Intrusion detection system (IDS):

- Host based Intrusion detection system
- Network based Intrusion detection system

Goal:
- Prevent damage from attack

Key idea:
- Program under attack behaves differently
- Monitor system calls to detect weird behavior

Policy (ostia):
- Generic definition of allowed operation.
- It is applicable to the untrusted code.

Model (IDS):
- Abstracted model of a specific application behavior.
- Model is applicable to the trusted code.

Two metrics:
- False positives
- False negatives
Generating models:
- Static analysis approach: control flow graph (CFG) $\rightarrow$ NDFA
- Dynamic analysis approach: Log real traces.

Trace based IDS:
- Levenstein distance
- N-gram model
  - T1: open, read, write, close
  - T2: open, read, read, close
  - T3: open, read, exec

  3- Gram model: (open, read, write) (open, read, read) (read, write, close) (read, read, close) (open, read, exec)

  Most common use: 6–Gram model
  
  Assumption: When N–gram model is executed 1 to N-1 models also execute.

Advantage with trace based IDS:
- Can capture site specific configuration

Disadvantage with trace based IDS:
- False positives
- While testing execution traces could be missed.

Suppose the attacker is aware of the model/policy then attacker can do a mimicry attack.

![Diagram](image)

When intersection is not null it leads to mimicry attack.
Policy construction / Model construction:

Example:
Void foo (int fd , char *buf)
{
    read (fd , buf , 4)
}

Int main (int arg , char **arg)
{
    Int x;
    Open (arg[1],O_R_Donly);
    If (....)
        foo(fd, &x);
    Setuid();
    foo(fd, &x);
    close(fd);
}

CFG model for foo:
if (fd>=0) \Rightarrow This means that the program has bug but this cannot be captured by above CFG.

- The model also does not capture foo return.
- Context insensitive.
- Overall performance is terrible.

Efficient context sensitive intuition detection:

- Tracks values flowing throwing the program.
- Track influence of system calls return values on subsequent executions.
- Worked on binaries.
- Dramatically reduces the chances of mimicry attacks
- Performance problem is really gone.