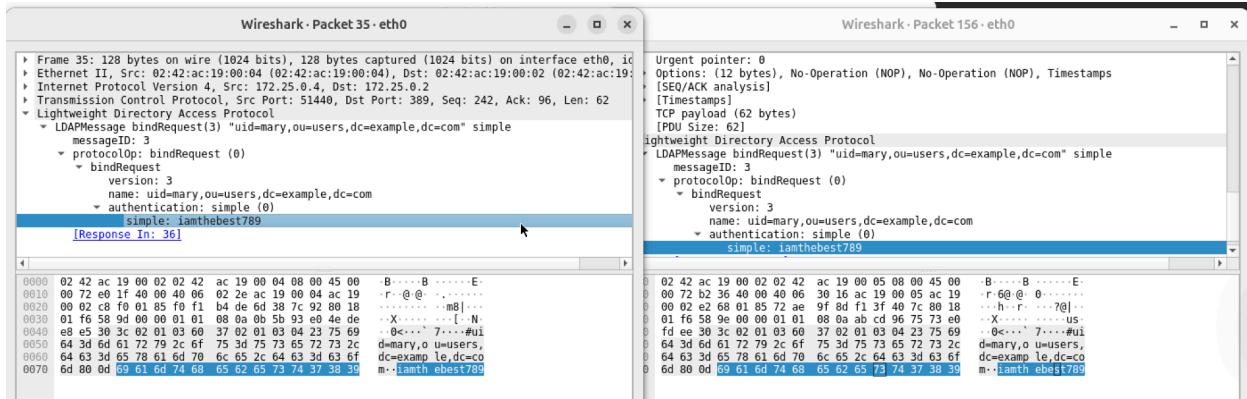
- Once the parent files have been created, we can move forward with the actual users we want to add. I have created LDIF files to input static information and used the command '**ldapadd -x -W -D "cn=admin,dc=example,dc=com" -f mike.ldif**', which was used to create Mike's user profile.

```

admin@ldap:~$ ldappasswd -s "iamthebest789" -W -D "cn=admin,dc=example,dc=com" -x "uid=mike,ou=users,dc=example,dc=com"
Enter LDAP Password:
admin@ldap:~$ 

```

- The final touch was to give Mary's user account a password so they can log in to servers 1 and 2. This was done with the command '**ldappasswd -s password123 -W -D "cn=admin,dc=example,dc=com" \ -x "uid=mary,ou=users,dc=example,dc=com"**'.



