Reexamining Intrusion Detection

Eugene H. Spafford
Director, Purdue University CERIAS
http://www.cerias.purdue.edu

Grad researchers:
Benjamin A. Kuperman & Diego Zamboni

Outline

- Basic premises
- Origins and definitions
- Shortcomings
  - One contribution: Target Monitoring
  - Instrumentation of targets
  - Interposition of calls and activities
  - Distribution of detection: AAFID
- Future work and observations
Misuse & Intrusion

- Security is defined by policy
- The goal of security is foremost to enforce policy
- Violations of policy need to be detected
  - Violation by non-users are intrusions
  - Violation by current users are misuse (overloaded term)
  - Intent plays a major role

Intrusion Context

- Intrusions can occur
  - Software faults
  - Hardware faults
  - Misconfiguration
  - Operator error
  - Unexpected interactions
- Changes often occur
  - Often the result of unknown flaws
  - In some cases, intrusion is professionally done and represents a grave danger
**Misuse Context**

- **Misuse can occur**
  - same causes as intrusion
- **Changes occur sometimes**
  - Often result of poor controls
  - May be result of intrusion
  - Accidents also occur
  - Insider misuse almost always leads to larger losses

**Big Myth**

The biggest threat is from the “outside.”

Reality: Only about 3% of serious incidents in a typical business environment are from “outside.”

- Bugs and errors: 65%
- Outsider attacks: 3%
- Disasters: 13%
- Insider misuses: 19%
Goal of IDS

Detect intrusions when they occur, and if possible, detect attempts at such violations before they succeed.

Either detect misuse as it is attempted, or detect variations from acceptable behavior.

Neither of these is achievable with 100% confidence. Therefore, seek to minimize errors.

Minimize Errors

- **False positive**
  - Alarm for normal behavior
  - Usually sysadmins
  - Results in gradual loss of attention
  - May lead to setting thresholds too high

- **False negative**
  - Misses violations
  - Difficult to detect as a problem

Both can occur in the same system!
Reexamining Intrusion Detection

Traditional Host-based Approaches

- **Misuse detection**
  - Using audit data, identify behavior in violation of policy (e.g., known attacks). Specifically, identify inappropriate behavior.

- **Anomaly detection**
  - Using audit and other data, identify profiles of “good” behavior and then flag any exceptions.

- **Hybrids**
  - Do some of each

Basic model by Dorothy Denning in 1987, based on ideas of Jim Anderson in 1980.

Representative Systems

- Behavioral patterns
  - NSM, W&S (USAF, DoE)

- Language-based matching
  - ASAX (Siemens)

- Compiled-matching
  - Stalker/Netstalker, NetRanger (Haystack Labs, WheelGroup/Cisco)

- Rule-based systems
  - CMDS (SAIC)

- AI Techniques
  - U.C. Davis group (DoD, ARPA)
  - NADIR (LANL)

- Pattern/event engines
  - IDIOT (Purdue COAST)
Misuse Drawbacks

- May miss new attacks
- May not work in real-time
- Needs constant update
- Difficult to capture all variants
- Need expert to build

Anomaly Drawbacks

- May generate high rate of false positives
- May not properly capture policy
- Needs constant update. but can be automated
- Needs training with specific data
A Different Approach

Target & goal monitoring:
Tripwire, IDIOT, AAFID

- Instead of (or in addition to) looking for bad behavior or monitoring good behavior, we monitor the “targets” of potential attack.
- We can also install “tripwires” to signal an attack.

Other examples: Stoll’s SDI files, Wily Hacker “jail”

What to monitor

- Change & access times
- Permissions and owners
- Sizes, links, type
- Contents
  - Message digests
  - CRCs
  - Custom
- Privilege level
- System state
- Thresholds
**Interposition of Critical Operations on Targets**

- **Intercept selected library calls**
  - Examine, record, alter its arguments
  - Examine or modify its return value
  - Prevent the call from occurring
  - Call a different function than the one specified
- **Maintain state**
  - Keep track of data between library calls
    - Number of times a function is called (e.g. crypt())
    - Arguments used (e.g. filenames, buffers)
    - Amount of memory allocated (for limits)
- **Implement the above without needing recompilation of the application**

---

**Why interposition?**

- No need to recompile the system library or application executables
- Able to span multiple shared objects
- Do not have access to all of the source code
- A single interposed library can be deployed on a variety of platforms
- Easy to configure for selected target monitoring
Expanding the Scope

Typical CMAD/ID systems are
- Large
- Slow
- Single point of failure
- Single point of attack
- Difficult to extend
- Some systems may be trained or anticipated

We need something that does not have these drawbacks

We want a system that will scale for a large network

Complications in a Network

- Single host vs. Network (Scaling)
- Audit Trail Collection & Standardization
- Reporting & Actions
- Testability
- Tuning
- Performance
- Parallel Attacks
Worst case

- Penetration is slow, over weeks or months
- Each step is done from a different machine, to a different target
- Each step is very small
- Penetration is almost identical to normal behavior
- Attack partitions system to isolate detector from monitor

Goals of the AAFID project

- Provide a framework for distributed data collection and analysis
- (Semi-)Autonomous agents & alarms
  - Independent entities
- Scalability, configurability and extensibility
- Graceful degradation of service
- Combine the best of host-based and network-based intrusion detection
AAFID2 prototype

- Road-test the architecture
  - We want people to use it
- Focus on usability and flexibility
  - Easy portability
- Run-time distribution of code
- Easy to add monitors
- Little focus on performance
  - However, it has low overhead!
- Provides infrastructure for development
- Uses pipes and TCP for communication
- Implemented in Perl5

Graphical User Interface
Per-host agent view

Current status

Prototype distributed to the public
- ftp://coast.cs.purdue.edu/pub/coast/AAFID/

We encourage you to...
- try it out
- play with it
- extend it
- modify it
- give us feedback (aafid-feedback@cs.purdue.edu)
**AAFID Detects**

- ARP cache poisoning
- Writable user and configuration files
- Suspicious sequences of commands
- Accesses to network services
- Health of system services (e.g. NFS)
- Repeated login failures
- Configuration problems in ftp and WWW servers

…more coming

**Future work**

- Develop many more agents and filters
- Extensions to the architecture
  - More complex data collection and filtering mechanisms
  - Identify different types of agents
  - Generalize the concept of “agent”?
- Port to other architectures (NT) and languages (Java?)
- Data reduction and user interface issues
- Security and failure resistance mechanisms
- Performance issues
  - Measurements and profiling
  - Low-level implementations
Reexamining Intrusion Detection

Additional Work

- Integrate interposer into AAFID
- Revise current library interposer
  - add additional signatures and audit information
  - interpose on additional system calls and targets
  - consider longer term race conditions (possibly use kernel space)
- Consider integration of signature patterns for known attacks (especially SUID and daemon binaries)
- Improve logging system
- Generate a specification language for configuration

Future possibilities

- IDS is the current “firewall” craze
  - Before that was content filtering, viruses, etc.
- Strike-back is gaining interest
  - A dangerous concept
- Forensics is being neglected as a field of study
- Identification methods are needed
Closing thoughts

- If systems will built with care and good software engineering practice, we would have a greatly reduced problem set.
- By definition, anything that happens on a system without a policy is not a violation.
- Wireless systems, mobile code, VPNs, ADSL/cable modems, and more will make this field much more interesting in future years.

Thank you!