Classifying Vulnerabilities

Matt Bishop
Department of Computer Science
University of California at Davis
Davis, CA  95616-8562
email: bishop@cs.ucdavis.edu
phone: (916) 752-8060

Where We’re Going

• Develop a scheme to classify vulnerabilities
  – Refer to any such scheme as a VCS (Vulnerabilities
    Classification Scheme)
To get there:
• Need an agreed-upon vocabulary
• Need some method of organizing the data

October 23, 1996
Requirements for Classification System

• Flexible
  – Scheme must serve different needs, environments and systems

• Extensible
  – New systems have new vulnerabilities and may introduce new classes of vulnerabilities
  – New systems have old vulnerabilities arising in new ways

• Useful
  – Easy to look up vulnerabilities based on criteria not known to the designers
  – Easy to find similar vulnerabilities, again where the metric for “similar” is not known to the designers

My Definitions

• Partition states into authorized and unauthorized
• vulnerable state: authorized state from which an unauthorized state can be reached
• compromised state: state so reached
• attack: sequence of authorized state transitions ending in a compromised state
• vulnerability: characterization of a vulnerable state distinguishing it from all non-vulnerable states

October 23, 1996
Approach

- Decompose vulnerability into characteristics
  - want a minimal set, \textit{i.e.}, if any of the characteristics are false, you don't have a vulnerability
- Characteristics may be at any level of abstraction and from any point of view
  - more on this later

Example: \textit{fingerd} Flaw

- Input from user put onto stack without bounds checking
- If input too long, overwrites PSW and return address
- So … load your favorite machine code into the buffer, and overflow, setting return address to address of buffer

Characteristics:
PD6, failure to validate type of object (input)
PI3: failure to check array or buffer bounds
E5: improper entered data
Example: *ypupdated* Flaw

- Authenticate remote use as root using Diffie-Hellman with 133-bit private key
- If no *root* key, assume remote user is generic *nobody* and use that key (preconfigured, well known)
- Authentication succeeds for *nobody*, but no indication it was the *nobody* user (so actions proceed for *root*)

Characteristics:
- PD2: improper setting of programming defaults
- PI5: improper choice of operand

Level of Abstraction

Absorbed into characteristics
- if design flaw, use design-oriented characteristics
- if implementation flaw, use implementation-oriented characteristics
- notion of containment: if $A \subseteq B$, then $B$ is a refinement of $A$, or $A$ is “more generic” than $B
Point of View

A qualifier to characteristics
• process(es) being attacked
• process(es) doing the attacking
• operating system
• possibly others?

Example: \textit{fingerd}

• \textit{fingerd} process
  – PD6, PI3, E5 (seen before)
• attacking process
  – PI5, improper choice of operand (input too long)
  – E5, improper entered data (input; it’s too long)
• operating system
  – PI9, unauthorized access to a portion of memory (writing to what should be protected, the return address and PSW)
  – PI1, TOCTTOU flaw (return address changes between storage and use)
  – E6, improper object permissions (can execute data)
Performing the Classification

- Problem: terms **not** canonical
  - Highly unlikely we’ll ever get a universally agreed-upon vocabulary for these
  - Relationship of terms may not be clear to a developer who is not an expert in the nature of vulnerabilities (or knows very little about security!)
- Answer: create a thesaurus
  - Organizes terms to enable classifier or user to find related terms quickly
  - Independent of organization of data

Approach suggested by Mike Raugh of Interconnect Technologies; work done with him and Diane Hillmann of the Technical Support Services, Olin Library, Cornell University and a member of Machine-Readable Bibliographic Information Committee of the American Library Association

Example Page from VCS Thesaurus

Program: Implementation

- **TOCTTOU style flaws**
  - **UF Time of check to time of use style flaws**
  - **UF Flaws, TOCTTOU**
  - **UF Improper change**
  - **UF Improper deletion**
  - **NT Interprocess communication**
  - **NT File accesses**

- **File accesses**
  - **UF Accesses, File**
  - **BT TOCTTOU style flaws**
  - **RT Interprocess communication**

October 23, 1996
(Example Page con’t)

interprocess communication
  UF Communication, Interprocess
  BT TOCTTOU style flaws
  NT access open
  NT stat open
  RT File accesses
access open
  UF access followed by open
  BT Interprocess communication
  RT stat open
stat open
  UF stat followed by open
  UF status open
  BT Interprocess communication
  RT access open

Comparison to Other Taxonomies

• PA, RISOS
  – these had very generic categories
  – as used, seemed to put all flaws into exactly one class
    (although no reason flaws could not be in multiple classes)
  – point of view, level of abstraction ignored

• Aslam
  – specific to flaws in UNIX systems and C programs, so
    everything at implementation level
  – decision procedure put flaws into exactly one class, thereby
    obscured nature of flaws with multiple characteristics

• Landwehr
  – built on PA
Future Directions

- Build a thesaurus
- Acquire network infrastructure systems (routers, etc.)
- Extend security checking tool slint to look for other vulnerabilities
  - Currently does race conditions, type checking
- Automated methods for including data into vulnerabilities database
- Focus on forensics of attack tools
- Obtain more systems, especially older systems, to help build historical record

Sponsors

- United States Air Force
  - Work on the taxonomy, database, tool building
- NIST (work in conjunction with Interconnect Technologies, Inc.)
  - Work on the digital library aspects of the database, especially the thesaurus and the representation of data
- SRI International
  - Work on some aspects of the database