Data collection for intrusion detection

Network-based Host-based

Indirect monitoring Direct monitoring

External sensors Internal sensors

Program

• A piece of code added to a program that monitors a specific variable or condition
  char buf[256];
  . . .
  log("len=%d", strlen(getenv("HOME")));
  strcpy(buf, getenv("HOME"));
  . . .

Embedded detector

• An internal sensor with logic added to detect a specific intrusion or attack
  char buf[256];
  . . .
  if( strlen(getenv("HOME")) > 255 ) {
    log("Buffer overflow");
  }
  strcpy(buf, getenv("HOME"));
  . . .
Using embedded detectors for intrusion detection

✓ Advantages:
  • Little extra resource usage
  • Very difficult to disable
  • Direct monitoring
  • Full access to data

✗ Disadvantages:
  • Very system-dependent
  • Need source code
  • Sometimes “too low”

Our implementation

- OpenBSD
- CVE
- Bugtraq
- SecurityFocus
- Logging mechanism

Over 100 detectors implemented so far. For example:
  • Land
  • Teardrop
  • Ping of death
  • WinNuke
  • Port scans
  • SYN flood
  • Smurf/Fraggle
  • Sendmail MIME buffer overflows
  • SSH vulnerabilities
  • IRIX buffer overflows
  • Apache buffer overflows
  • Solaris telnet DoS
  • TCP seq/qq prediction
Web server performance

Size of detectors
We measure Executable Statements Added or Modified

Chart includes 117 detectors
More than 90% are 6 ESAM or less!

What have we learned?

- Some patterns start to emerge (generic detectors)
- Stateless and stateful detectors
- Build an IDS based on what we need, not what we have

Still in the works…

- Detection of unknown attacks
- Detailed characterization of sensors and detectors
- Detailed description of implementation guidelines