Review:

Every call will have a unique number

Null(2);
Log(..);
Null(-2);
Null(3);
Log(..);
Null(-3);

The system will make a decision in the system call.

Efficient Context Sensitive ID:

--Binary rewriting using null call insertion: null calls eliminate all ambiguity about stack and current function. But we cannot statically compute the cost of a particular null call insertion point, so it may possibly leading to high cost.

And also the execution context information is accurate only until an attacker takes control of the application.

Solution: Dyck model

Use a stack to record function call return locations. The monitor updates the stack precisely when that update reflects actual program behavior.

Null call log:

|   2 | -2 | 3 |

Queue for the later transition

Monitor can trace the queue.

*The Dyck calls will not instrument recursive call sites. And it will not monitor the function without system calls.*
The attacker can make the buffer overflow, then modify the queue.

Suppose the attacker takes over at E, he can write something in the queue to get the function it wants. When the queue is empty, and the attacker takes over, he can do something he wants by rewrite the queue. *But any modifications to the state kept in the application still must produce valid call sequences to be accepted by the monitor.*

**Mimicry attacks**

- Attacker mimics correct behavior
  - If we know there is a system call in the argument, we can modify that argument to defend the attacker. Using the data dependence graph (DDG).
  - For example: replace the `read(?, buf, ?)` with `read(=open, buf, {16, 128})`
  - We can also use branch analysis to prevent the mimicry attacks

How is the overhead of this efficient context sensitive?

Low overhead.

The evaluation metric measures model precision:

**Average Branch Factor:**

All potentially dangerous calls (all the out going edges)/number of ε updates performed by the monitor.

The best is to have one branch, so that the attacker has no other choice.

How about the multi-thread?

- Construct the model for the entire program
- Keep separate starts for each thread

The program we discussed above is a good program with bugs.

**Sand Boxes**

Goals: give untrusted programs limited rights. – *least privilege*