Lecture Notes – Feb 20,2007

Model Checking

There are several model checkers available that can used for model checking. These include:
- MECA/MC
- MOPS
- ESP
- SLAM/BLAST

MOPS: Open Source Model
MECA: Developed by the company 'Coverity'
They do not want to disclose the source code.
You have to give your code to them and they will run the model and find the bugs and give it back to you.

Temporal Ordering Rules

1) Never call exec while holding root privileges
2) Never call two syscall on the same filename.
   A more specific statement of rule (2) can be given as:
   Never call open(f) after access(f) where 'f' is the filename. Why?

Access/Open Race Condition (Time of Check to Time of Use Bug)

Consider you the have the following statement:

```
if (access(filename, W_OK))
    open (filename, O_WROnly);
```

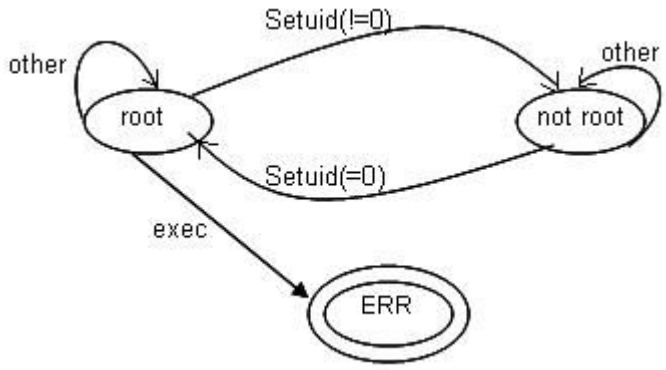
---------- > indicates 'Context Switch'

Victim		Attacker
		ln –s dummyfile fname victim fname
Access(fname)	< ----------	
……		
…….		
OK	---------- >	rm fname ln -> /etc/password fname
open("fname")	< ----------	

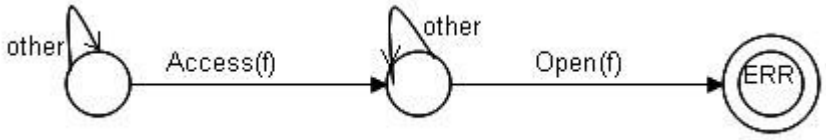
There is no simple, portable solution to solve this race condition.
However, there are workarounds available to solve this problem.
The solutions are:

1) Drop privileges temporarily, but this is not portable.
2) Fork a child, have the child drop privileges permanently and open the file, and then have the child pass the file descriptor back to the parent over IPC. This is portable, but inefficient.

The state diagram from the temporal ordering rule mentioned above can be shown as follows:

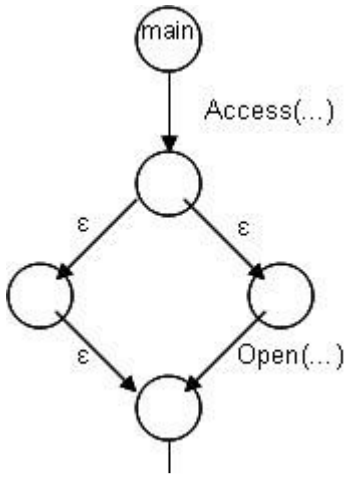


The above state machine if represented by M, represents a language defined by L(M).
We can also have a state machine as follows: this machine indicates that open(f) follows access(f) with other statements possibly occurring in between.



The following program can be converted into a state machine as follows:

```
if(access(…))
  open(….);
```



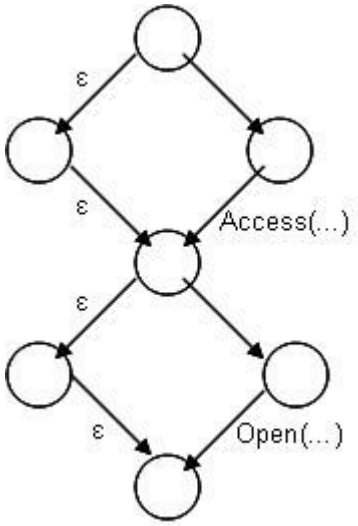
If the above state machine is denoted as ‘P’ and the language represented by it is L(P);
then we say that ‘P’ may have a bug if
$L(P) \cap L(M) \neq \Phi$

The following example demonstrates that, when we convert a program into a state machine, the language of statements accepted by the state machine will be a superset of the language of statements accepted by the program.

