- the identified vulnerability/vulnerabilities  
- how your exploit program exploits it/them  
- how it/they could be fixed (by specific changes to the vulnerable program itself, not by system-wide changes like adding ASLR, stack canaries, NX bits, etc.)

Sploit1.c

In sploit1.c, it exploits a buffer-overflow vulnerability in the program pwgen at line 264.

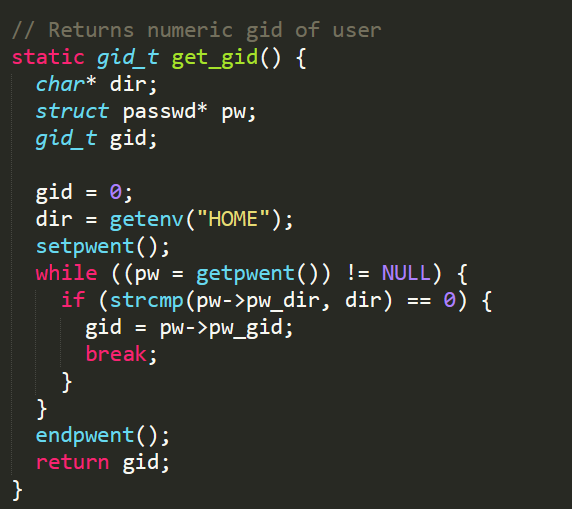
strcpy(args.filename, optarg);

where args.filename is a variable on stack, and optarg is an argument provided by the user. We can exploit it by putting the NOP, then shellcode in the optarg, and the address of args.filename, causing a buffer overflow, and thus change the return address and exploits the program, opening a shell with root priviledge.

Sploit1.c supplies pwgen with arguments -e, where it will go into the code.

To fix this, we can use strncpy with FILENAME\_SZ instead of strcpy. It will ensure that only the first FILENAME\_SZ characters are copied to args.filename and null-terminated buffer.

Sploit2.c



In sploit2.c, it exploits an incomplete mediation vulnerability in the get\_gid function. The function didn’t verify the user input as the environment variable can be changed. So, if we were to change the environment variable of HOME, then get\_gid will return the id 0, which is the root user. Then pwgen will change root’s password. We can exploit it by using the password generated by pwgen to log in into the root.

This can be fixed by avoiding using getenv to check for gid. Use the c getuid() function instead.