System Call monitor based Intrusion Detection Systems AKA Host Based IDS

- The attacker must make system calls to cause damage.
- Idea is to monitor system calls made by application to detect any deviation from expected behavior.

- Application Model describes the normal behavior of the application.
- Monitor monitors the system call made by application for any abnormal behavior.
- If found so, it terminates the application

Set based model for a “cp” (Copy) function: `{open, read, write, close}`
For “cp”, monitor checks that any call made by application is from the set or not. (i.e. read, write, open or close [in this example])

**Incorporating Ordering Constraints (Methods)**
- Model = Finite State Machines

- Model = Push Down Automata
How can we construct models..?
1. Manually
2. Trace Based
3. Statically

1. MANUALLY
   Construct manually

2. TRACE BASED
   • Run application in a safe environment where it can’t be attacked.
   • Collect traces of system call. (Done by the monitor)
   • We have to generalize it, (to prevent any situation which never occurred in tracing period), by constructing N-Gram model

**N-Gram Model**
It is a finite state machine.
State = last (n-1) system calls made by the system

Issues:
1. What should n be? //mostly n=6 does fine job.
2. How much trace data is needed? //more the merrier
3. False Alarms? (Caused by very large n + small trace data)

**Note:** If the attacker knows the model used by IDS, and there are too many states (unnecessary), then he can use these valid states to perform attack. This is called **mimicry attack**.

**Assumption:** Attacker has access to the model.

**Bottom-line:** We need the model to be as precise as possible but also complete.

It can be sometimes very difficult to do. Such as in case of interpreter.

### 3. STATICALLY (Staic Model Construction)
Control Flow Automaton
Its basically a Control Flow Graph, except that we use edges here instead of nodes.

Example: