Temporary Files, Shared Libraries

Petr Ročkai

Temporary Files

- useful for communication
- and/or co-opting external tools
- to a lesser degree to store large data
- should be removed upon exit
- usually created in a special, system-wide directory
  - like `/tmp` and `/var/tmp` on most UNIX systems

POSIX APIs

- `mkstemp` and `mkdtemp`
- temporary files and directories respectively
- details in man pages: `man mkdtemp`

How to use:

```
char fn[] = "/tmp/mytmpdir.XXXXXX";
if ( !mkdtemp(fn) )
    return ERROR;
/* ... */
rmdir( fn );
```

Exercise 1

- create a temporary directory in `/tmp`
  - create a named fifo in the directory (`man 3 mkfifo`)
  - print the path of the fifo on `stderr`
- open the pipe for reading
  - anything you read, print on `stdout`
  - use the POSIX API for reading: `man read`
- in another terminal, run `echo hello > fifo`
  - substitue the path the program printed for `fifo`
  - run another copy of your program

Exercise 1 Bonus

- the program is stopped when the user hits `^C`
- ensure that the fifo and the directory are removed
  - you can use the `signal` function for this
  - see `man signal`
  - the fifo must be `unlink`-ed
  - the remaining empty directory can be `rmdir`'d

Dynamic Linking Redux

- an application consists from multiple modules
- upon load, the modules are linked by the runtime linker
- on UNIX, the linker is known as `ld.so`
  - see the manual page: `man ld.so`
  - look for the description of `LD_PRELOAD`

Hooking Calls

- call targets are resolved by `ld.so` at load/run time
- procedures are referenced as symbols (ASCII names)
- multiple libraries can provide the same symbol
- `ld.so` has rules to decide which function to use
- preloaded libraries come before all the others

Exercise 2

- get `ld_preload.tar.gz` from study materials
- `preload.c` is a small C library that implements `open`
- the script `preload.sh` runs a command given as argument
  - it first builds `preload.so` from `preload.c`
  - then preloads it into the application
- try for instance `./preload.sh cat /dev/null`
  - compare to just `cat /dev/null`

Exercise 3

- modify `preload.c` to hijack `read` calls
- create a file like `/tmp/.exfiltrate.XXXXXX`
  - you should use `mkstemp` for this
  - do not unlink the file when the program exits
- print all the captured `read` data into the file
- run your program from Ex. 1 using `preload.sh`
  - verify that this behaves as expected

Homework

- there is no homework

Next Week

- no lecture on Monday
- project defences on Thursday