IFIP WCC Topical Day on
Open Source Software in Dependable Systems

Trusting Strangers
Open Source Software and Security

26 August 2004

Presented by
Vipin Swarup
The MITRE Corporation

Carl Landwehr
Institute for Systems Research
University of Maryland

Outline

1. Software and Trust
2. Certifying Security
3. Open vs. Closed

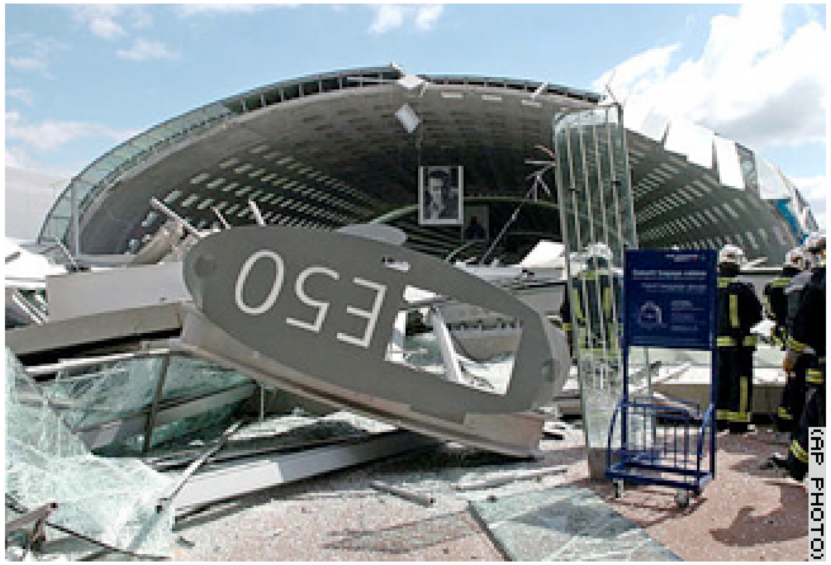
Visible (inspect-able?) systems

Less visible

- Even a basic car like a Citroen 2-cv hides a lot under the hood
- Consider a modern airliner

What about this building?

Or this one?



(CDG terminal 5/23/04)

Or your microwave oven?

We rely on many anonymous strangers to design, build, deliver, and maintain critical systems

But it's not blind trust

- We have building codes and inspectors
- We have safety regulations
- We have product liability
- We have publicity when accidents and failures occur, and consumers react

Software is an unusual artifact

- Little physical substance, but can convey sensitive information and control significant energy
- Significant costs in design and implementation
- Low cost of replication
- Small changes to its representation can yield major behavioral changes to systems
- Usually licensed, rarely sold
- Licenses typically relieve producer from product liability

Certifying Software Systems

- Safety certification:
  - Baseline assumption: incompetence, not malice
  - Typically a combination
    - Development process controls
    - Inspection and testing
  - Additional strong economic factor:
    - consumer response to accidents
  - Status: not perfect, but reasonably effective

Certifying Security

- Baseline assumption: malicious attacker
- Common Criteria (CC) scheme
  - Permit separate specification of function and assurance requirements
  - Develop Security Target (specification)
  - Develop Target of Evaluation (implementation)
  - CC Testing Lab checks whether TOE meets ST
- Issues:
  - Unless relatively high assurance levels are requested, source code will not be reviewed by lab
    - And most flaws exploited in today's attacks are in the implementation, not the spec
  - Scheme remains component-oriented
    - Security is a system property
  - Cost-effectiveness unknown

Open vs. Closed

- Should we encourage/allow/disallow the use of open source software in security-critical applications?

  + Arbitrary tools can be used to investigate, modify, re-link, rebuild, analyze, the software
  + Third party can examine in as much detail as you can afford

  but

  - Liability for the results will rest with you
  - Lf you don't review the software, there's no guarantee anyone else has either
    - most of those "thousands of eyes" lack expertise and interest
    - some of them might be malicious

Is closed source better?

- Carries the producer's economic interest in the product – a potent factor
  + Can drive control of software development
  + For large companies, reputation is a factor
- But
  - Not much product liability for licensed software either
  - Hackers find flaws even without source access

Conclusions

- Caveat emptor
  - Neither open nor closed source produces "bullet-proof" software without specific investment for that purpose
  - Exposing source doesn't automatically improve its security properties
  - Neither does hiding it
- Seek product and architectural assurance
  - Process assurance is uncertain in a world of outsourced component software modules
- Exploit what you know, and what know you don't know
  - If you use open source, consider whether to reconfigure or rebuild
  - If you purchase closed source, investigate the developer's processes, motives, independent evaluations
  - Build system architecture taking these into account

Thank you!

Discussion?

Acknowledgements:
Thanks to Michael Hicks, U. Md for discussions
Thanks to Vipin Swarup, MITRE for presentation!

"I still don't have all the answers, but I'm beginning to ask the right questions."