|  |  |
| --- | --- |
| Describe | Vulnerability |
| Service | SSh |
| Port | 22 |
| Version | 2.4.18 |
| Severity | Medium |
| CVE id |  |
| Cvss score | 6/10 |
| Remediation | 1. **Exposure of sensitive information via** *robot.txt*  * **Issue:** The robot.txt file discloses sensitive paths, such as /ssh\_cred.txt, which can be accessed directly, revealing credentials. * **Remediation:** * **Avoid listing sensitive directories:** Do not include sensitive paths in robot.txt. * **Implement Access Controls:** Protect sensitive directories and files with proper authentication mechanisms. * **Regularly review robot.**txt: Ensureit doesn’t inadvertently expose critical paths.  1. **Use of Weak or hardcoded credentials**  * **Issue:** credentials like eric:1mw4ckyyucky6 are exposed and can be easily exploited. * **Remediation:** * **Enforce strong password policies:** require complex, unique passwords for all user accounts. * **Avoid Hardcoding Credentials:** never store credentials in files accessible via the web. * **Regularly audit credentials:** periodically review and update credentials to ensure they remain secure  1. **Privilege Escalation via Misconfigured Cron jobs**  * **Issue:** a cron job runs a script (/opt/clean.sh) with root privileges, which root privileges, which can be exploited to gain unauthorized access. * **Remediation:** * **Restrict script permissions:** Ensure that scripts executes by cron jobs are not writable by non-privileged users. * **Validate cron jobs:** regularly audit cron jobs to ensure they don’t introduce security vulnerabilities. * **Implement principles of least privilege:** Limit the permissions of scripts and users to the permissions of the scripts and users to the minimum necessary.  1. **Inadequate input validation and encoding practices**  * **Issue:** the application uses base64 and Ook encoding to obfuscate sensitive information, which can be easily decoded. * **Remediation:** * **Avoid obfuscation for security:** do not rely on encoding methods like base64 or Ook for securing sensitive data. * **Implement proper encryption:** use strong encryption algorithm to protect sensitive information. * **Implement proper encryption:** usestrong encryption algorithm to protect sensitive information. * **Validate user input:** ensure all user inputs are properly validated and sanitized to prevent injection attacks.  1. **Lack of proper subdomain management**  * **Issue:** subdomain like test.drifftingblues.box are accessible environments. * **Remediation:** * **Restrict access to non-production subdomain:** ensure that development or test environment are not publicly accessible. * **Implement access control:** protect subdomains with authentication mechanism. * **Regulary audit DNS records:** review and manage DNS records to prevent unintended exposure. |
| Poc | Step 1  First I find our ip in kali Linux    After this I find target ip add with the help of netdiscover    I get the target ip (192.168.1.107)  Now I use nmap for looking open port of target machine    As you see 2 ports are open ssh, http (22,80)  Step 2  So now I start to research about target ip, at first I see the website    Here is the website of target machine so now I start my research towards it    When I scroll down I find 2 name “Sheryl and eric” so I think it’s name helps me while “**ssh login”** so I’II save both name in my terminal for future reference    After this I look source code of website for hidden/sensitive detail (if data available in source code then it’s useful)    As I guess I get the encode data In source code so now I decode in kali    As you see after decode I get hide directory (/noteforkingfish.txt ) so now I look what inside in this directory    In that directory I get lot of “Ook! Ook.” Message or hidden code. after some research I find a clue this okk is not a message is a encode data so now I find decoder in google    After decoding this data I get hidden message by “eric”  ‘My man, I know you are new but you should know how to use host file to reach our secret loaction’  Step 3  So now I use dirbuster for getting more directory and also finding secret location!!  Gobuster:- Gobuster dir -u <http://192.168.1.107> -w /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt -x html,php,txt    After running gobuster I get *“secret.html”* so, let’s check out!    After using “*secret.html”* directory I get another message ‘dig.. deeper.. maybe you’  Now I want to dive deeper and gather more information. So now I use gobuster ‘vhost’ mode  **In vhost mode, vhost enumerate is the process of discovering these hidden or lesser-known subdomain configured as virtual hosts on a target**  Gobuster vhost -u (DNS name) -w /usr/share/wordlists/dirb/common.txt –append-domain      As you see I get new hidden DNS ‘test.driftingblues.box’   |  | | --- | | * **TIP:- if you found error while enumeration do nano /etc/hosts and put ip and DNS name problem solve**     **Like this and then run cmd again** |   Step 4  After that I assign that DNS and ip in host file like this    After this I access website with new domain    Here I get message work in progress so now I do gobuster to discover directory and hidden directory on the new domain    And here I get one of most important file ‘*robot.txt*’ file so its time to get something great    After opening robot.txt directory I get ssh password “1mw4ckyyucky” but here the problems is sherly said she add a number in password but in this situation I didn’t see any type of number in password. So now I get one job to add correct number in password to get access of ssh  For adding number I use cmd that chat-gpt give me    **(command info)**  for i in {1..20}: This is a loop that iterates from 1 to 20.  do echo "1mw4ckyyucky$i": For each iteration, it prints the string 1mw4ckyyucky followed by the current number ($i).  done: Ends the loop.  > passwords.txt: Redirects all the output (the list of generated passwords) into a file named passwords.txt.  After this I use cat command to view the password file    As we see I get passwords with ends with number  So now I have username and password, now I can use hydra command for getting perfect username & password    Step 5  As you see I get perfect combo of username and password so now I can access ssh.    Its time to find user flags and root flag so let’s find out!    In this place I get hint, ‘flag 1/2’ so it means only root user left so now I find more information and trying to accessing root user  after gathering some information I get something interesting in backup file    #!/bin/bash  /usr/bin/zip -r -0 /tmp/backup.zip /var/www/  /bin/chmod  #having a backdoor would be nice  sudo /tmp/emergency  1. #!/bin/bash  This is a shebang line that tells the system to run the script using the Bash shell.  2. /usr/bin/zip -r -0 /tmp/backup.zip /var/www/  This command creates a compressed archive named backup.zip in /tmp, recursively including everything from /var/www/. The -0 option means no compression (just archiving).  3. /bin/chmod  This line is incomplete. The chmod command is used to change file permissions, but it's missing the required arguments (e.g., chmod 755 filename).  4. #having a backdoor would be nice  This is a comment in the script. While not executed, it implies malicious intent or a joke about it.  5. sudo /tmp/emergency  This command executes a file named emergency located in /tmp with superuser privileges. If this file is malicious, it could be very dangerous — it may represent a backdoor or some unauthorized program.  Step 6  First I try to go /tmp/emergency directory then I create backdoor    No such directory name by emergency so I crate one after that I can execute payload, after that I get root access, so let begin    (echo ‘cp /bin/bash /tmp/[filename];chmod +s /tmp/[filename]’ > /tmp/emergency  Filename:- you can give anyname just like I use ‘superuser’  After this I use chmod 777 emergency for make file executable    Now I see all file with ‘ls -all’ cmd    At first time file didn’t come after waiting 5-10 seconds I get file name *“superuser”*  So it’s time to become root now I use command ‘./’ for execute superuser    Command didn’t work propely so now I do little bit changes so my file execute propely  For that I just add -p and execute again  The -p is likely a command line argument passed to the superuser program. The meaning of ‘-p’ depends entirely on how the superuser program is written.  Example:- privileged mode    So it’s time to find root access    And here I find root access and  now I complete this machine successfully |
| Reference |  |

Csv:- [..\drifting blue 1.csv](../drifting%20blue%201.csv)