Building accurate intrusion detection systems

Diego Zamboni
Global Security Analysis Lab
IBM Zürich Research Laboratory
Outline

- Brief introduction to intrusion detection
- The MAFTIA project
- Accurate intrusion detection systems
- Our work (GSAL @ IBM ZRL)
Why intrusion detection?
Experience shows that...

- ... users want features despite risks
  - javascript, shared files, e-mail attachments, ...
- ... it's hard to get rid of existing problems
  - unsupported OS release, TCP/IP, ...
- ... new systems contain “old” vulnerabilities.
- ... secure + secure ≠ secure.
- ... people make mistakes.
Intrusion detection

- Seminal paper in 1980 by Anderson
- “Modern” intrusion detection started in 1987 with paper by Dorothy Denning
- Is the new “hot buzzword”
  - Many commercial products
  - Many research projects
- No magical solution
Analogy: protecting your home

- **Prevention**
  - Locked doors, secured windows, wall around property, etc.

- **Detection**
  - Motion detectors, fire alarm, dog, etc.

- We need both

- We may also need a response
  - Call police, disable intruders, etc.
Diversity is a good thing

- Multiple specialized sensors

vs

- Single large monolithic sensor

But how to make sense of this diversity?
Characteristics of IDSs

- Intrusion Detection System
  - Audit Source
    - Host
    - Network
  - Method
    - Knowledge-Based
    - Behavior-Based
  - Usage
    - Sporadic
    - Continuous
  - Reaction
    - Passive
    - Active
Audit source

- **Network**
  - Headers or content of network packets
  - Behavior must be deducted
  - Subject to insertion/evasion attacks
  - Easier to deploy
  - No performance impact

- **Host**
  - Audit trails (log files)
  - ...or direct monitoring
  - Can directly observe behavior
  - More difficult to deploy
  - Possible performance impact
Method

- **Knowledge-based**
  (aka misuse detection)
  - Known events are suspicious
    (signatures)

- **Behavior-based**
  (aka anomaly detection)
  - Unknown events are suspicious
    (profiles)
Outline

- Brief introduction to intrusion detection
- The MAFTIA project
- Accurate intrusion detection systems
- Our work (GSAL @ IBM ZRL)
MAFTIA

▪ Malicious and Accidental Fault Tolerance for Internet Applications
  – How to build reliable and secure systems out of insecure components?
▪ European project
  – 3 years
  – 6 partners
  – 6 technical workpackages
Why is ID a part of the picture?

- Things **will** go wrong
- Even when our system recovers or resists failures and attacks, we want to know that something happened
- IDSs need to be protected too
  - How to prevent the IDS itself from being disrupted?
  - Make sure what the IDS reports is true
  - *Who watches the watcher?*
Outline

- Brief introduction to intrusion detection
- The MAFTIA project
- Accurate intrusion detection systems
- Our work (GSAL @ IBM ZRL)
(some) Limitations of existing IDSs

- Unable to detect unknown attacks
  - sometimes not even known attacks!
  - constant updating needed
- Large number of false alarms
- IDSs assume they cannot be corrupted
Characteristics we would like in an Intrusion Detection System

- Good coverage
- No false alarms
- Resilient sensors
- No training needed
- Capability for automatic updating
Outline

- Brief introduction to intrusion detection
- The MAFTIA project
- Accurate intrusion detection systems
- Our work (GSAL @ IBM ZRL)
Combining sensors to maximize coverage

- Goal: find the combination of IDSes that meets our requirements (e.g. 80% coverage, 80% rating precision)
- Optima at concurrent 100% coverage and 100% rating precision
Dealing with false alarms (1/2)

- **Problems:**
  - >95% false positives!
  - Alarm flood worsens as number of signatures rises.

- **Conclusion:**
  - This noise makes it impossible to correlate events of those sensors.
  - We need to figure out how to automatically remove it.
Dealing with false alarms (2/2)

Data Mining yields ZCE rules!

Fix, Block, Filter, ... to Reduce Alarm Load

Summary
1. Pattern 1
2. Pattern 2
3. ...

Understand Root Causes

Data Mining key facts:

✓ Self-tuning: No human intervention!
✓ Stellar results: 2-4 pages long, extremely easy to understand!
✓ High impact: 90% alarm reduction!
✓ Validated: On >100 million alarms!
Building good intrusion detection sensors

- **Host-based sensors**
  - Increased accuracy and access to good data

- **Behavior-based sensors**
  - Can react to unknown attacks
  - They tend to generate lots of false alarms
DaemonWatcher

- Detect suspicious behavior of UNIX processes
- Principle:
  - A process is characterized by the sequences (patterns) of system calls it generates
  - The patterns can be used to model the normal behavior of a process
  - Intrusions are assumed to exercise abnormal paths in the executable code
DaemonWatcher: related work

- **UNM**
  - The first to propose this approach
  - Used fixed-length sequences of system calls

- **CMU**
  - Analyzed the choice of sequence length
Exorcist

- Detects code insertion attacks
  - Buffer overflow, parasitic viruses
- Host-based
- Behavior-based, but no training phase
  - Profile is built by static analysis
- Components:
  - Analysis phase
  - Sensor
Buffer overflow
Parasitic virus
Exorcist overview

Analysis phase

Program executable

Analysis

NFA

Runtime

Program

open
write
close

Pattern matcher

Operating system

Alarm
Analysis phase

- The executable is analyzed using no external information
- Binary code transformed to Control Flow Graph
- End product is an NFA

- Very limited data flow analysis
- No dependencies on source code availability
- No training required

- Compiler, library and processor dependencies
  - e.g. Assumes static linking, ELF files, gcc, glibc 2.x
int main(char **argv, int argc) {
    int fd = open("test", O_RDWR);
    char *stuff;
    if (fd >= 0) {
        stuff = "Howdy";
        write(fd, stuff, strlen(stuff));
        close(fd);
    }
    return 0;
}
The Exorcist sensor

- Implemented in the Linux kernel
  - User-space sensor for testing (based on strace)
- Match against stream of syscalls, using an NFA
- Syscall parameters are not considered
- Signals caught and handled separately
- Threads are currently not handled
Sample Exorcist alarm
Other approaches

- Closely related work:
  - stide (UNM), DaemonWatcher (IBM ZRL)
  - David Wagner's static analysis (UC Berkeley)
- Policy-based protection:
  - BlueBox (IBM Watson)
  - RSBAC (Amon Ott, rsbac.org)
  - LIDS
- Other buffer overflow protection mechanisms:
  - StackGuard, StackGhost, PAX, etc.
Exorcist benefits and drawbacks

✓ No training needed
  – Only update profile when program changes
✓ Sensor resistant to attacks
✓ False-positive free by design
✓ Detects new attacks
✓ Can potentially stop attacks
× Prone to mimicry attacks
× Currently requires patching the kernel
Exorcist present and future

- Being tested internally
  - Performance tests, accuracy tests
- Windows version? (DLL + threads)
- Product or Open Source?
- Improve analysis phase
- ob. Autonomic Computing
  - can identify new attacks
  - can automatically protect new programs
  - can be part of an immune system infrastructure