# Risks of Untrustworthiness

Peter G. Neumann
Principal Scientist
SRI International ComputerSciLab
Menlo Park, CA 94025-3493
Neumann@CSL.sri.com
http://www.csl.sri.com/neumann
Tel 1-650-859-2375
ACSAC 2006, Miami, 14 Dec 2006

# Complexity

- "Everything should be made as simple as possible, but no simpler." Albert Einstein
- Simplicity is highly praised, but Security is Inherently Complex. Oversimplifying it creates flaws.
- How can we manage complexity?

## Security

- Security is a set of end-to-end total-system emergent properties, some of which must be avoided.
- Strength in Depth is desirable, but we have Weakness in Depth.
- Achieving better security is only part of the problem.
- We need Trustworthiness.

# Trustworthiness

- Trustworthiness of S (a system, network, subsystem, enterprise, ...) implies S is worthy of being trusted to satisfy its specified requirements R (for security, reliability, human safety, system survivability despite a realistic range of adversities, ...), with some measures of assurance Q.

# Trustworthiness is Holistic 1

- Holistic approaches consider systems and enterprises in their entirety in the context of their environments, lifetimes, and total ranges of uses.
- Trustworthiness involves many end-to-end emergent properties, many of which must be avoided.

# Trustworthiness Is Holistic 2

- **Trustworthiness is Pervasive. Systems need to satisfy all critical requirements, not just security.**
- **Trustworthiness is highly multidimensional. It is not a local property, especially for applications. Total-system analysis is needed.**

# Trustworthiness Is Holistic 3

- Security, reliability, and other critical requirements interact, and can be incompatible.
- Effects of flaws and bugs can propagate widely.
- Application security is easily undermined by poor OS security.
- Outages must be anticipated.

## Systems Need Holistic Analysis

- **Energy: future-oriented/ short-sighted optimization**
- **Agriculture: natural/industrial**
- **Health care: prevention/"cure"**
- **Systems: principled/unprincipled**

**See PGN, Holistic Systems,** *ACM SIGSOFT Softw.Eng.Notes,* **Nov. 2006**

# Energy

- Renewable resources (solar, wind, biomass, hybrids) can be viable if considered holistically.
- **Fossil fuels are short-sighted, nonenvironmental, nonrenewable, contribute to global warming. Nuclear waste has long life.**

# Agriculture

- Sustainable agriculture uses natural fertilizers/pest-controls, crop rotations. It is healthful.
- Industrial agriculture causes soil depletion, toxic runoffs, worker and consumer health problems.

# Health Care

- Alternative/preventive methods are holistic and environmental.
- "Modern" medicine seeks quick fixes that suppress symptoms rather than eliminating causes. It may be iatrogenic, trigger bacterial mutations, ...

# System/Network Development

- Principled system development has many long-term benefits.
- Bad practices include monolithic nondecomposable systems, poor software engineering, sloppy software, unsafe languages, overdependence on patches, and create many problems.

# Avoiding System Risks

- Build constructively trustworthy systems with predictable composability and interoperability.
- Ted Glaser: "A modular system is one that falls apart easily."
- Modularity is not enough; we need encapsulation, compatibility, interoperability, noninterference...

# Holistic Analysis Is Needed

- **Principled development of trustworthy systems must be demonstrably cost-effective before it can become pervasive. How can this be accomplished?**

# Principled System Development:

- **Holistic approaches to complexity: sound requirements, structured architectures, principles, good software engineering practice, design for trustworthiness, usability and administrability, pervasive assurance analysis, formal methods, and lots more.**

# Principled System Design

- Management of complexity through constructive architectures that modularly localize what must be trustworthy, such as separation kernels, virtualization, alternative approaches to multiple security levels, and so on.

# Principled System Implementation

- Property-preserving refinements
- Sound software engineering
- Sound programming languages
- Invariant design composability
- Proactive code analysis
- End-to-end self-checking

# Principled System Assurance

- Pervasive assurance throughout development/use cycles.
- Assured composability, with with hierarchical closure as in the Boyer-Moore stack, Robinson-Levitt (PSOS), Rushby-DeLong (new work).
- Assured multilevel security?

# Deja Vu All Over Again, Yogi Berra

- Unfortunately, the same types of mistakes (design flaws, software bugs, operational errors) recur.
- There is much to be learned, from many past mistakes. Educational is crucial.
- Various examples follow.

# Backup and Recovery Risks 1: Air-Traffic Control Failures

- LA Palmdale ATC Jul 2006 power
- Reagan National Apr 2000 power
- LI NY ATC SW upgrade Jun 1998
- LA ElToro ATC 104 failures/day 1989 (no previous system saved)
- 3 NY airports 1991 (on batteries)

# Backup and Recovery Risks 2

More total system/backup failures:

- Swedish central train res system
- Washington Metro Blue Line 1997
- SF BART SW upgrades Apr 2006
- Japanese stock exchange Nov 2005

Cases of losses with no backup:

- NY Public library references
- Dutch criminal mgmt system

# Propagation Risks 1: Widespread Network Outages

- 1980 ARPANET collapse: router memory errors, weak garbage collection of old status messages, memory overflow in every node
- 1990 AT&T longlines collapse: untested change in recovery code, repeated crashing for half a day.

# Propagation Risks 2: Widespread power outages

- Northeast US, Nov 1965
- Lower NY State, Jul 1977, >26 hrs
- Ten Western states, Oct 1984
- Western US, Jul 1996, heat/tree
- Western US/Canada/Baja, Aug 1996
- Northeast, Aug 2003, >2 days

# Propagation Risks 3: Power outages in 2006

- Queens, NY, week-long, wiring
- Portland, Oregon, October
- Ems River, Germany, November. preventive shutdown failed to consider iterative implications (N-1), affecting 10 million in 6 countries from Austria to Spain.

# Software Flaws 1

- **Buffer/stack overflows, missing bounds checks, type mismatches, and other flaws are ubiquitous and keep recurring. This seems rather ridiculous.**

## Software Flaws 2

- Multics prevented stack overflows.
- Progr. languages are a mixed bag.
- Analysis tools: StackGuard (Cowan), buffer overflow analyzer (Wagner), lint family, Coverity (Engler), Fortify (Chess), MOPS (Chen); Microsoft: Spec#/Boogie, PREfast/PREfix, RaceTrack, ...

# Election Systems 1

- Elections should have end-to-end integrity/reliability/accountability, nonsubvertible audit trails, uncompromised voter privacy, etc.
- The entire process is vulnerable: registration, voter authentication, authorization, voting, counting, certifying, recounting, etc.

# Election Systems 2

- Weakness in depth: every step is a potential weak link.
- All-electronic paperless systems are unauditable, lacking integrity, and subject to errors, fraud, and nontechnological problems, as seen in 2000, 2004, 2006.
- HAVA, EAC, evals: simplistic.

# Conclusions 1

- The individual cases may be less important than the fact that we repeatedly see the same types of problems.
- Many common vulnerabilities can be relatively easily avoided with more principled approaches.

# Conclusions 2

- Computer development is mostly an incremental process, driven by marketplace forces. But security research and assurance are slow to be adopted, despite vital needs.
- Incentives are needed to make better use of past lessons.

# Conclusions 3

- Better trustworthiness is urgently, needed, and should be approached holistically, with composable architectures and principled system developments.
- Development and operation of trustworthy critical systems require massive cultural changes.

# A Few Relevant References

- Principled Assuredly Trustworthy Composable Architectures:
www.CSL.sri.com/neumann/chats4.html, .pdf, .ps
- PSOS Revisited, ACSAC 2003:
www.csl.sri.com/neumann/psos03.pdf
- ACM Risks Forum, www.risks.org
- PGN, www.CSL.sri.com/neumann

# Old PSOS/HDM References

- L. Robinson, K.N. Levitt, Proof Techniques for Hierarchically Structured Programs, *CACM,* Apr 1977.
- Neumann, Boyer, Feiertag, Levitt, Robinson, A Provably Secure Operating System: The System, Its Applications, and Proofs, SRI CSL-116, May 1980, large .ps file online.

# Early Alternative-MLS References

- J.M. Rushby and B. Randell
A distributed secure system, *IEEE Computer,* 16(7):55–67, Jul 1983
- N.E. Proctor and P.G. Neumann, Architectural implications of covert channels, *15th National Computer Security Conference*, 13-16 Oct 1992
...csl.sri.com/neumann/ncs92.html