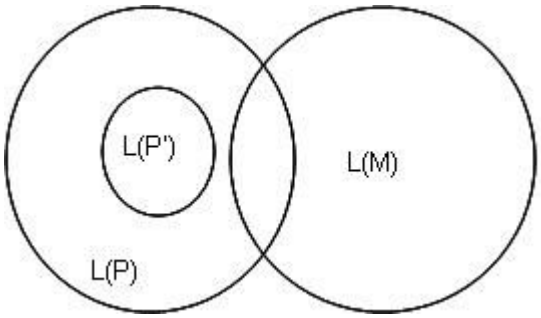
Example:

```
if(dump)
{
  access(…..)
}
if(!dump)
{
  open(…..)
}
```

The state machine can be shown as follows:



If the above state machine is represented by P', then we are safe (no bug) if $L(P') \cap L(M) = \Phi$

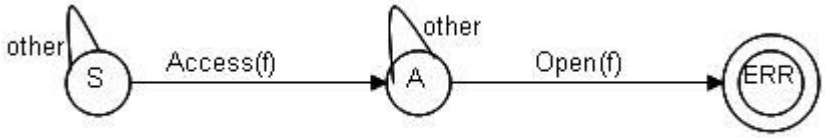


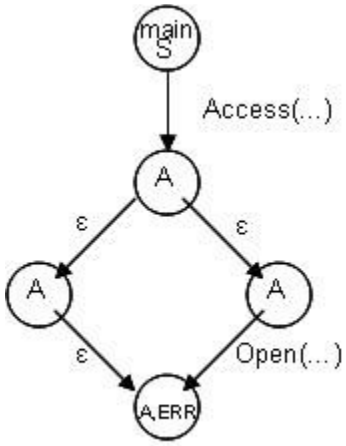
In the above diagram, we see that $L(P') \cap L(M) = \Phi$. Hence, we can state that we are safe.
The above is an implementation example of MOPS.

MOPS: - Specification: FSM
- Program: PDA
- Found a few dozen bugs.
- About 80% False Positives (FP) (Soundness Measure).

The high percentage of false positives can be reduced if we use MECA.
In the MECA algorithm, we push the states of the property state machine around the states of the program state machine.

If we name the different states that the FSM is in, we can actually represent the FSMs used above as follows:

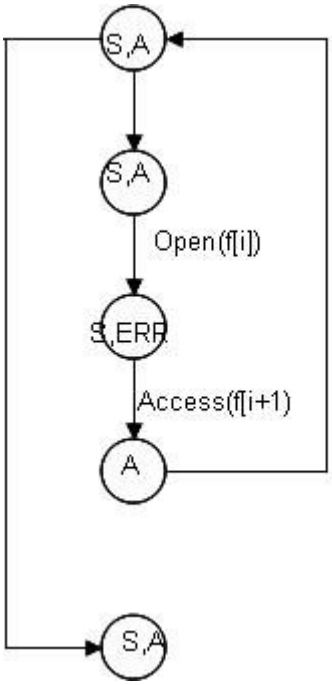




Consider a for loop where we open and access the file 10 times;

```
for(i=0;i<10;i++)
{
    open(f[i]);
    access(f[i+1]);
}
```

The state machine can be shown as follows based on the state diagram shown above:



Why does MECA have low FPs?

1) It is intraprocedural.
2) Liberally model data flow.
3) Use programmer checks to infer state.
4) Liberal Checks.

Example: