Cooperative Vulnerability Database

Lingfeng Ma	lingfeng@cerias.purdue.edu
Salvador Mandujano	sam@cerias.purdue.edu
Guangfeng Song	songg@cerias.purdue.edu
Pascal Meunier	pmeunier@cerias.purdue.edu

NOT	full disclosure
NOT	need to know
NOT	state information
NOT	public information
Web	accessible

Other Databases

Bugtraq:	Little QA
	Standard search options
	No classification of vulnerabilities
CERT:	Impose disclosure time
	More concerned with incidents
We:	Information review process
	Smarter search criteria
	Vulnerability taxonomy
	Better mechanisms for disclosure
	Cooperation and sharing

Vulnerability Workshop

3 main models:

- Open model
- Centralized model
- Federated model
- * Balkans/Status quo

The CoopVDB:

- A central repository is maintained
- Multiple entities contribute to the contents
- Information is made available in a controlled manner

Vulnerability information sharing

- Reasons not to share
  - ”not a problem until it is exploited”
  - Leave well-enough alone
  - Sharing encourages attacks
  - Immediate cost:
    - Our customers could get hurt
    - It’s expensive to fix vulnerabilities

- Reasons to share
  - Security is important to customers
  - Unknown risks are scarier
  - Information warfare
    - others are spending resources on finding vulnerabilities against you
  - Learn from mistakes
  - Motivate vendors to fix vulnerabilities
  - Indirect reward for responsible sharing

- Wrong: Should I share?

- Right: When should I share, and with who?

- How do I get credit for doing the right thing?

When to share

Time periods:

1. Pre-patch, pre-workaround
   - There are no patches or known workarounds
   - Sharing vulnerability information with everyone is dangerous

2. Pre-patch, known workaround
   - There are no patches available, but a workaround has been found
   - Sharing vulnerability information with everyone is less dangerous

3. Post-patch, pre-installation
   - The patch has been released by the vendor, but very few people have had time to install it
   - Sharing vulnerability information is necessary to motivate the uniform installation of patches

4. Post-patch, post-installation
   - Most people have installed the patch, and the fix is now included in the normal release
   - Vulnerability information is of academic interest

Intended Usage

- Share within trusted groups:
  - Inside a company
  - Across partner companies
    e.g., CERIAS sponsors

- Let vendors have some control over disclosure
  - Submit vulnerabilities to the editor
    representing the company who made the
    product

  - Nominate a reviewer from that company
  - Withold vote until workaround is available.

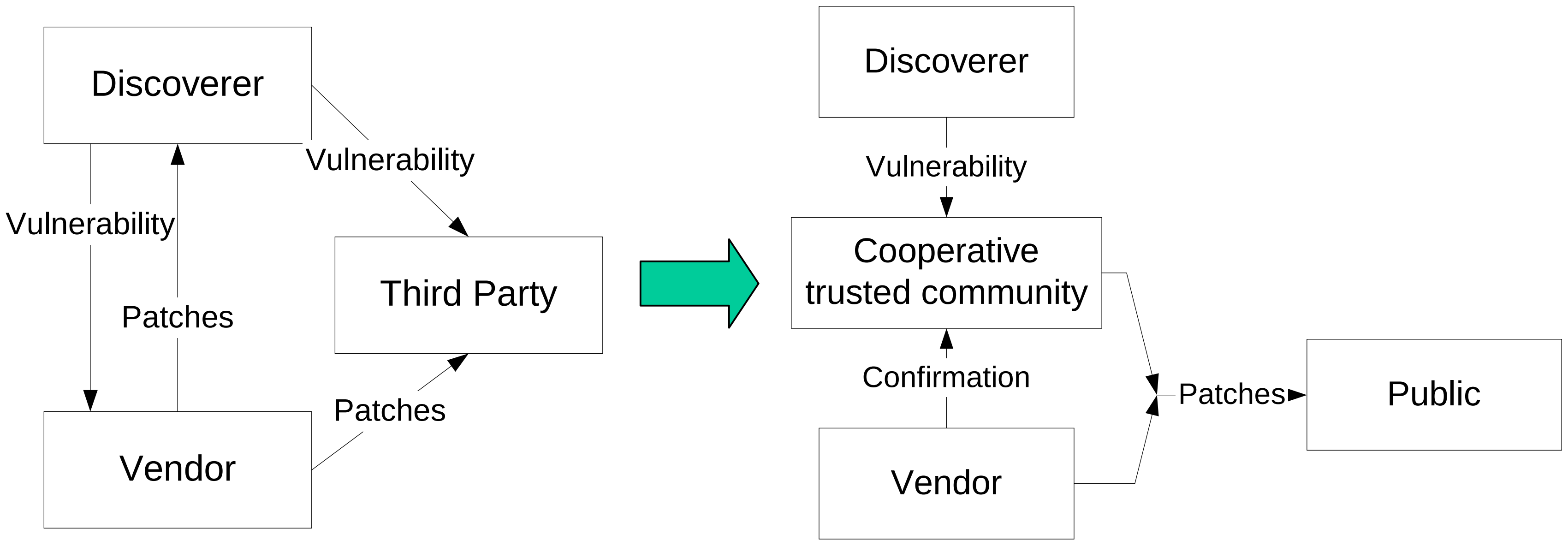
- How to convince companies to use it?
  - If no vendor participation, disclose to trusted
    community immediately after review

- How to convince finders to use it?
  - Time-stamped channel
  - Kudos

- Primer: CERIAS uses it.

- Dangers: community pollution
  - Leakage outside trusted group
  - Fragile trust
  - Trust drift (a -> b -> c -> d does not imply
    a -> d)

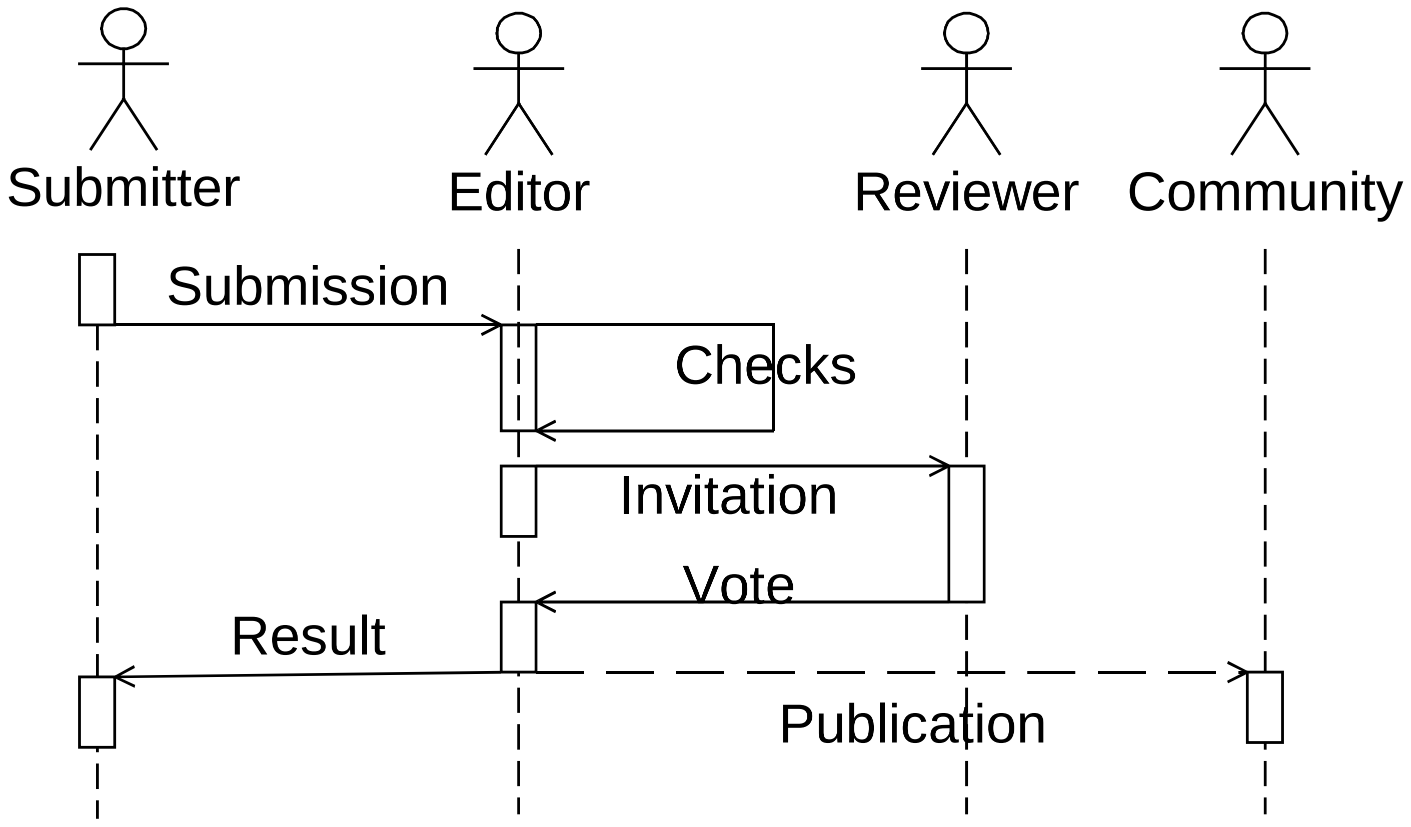
Extended Model of Disclosure



Key points:

- Information need to be shared among trusted parties
- Information validation and quality control are important

