HBIDS – host based intrusion detection system
- monitor syscalls.
- have model of correct application behavior.
  1. set of syscalls
  2. n-gram
  3. FSA
  4. context-sensitive
  5. Dyck model (requires code transformation)
- Monitor application to check against model
- Mimicry attacks

1. Mimicry attacks

While(...) {
    If(running_as_root)
        Setuid(-1);
        Read(…, buf, …);
        System(buf);
        If(running_as_root)
            Setuid(0);
    } Return;

Attacker code

System("/bin/ls");  → NOP
Setuid(0);
Read(1200,…,…);  → NOP
System("/bin/sh");
- common for mimicry attack to require hundreds of dummy calls.
- One researcher argues that one model had shortest mimicry attack > 300 calls

2. static and dynamic
   1) semantics of static models
      Application will follow a static model unless one of our assumptions while constructing the model is violated
      - Typically, these assumptions can only be violated by a memory corruption attack.

   2) dynamic models
      can capture other information e.g.) config files, and thus further constrain correct behavior.

3. Application Sandboxes
   - untrusted, potentially malicious applications
   - Goal : prevent damage caused by malicious code
     o monitor system calls
   - “Model” = user-defined policy -FPs
   - Sandbox policy
     o Write system files
     o Write personal files
     o Read personal/critical system files \( \rightarrow \) can be extended to “forbid network access after reading sensitive files”
     o Execute child programs
     o Listen
     o Kill other processes

   Problem1 : late discovery of policy violations
   Problem2 : dialog fatigue

4. Model carrying code (MCC)
Can model violate policy?

- Any policy violation after model passes check is strong indicator of malice.