

Spring 2009 - CSE509 - System Security

Lecture Notes – 02/27/2009

Sequencing Bugs

There are three rules that should followed to avoid sequencing bugs.

- (1) Don't allow *longjmp()* calls as *root*.
- (2) Don't allow calls to *exec()* as *root*.

```
setuid(0);  
...  
exec(helper_program);
```

- (3) The call to *chroot()* must always be followed by *chdir()*.

```
chroot(some_dir);  
chdir("/");
```

FSA Representations

These three rules can be represented as a Finite State Automaton and that would be very handy in performing model checking. Each node represents a state and the arrows mark the transition from one state to another. The terminal node (state) has a dark boundary like which differentiates it from the other nodes.

Figure 1 has the FSA representation for the rules (1) and (2) and Figure 2 has a representation for the rule (3).

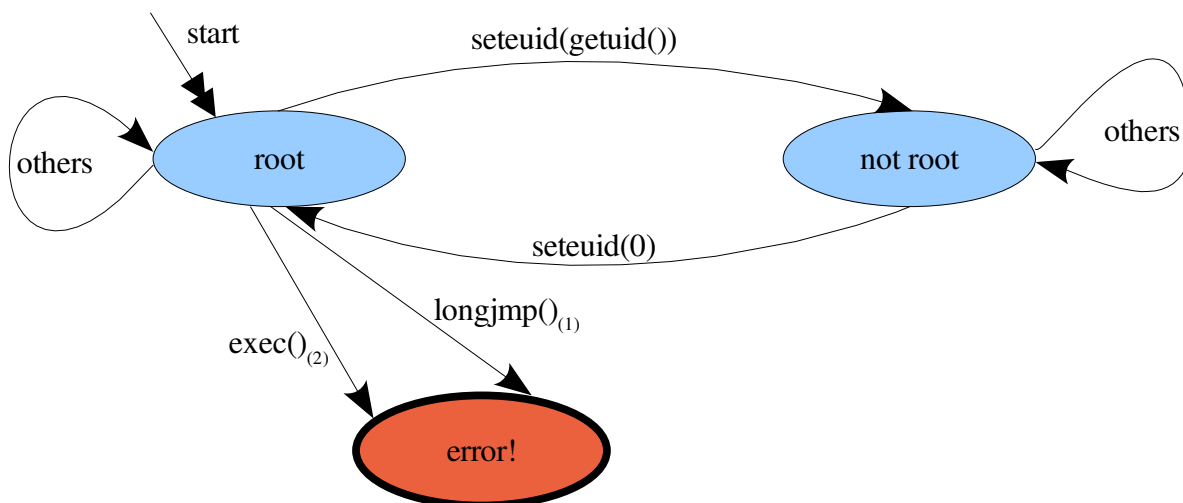


Figure 1: FSA representation for the rules (1) and (2)

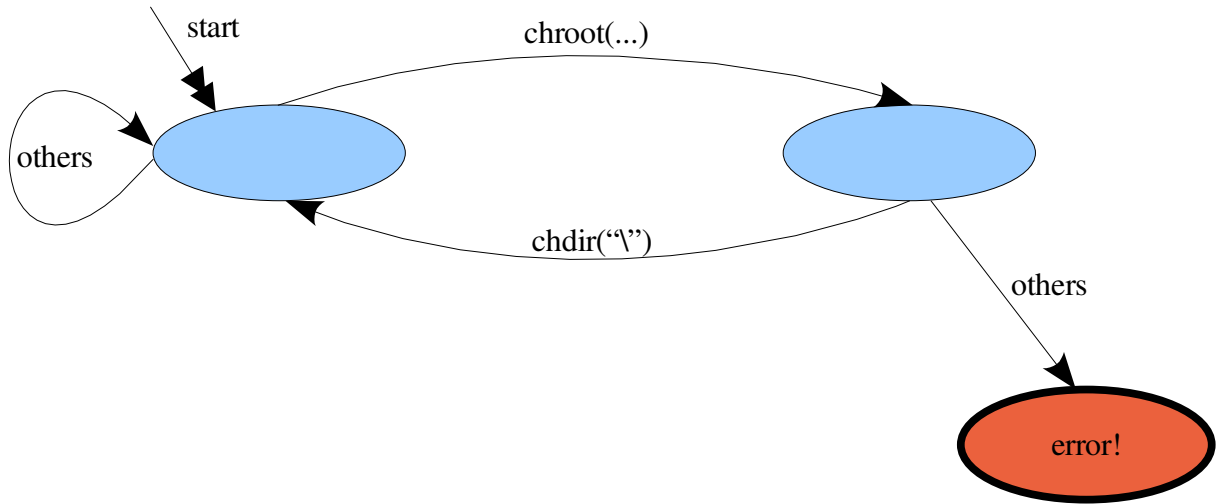
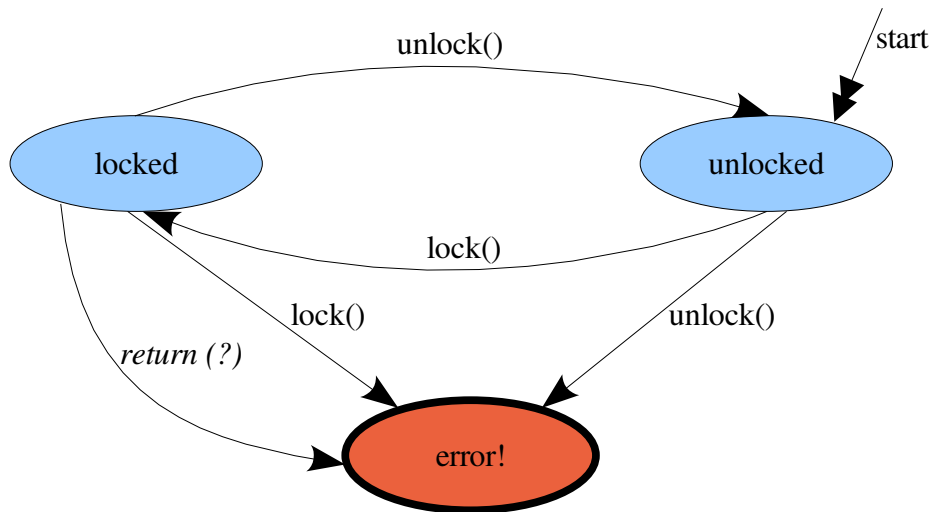


