Intrusion Detection System (IDS)

Model Construction:
- Static-- from source
- From execution traces
- Manual (bad way)

Type of models:
- Set model
- N-gram model
- State machine control flow automaton

State machine control flow automaton for functions:

Consider the following snippet:

```c
main (){  
    setuid(1000);  
    fnLog();
  
  //system() should NOT be executed when UID=0  
  system();  
  setuid(0);  
  fnLog();
 }

fnLog (){  
//some code  
write();
  }
```
State Flow diagram of the above program:

Problem with this model:

The first call to fnLog() have no issues. The flow in the graph is:
fnLog->fnLog0->fnLog1->main2

Now consider the second call to fnLog(). We have TWO paths that can be covered:
main4-> fnLog0->fnLog1->main5
OR
main4-> fnLog0->fnLog1->main2

IDS should not allow this. As are two path for a single code pattern.
How do we overcome the previous fault? How do we regulate the flow of the graph?

- We use push down automaton
- Push down automaton is represents every edge by \((\alpha/\beta/F)\)
  - \(\alpha \rightarrow \) The input from previous stage
  - \(\beta \rightarrow \) Check AND the pop the first item from the stack
  - \(F \rightarrow \) Push the item on to the stack

Q. What does \(\varepsilon/\varepsilon/2\) mean?
   - \(\alpha \rightarrow \varepsilon : \) Follow the graph
   - \(\beta \rightarrow \varepsilon : \) do NOT pop out anything from stack
   - \(F \rightarrow 2 : \) Push 2 (the return address of \(fnLog()\), set it to main2)

Q. What does \(\varepsilon/2/\varepsilon\) mean?
   - \(\alpha \rightarrow \varepsilon : \) Follow the graph
   - \(\beta \rightarrow 2 : \) Check if the top of stack is 2 and pop it out from stack
   - \(F \rightarrow 2 : \) do NOT put anything into the stack
Q. What if there is if-else statement in the program?
A. We do NOT care about if-else statement. We care ONLY about the system call.

Q. What if there is if-else statement which takes decision based on the output of a system call?
A. We should consider this scenario.

Consider the following function:

```c
fnFoo()
{
    if ()
        funct1() //call to system call
    else
        funct2()
}
```

- The return address of fcnt1 and fcnt2 MAY not be same
- If there are other functions called from these, we will have to send copy of stack to every function with modified return address.
- In the above case we will have to keep track of stack. It is done by null call. We check the flow of the graph using null call.