- I went ahead and tried Mary's new account by logging into both servers, which was a success. Simultaneously, I ran Wireshark to capture packets and analyze communication, and to my surprise, Mary's credentials were being shared over the network via plaintext. This poses a huge risk to unauthorized access if a real threat actor were to intercept these credentials through this LDAP vulnerability.

```

admin@server1:~          admin@server2:~          admin@server1:~
admin@server1:~          admin@server2:~          admin@server1:~

GNU nano 4.8              /etc/ldap.conf

# Your LDAP server. Must be resolvable without using LDAP.
# Multiple hosts may be specified, each separated by a
# space. How long nss_ldap takes to failover depends on
# whether your LDAP client library supports configurable
# network or connect timeouts (see bind_timelimit).
#host 127.0.0.1

# The distinguished name of the search base.
base dc=example,dc=com

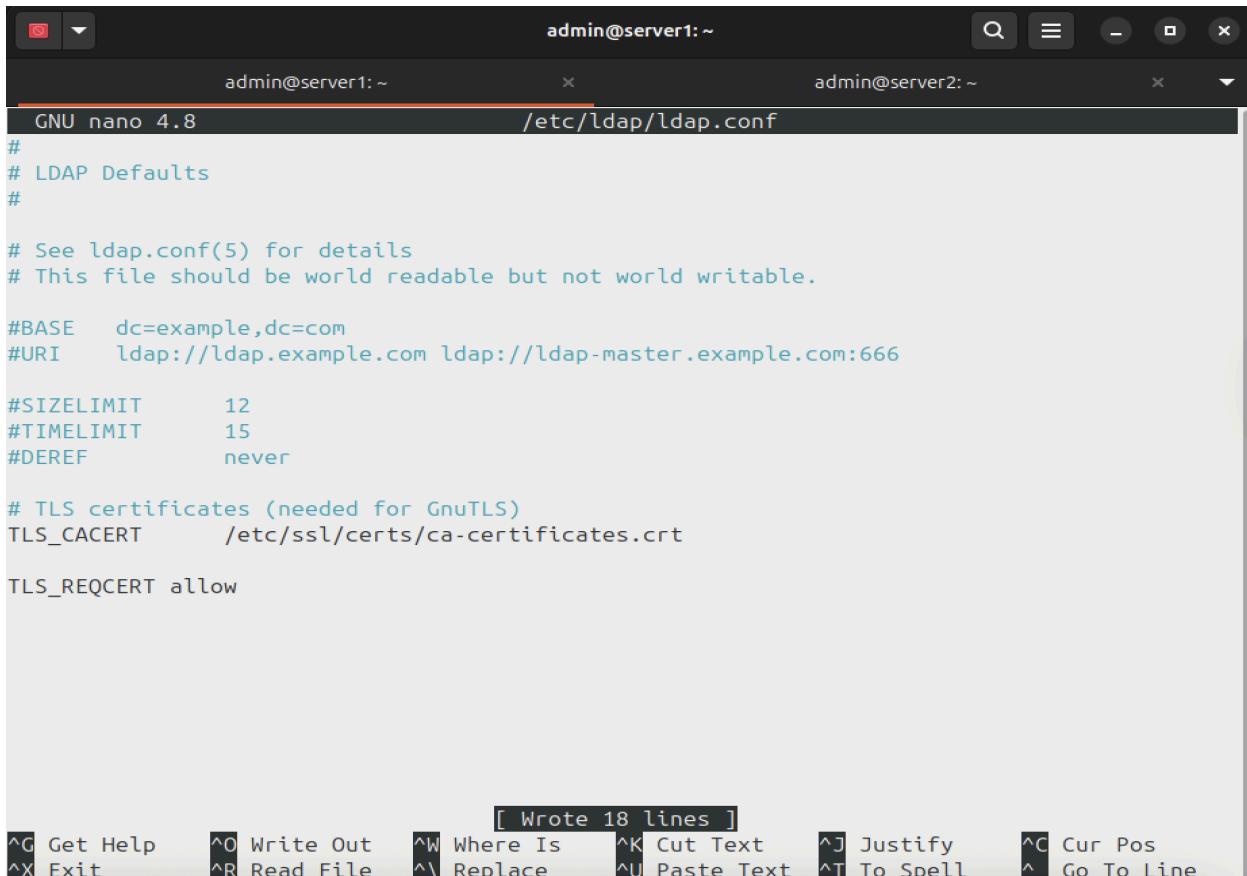
# Another way to specify your LDAP server is to provide an
uri ldaps://ldap
# Unix Domain Sockets to connect to a local LDAP Server.
#uri ldap://127.0.0.1/
#uri ldaps://127.0.0.1/
#uri ldapi://%2fvar%2frun%2fldapi_sock/
# Note: %2f encodes the '/' used as directory separator

# The LDAP version to use (defaults to 3
# if supported by client library)
ldap_version 3

# The distinguished name to bind to the server with.
# Optional: default is to bind anonymously.
#binddn cn=proxyuser,dc=padl,dc=com

^G Get Help    ^O Write Out   ^W Where Is    ^K Cut Text   ^J Justify   ^C Cur Pos
^X Exit       ^R Read File   ^\ Replace     ^U Paste Text ^T To Spell  ^_ Go To Line
                                         Trash

```



```

GNU nano 4.8
#
# LDAP Defaults
#
# See ldap.conf(5) for details
# This file should be world readable but not world writable.

#BASE    dc=example,dc=com
#URI     ldap://ldap.example.com ldap://ldap-master.example.com:666

#SIZELIMIT      12
#TIMELIMIT      15
#DEREF          never

# TLS certificates (needed for GnuTLS)
TLS_CACERT      /etc/ssl/certs/ca-certificates.crt

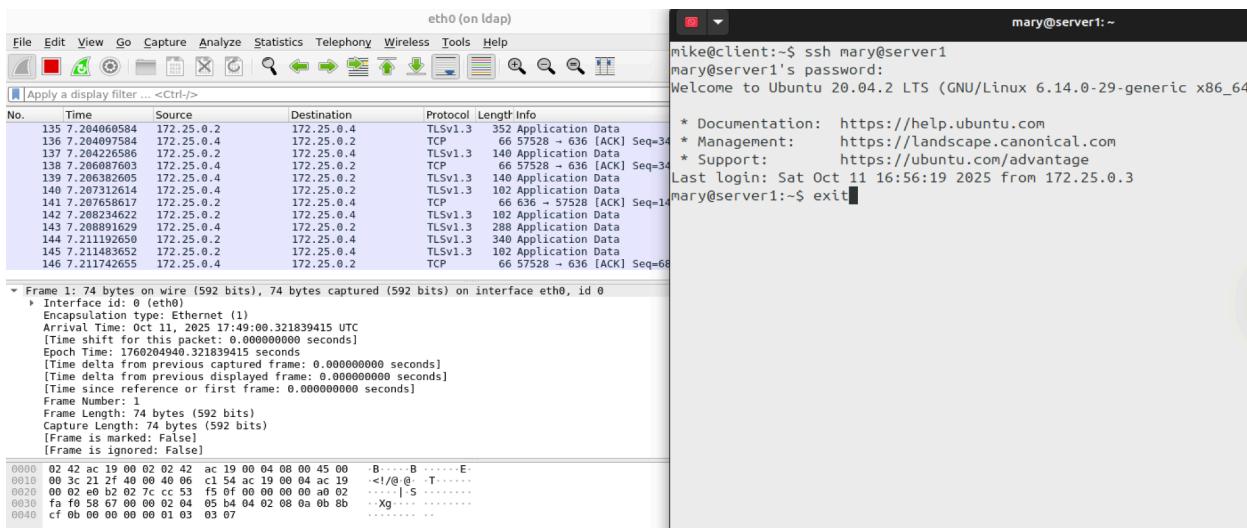
TLS_REQCERT allow

```

[ Wrote 18 lines ]

^G Get Help ^O Write Out ^W Where Is ^K Cut Text ^J Justify ^C Cur Pos  
 ^X Exit ^R Read File ^\ Replace ^U Paste Text ^T To Spell ^\_ Go To Line

- It was time to upgrade LDAP communication to our servers, so on the server terminals, I went into the LDAP configuration files using a simple text editor, **nano**, to add SSL encryption to LDAP. I changed the uri to display ldaps instead of ldap and removed the # in front of the line that had ‘SSL on’. Lastly, I was able to write out ‘**TLS\_REQCERT allow**’ on the /etc/ldap/ldap.conf file path to finish the secure LDAP configuration for both servers.



eth0 (on ldap)

