Git-Gambit Write-up

Author: Anirudh, Adars

Upon running nmap verbose nmap scan on the target we see that the port 2222 and 5555 are open and they are serving SSH and FTP respectively. We also come to know that anonymous login is allowed on the ftp server

Upon authenticating in the ftp server , we can see two files , them being a auth.kdbx file (Keepass) and crash.zip file .

We download them into our local machine , and upon extracting the zip file , we get a crash.dmp file. Upon running the file command we get the Keepass version , which has a vulnerability that allows us to extract master-password as plain text from the Keepass crash file https://nvd.nist.gov/vuln/detail/CVE-2023-32784

Upon exploiting the crash file we get the master password , which allows us to unlock the kdbx file. This gives us user johns Ssh credential

Upon Sshing into the machine , we get the user flag.

We then enumerate the system and find out that Gitlab 16.0.0 is running locally which is vulnerable to arbitrary file read https://nvd.nist.gov/vuln/detail/CVE-2023-2825

For this exploit to work , we need the gitlab credentials. We further enumerate the machine anf find that the Gitlab creds have been leaked in the gitlab-bootlog file. We can also interestingly see that the id_rsa (ssh private) that was generate during the bootup belongs to the user bob (since the public key is stored under

bobs name directory. As a result the id_rsa present in the /etc/gitlab which belongs to bob is readable by git user

So we exploit the Gitlab service , my tunneling it to our local machine. We exploit this to read the id_rsa file.

We then use this to login as bob user. We find that there are two sub folders present in bob’s user directory , tools and games. In the tools directory , we see that there is a binary file , that when executed extracts the gitlab password from the gitlab-bootlog file.

The intersting fact is that , the file has an suid bit set , and upon reversing the file , we see that the program uses grep function to do so and full path of the grep function is not specified

Therefore we exploit the path env variable and make the binary read our executable grep which is /bin/bash . This gives us the root shell and gives us the root flag.