Figure 2: FSA representation for the rule (3)

There is another example with locks which is similar to these rules. This rule enforces the policy that, processes are not allowed to call the lock() function when it is already in the locked state. Similarly, the processes are not allowed to call the unlock() function when they don't have any acquired locks. These two situation leads to an error and the below FSA depicts that.



The above FSAs are denoted as M_S , model of specification for all possible error conditions.

MOPS (Model Checking)

Lets now check out a program (P) and its model representation (M_P).

```

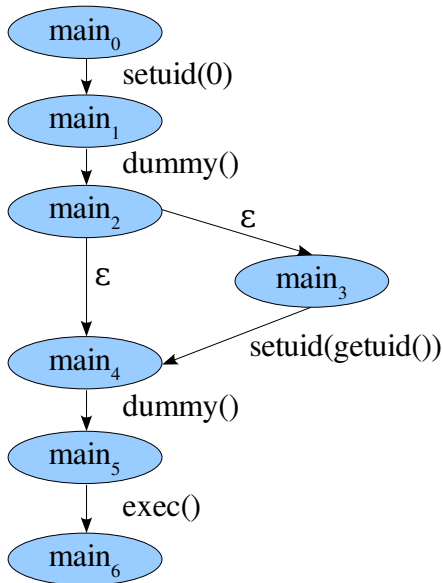
int main()
{
    ...
    setuid(0);
    dummy();
}
  
```

```

if (...)
    setuid(getuid());
dummy();
exec();
...
}

```

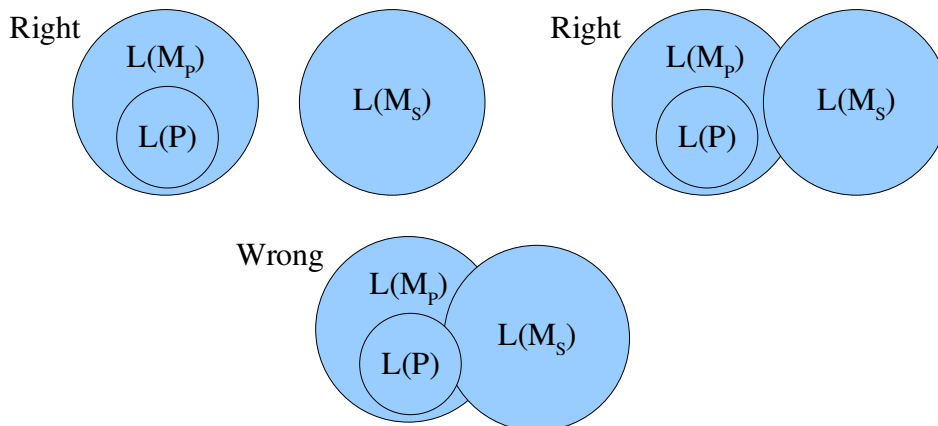
From the above code snippet, it is clear that by chance if the *if* condition is not true, the `exec()` call will be executed with super user privileges. The above program's state diagram is drawn below and it is denoted as M_p .



