

Explaining and Recovering from Computer Intrusions: Status 2/97



Douglas B. Moran

**Artificial Intelligence Center
SRI International
333 Ravenswood Avenue
Menlo Park CA 94025**

Overview



n **Apply AI Technology to Network-based Intrusions**

- diagnosing
- explanation and reporting
- repair and recovery: automated security manual

n **Tool:**

- manages other tools (esp. OTS) for data collection
- rule-base to guide/advise user
- add additional tools as agents
- add intrusion scenarios

n **Approach: start with common cases and build outward and upward**

Initial Focus: Rootkit



- n **Widely available, commonly used cracker tool**
- n **Involves many of the technical issues for diagnosis of intrusions**
 - **Multiple entry points to diagnosis**
 - **Alternative sources of information**
 - » **Cheap vs expensive**
 - » **Direct vs inferred**
 - **Incomplete information**
 - » **attack may not use all components of Rootkit**
 - » **tools to delete and hide information**

Rootkit: Background



n Modified Commands

- process listings (*ps*)
- file listings (*ls*, *du*)
- network status (*ifconfig*, *netstat*)

n Modify log files

n Modify file info

- dates
- checksum

Multiple Entry Points



n Example 1:

- user found hidden process
- suspect *PS* substitute
- confirm
- suspect Rootkit
- look for other substituted commands
- suspect log files modified

n Example 2

- user finds hidden file
- suspect *LS* substitute
- confirm
- suspect Rootkit
- look for ...

n Example 3:

- find Rootkit source

n Example 4:

- find Rootkit config files

Multiple Entry Points: Bi-directional Paths



- n **“Forward”**: hidden process -> look for its name within *PS* binary (strings command) either directly or indirectly (find config file)
- n **“Backward”**: substituted *PS* -> use *strings* to find config file or process name -> find process or file (executable or source)
- n **Different costs**: “forward” easier: finding known name of process easier than finding that name among miscellaneous strings
 - Note: finding config file name is common to both
- n **Limitation**: each direction coded separately

Alternate Sources of Evidence



n Example: substituted commands

- Other commands producing similar output:
 - » eg *PS*, *SPS*, *TOP*
 - » common extensions, but not in official release
- Dates: compare to siblings and parent
- *Strings* (compare to “official” copy)
- Checksums: cheap if on-line, expensive (time delay) if they need to be mounted, especially if on tape

n Example: altered log file

- cheap: *wtmp* start-end mismatches
- cheap: *lastlog* vs *wtmp* inconsistencies
- more expensive: various more sophisticated analysis

Alternate Lines of Attack



n **Example: to hide objects (processes, files)**

- modified command
- modified string comparison routines in library

n **Example: hidden process**

- may not run all the time

Incomplete Evidence



n Declaring “Rootkit suspected” is only a guide for what else to look for

- Variants of Rootkit
- Incomplete use of components
- Similar functionality in other attack kits/scripts

n Level of confidence

- suspicious process with 4 letter name, *strings* command finds name in *PS*, but dates OK, and checksum info off-line.
- Enough to trigger search for related evidence, but defer costly steps to confirm

Implementation Approach



n Middle-out

n Initially:

- Assume that user is suspicious that there has been an intrusion and has supplied starting point to system (eg, hidden process)
- Data requests to very low-level system components are stubs (hand simulated by developer)

n Low-level components

- Prefer: borrow, reuse, modify
- Otherwise: create (keep simple)

n Design for portability to other platforms

System Engineering Issues



- n **Prototype to be experimental testbed**
- n **Tradeoff on size of procedures (Acts)**
 - efficiency vs reusability
 - comprehensibility to and customizability by end-user
- n **Tool boundaries**
 - Primary reasoning engine may be mismatch for some of the diagnosis
 - Multiple reasoning engines?
 - Split of rules natural?
- n **Scheduling: cost, benefit, priority**
 - automatically/dynamically determined?
 - extent of human interaction

System Engineering Issues



n Coordinating rules for bi-directional analysis

- automatically generated: stubs, templates?
- automatically check: presence, parallelism

n Bundling expensive operations:

- from queue
- from rules, but not on queue
- Example: retrieving checksum data from tape

Conclusion



- n **In early phase of building prototype**
- n **Local expertise being incorporated into rules**
- n **Even “simple” example of *Rootkit* has produced a complex web of relationships and alternatives**