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

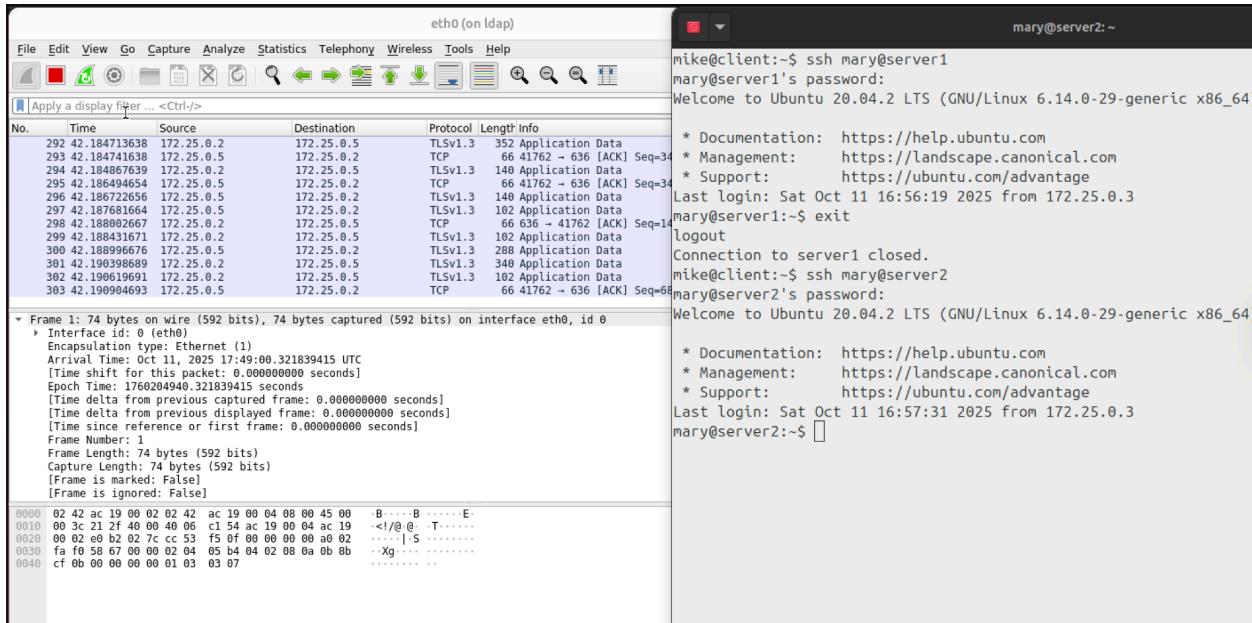
Apply a display filter ... <Ctrl-/>

No.	Time	Source	Destination	Protocol	Length Info
135	7.204060584	172.25.0.2	172.25.0.4	TLSv1.3	352 Application Data
136	7.204097584	172.25.0.4	172.25.0.2	TCP	66 57528 - 636 [ACK] Seq=34
137	7.204226588	172.25.0.2	172.25.0.4	TLSv1.3	140 Application Data
138	7.204267588	172.25.0.4	172.25.0.2	TCP	66 636 - 57528 [ACK] Seq=34
139	7.206302605	172.25.0.4	172.25.0.2	TLSv1.3	140 Application Data
140	7.207312614	172.25.0.4	172.25.0.2	TLSv1.3	102 Application Data
141	7.207658617	172.25.0.2	172.25.0.4	TCP	66 636 - 57528 [ACK] Seq=141
142	7.208234622	172.25.0.2	172.25.0.4	TLSv1.3	102 Application Data
143	7.208891628	172.25.0.4	172.25.0.2	TLSv1.3	288 Application Data
144	7.211192650	172.25.0.2	172.25.0.4	TLSv1.3	340 Application Data
145	7.211483652	172.25.0.2	172.25.0.4	TLSv1.3	102 Application Data
146	7.211742655	172.25.0.4	172.25.0.2	TCP	66 57528 - 636 [ACK] Seq=68

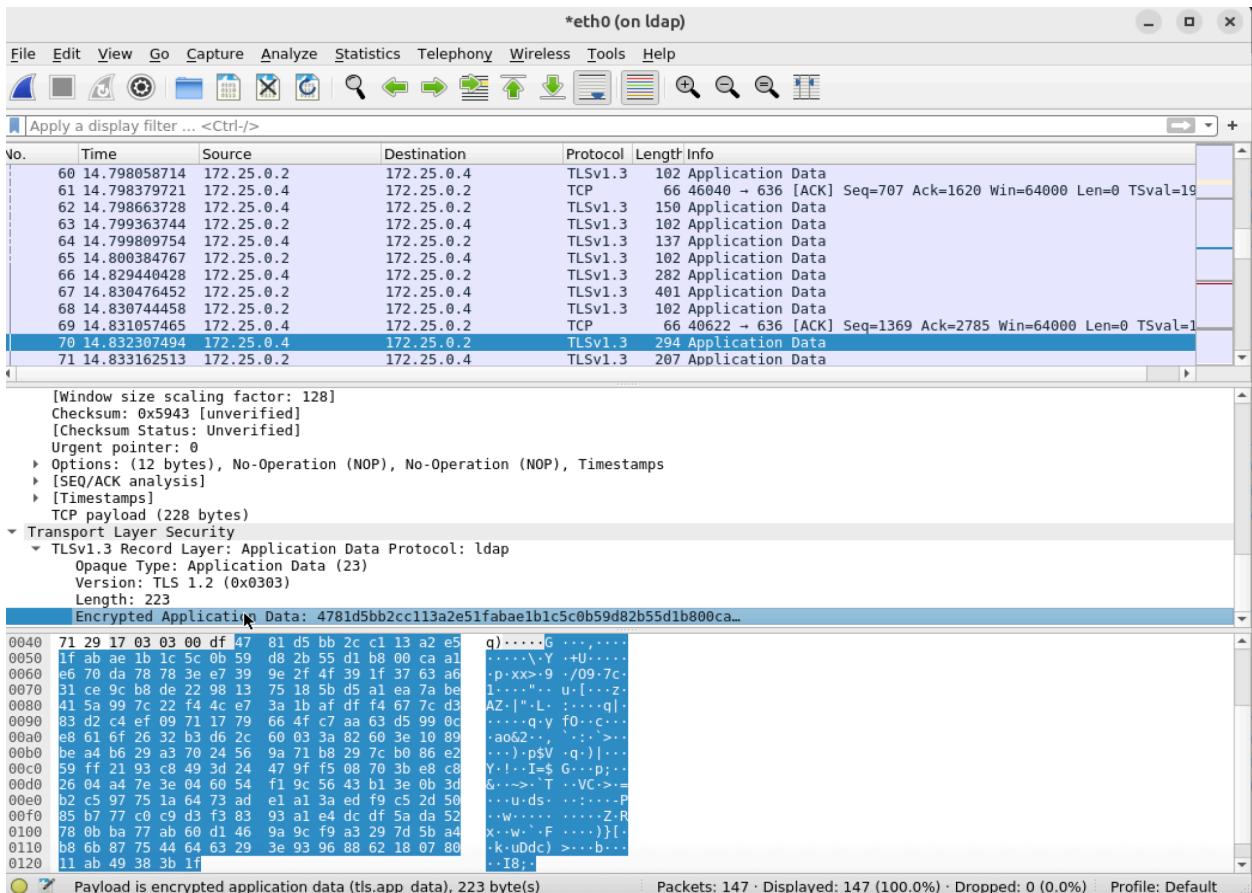
Frame 1: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface eth0, id 0