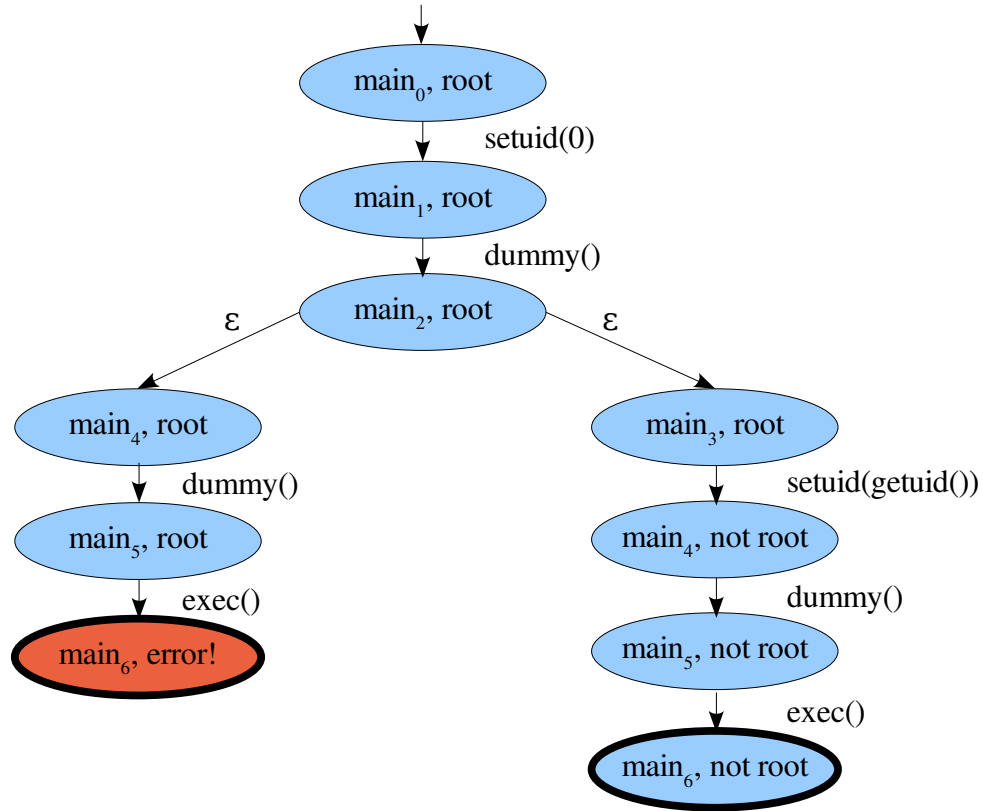
Lets denote $L(M)$ to represent the language generated/accented by a model M .

Our goal is to make sure that the below condition is true, so as to make sure that the program is free from sequencing bugs.

$$L(M_s) \cap L(P) = \emptyset$$



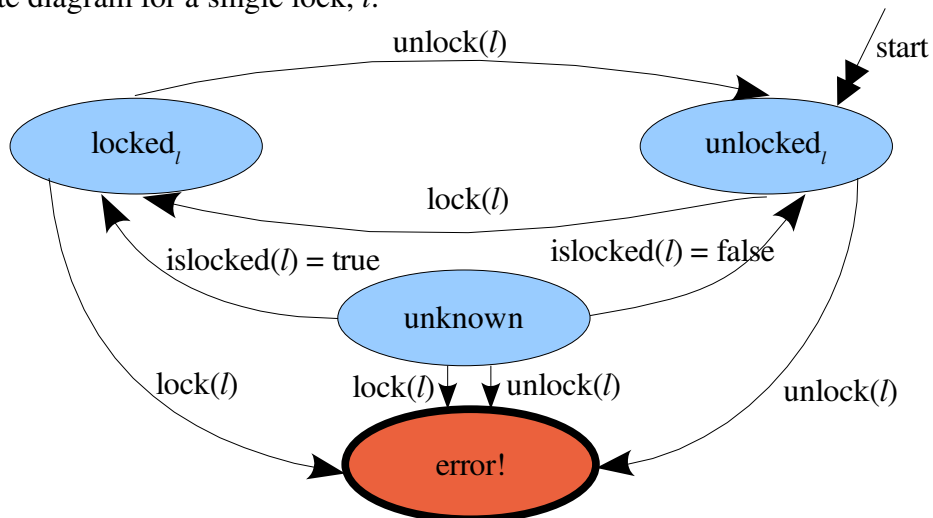
Here is the representation of $M_P \times M_S$



Metal Extensions

- Test conditions
- Mention data
- Track data state
- Not sound

Lets consider the locking example discussed above with incorporating Metal extensions. The below diagram is a state diagram for a single lock, l .



Here is a code snippet to demonstrate this model, which is fool proof.

```
lock(l);  
if (...)  
    unlock(l);  
.  
.  
  
if (islocked(l))  
    unlock(l);
```

The state diagram for this code snippet is given below.