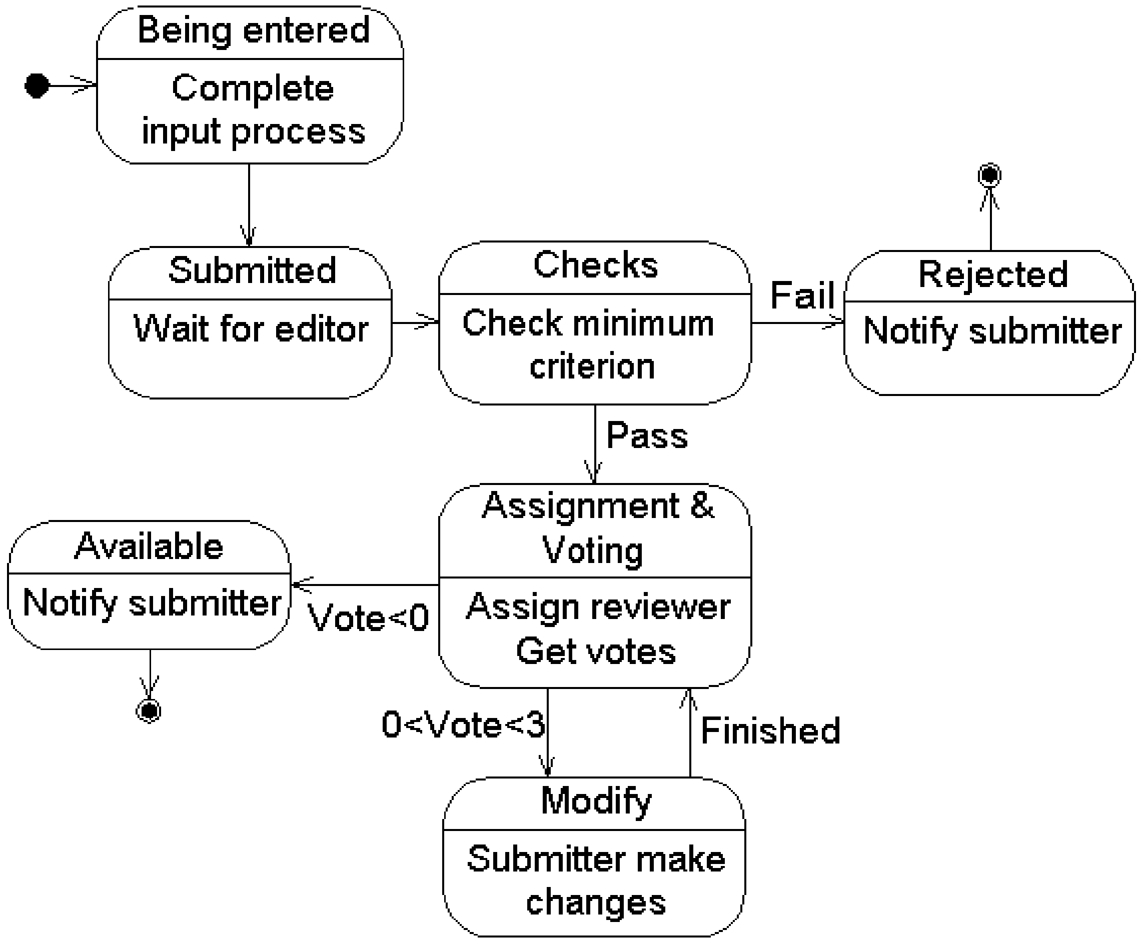
Collaboration



Vulnerability TYPE

- Grouping and classification
  - Features derived by classification
  - Meaningful identity created by grouping features
- Practical usefulness
  - Easy to understand, remember, and faster input
- Example:
  - Nature object, method, input, effect

Vulnerability workflow



Future Enhancements

- Submitter rankings (Top Ten)
  - # accepted submissions
  - “Stars” as suggested in Ranum [CSI XVII, Number 1, 2001] (“Towards an economy for vulnerability disclosure”)

- Pre-flight checks
  - Patches applied?
  - Vulnerability already known?
  - Try to reduce effort for participating vendors

- To limit trust drift:
  - Database owner nominates editors
  - Editors nominate only normal users

- Feed the CVE with good information

- Public version
  - Post-patch disclosure
  - Linked to announcement service (e.g., Cassandra)

Technical Aspects: Overview

- Developed with PHP and MySQL
- Secure connection (SSL 3.0 or TLS)
- Small functionality-based modules
- Library of utility functions
- Code review

Technical Aspects: Validation of Submission

- Problem: Submitted input fields in HTML codes may subvert the system
- Solutions:
  - All inputs run through “sanitization” routine before entering the database
  - No improper operation will be performed once the data is stored onto the database
  - The functionality of the system will not be affected by values being read from the tables

Technical Aspects: Cookies

- Session log table: Record successful authentication and session id
- Cookies: Identify session
  - Randomly generated large number as session id
  - Checked at the beginning of every script
  - Must match username / sessionid pair in the sessionlog table

Technical Aspects: Access Control

- Mandatory Access Control
  - Clark-Wilson model
  - Verify troplet {userid, action, vulnerability}
  - Done as necessary and for customized interface

Technical Aspects: Miscellaneous

- Uniform PHP coding style
- Display: header, footer and navigation
- Standardized error handling routine
- User-friendly interface
- No java/javascript/ActiveX, fewer vulnerabilities