Interface id: 0 (eth0)  
 Encapsulation type: Ethernet (1)  
 Arrival Time: Oct 11, 2025 17:49:00.321839415 UTC  
 [Time shift for this packet: 0.000000000 seconds]  
 Epoch Time: 1760204940.321839415 seconds  
 [Time delta from previous captured frame: 0.000000000 seconds]  
 [Time since reference or first frame: 0.000000000 seconds]  
 Frame Number: 1  
 Frame Length: 74 bytes (592 bits)  
 Capture Length: 74 bytes (592 bits)  
 [Frame is marked: False]  
 [Frame is ignored: False]

No.	Time	Source	Destination	Protocol	Length Info
0000	02:42:ac:19:00:02	02:42:ac:19:00:04	00:08:00:45:00	B.....B.....E	
0010	00:3c:21:2f:40:00	00:40:c1:54:ac:19	00:08:00:45:00	<1/0@...T.....	
0020	00:02:ea:b2:7c:cc	53:f5:01:00:00:00	00:00:00:00:00:02	..... S.....	
0030	fa:f0:58:67:00:00	00:02:04:05:b4:04	02:08:0a:0b:08:0b	...Xg.....	
0040	cf:0b:00:00:00:01	03:07			



- Now it was time to test the newly configured LDAP with SSL by logging in to the servers with a user account, and with a new Wireshark session, I was able to keep track of the time of packets being sent out and in.



- Taking a closer look at the packets, we can see a few changes, starting with the protocol being displayed with the name ‘TLSv1.3’ instead of the old LDAP protocol. Also, when inspecting a single packet, there is new information saying Encrypted Application Data, in which packet information is no longer sent in plaintext, and we are only given a bunch of random characters showcasing the encryption over this network connection.

## **Unencrypted vs. encrypted LDAP traffic**

In the Labtainers LDAP lab, I saw why encrypting LDAP matters. Unencrypted LDAP on port 389 sends binds, searches, and modify operations in cleartext, so anyone sniffing the network with tools like Wireshark can read domain names, attributes, and even credential data. Encrypted LDAP using SSL(LDAPS) protects those same operations with TLS; therefore, credentials and directory contents are not easily intercepted or replayed. Good security steps are simple and effective. Future businesses should require LDAPS for all clients and servers, make clients validate the server certificate, and install a trusted CA. Apart from encryption, there should be strict ACLs so non-admins cannot read sensitive information, as well as enforce strong password policies, and limit which hosts can reach the LDAP port with firewalls or VLANs. Enabling logs and monitoring of LDAP activity, rotating certificates regularly, and testing changes in a controlled environment before rolling them out can provide multiple layers of security control. Documenting my experience during the Labtainers LDAP exercise will benefit future assessments of similar situations and establish a foundation for Lightweight Directory Access Protocol security.