Relating Software Engineering and Information Security

Eugene H. Spafford
8 May 2003

Image removed to respect copyright.
Sorry.
Let’s Start with Some History

- Most of our computing disciplines are “new”
- So is our basic computing infrastructure
  - Experimental protocols
  - Interconnection of smaller networks
  - Commodity software/hardware

Consider.....

Looking Back: 30+ Years Ago

- No significant networks
- Mainframe computing
  - Batch, not interactive
- Computer security was physical security
- Users in the 10s of thousands
- First CS program in US in 1963
Looking Back: ~20 Years Ago

- First Intel-based PCs
  - Apple II, Commodore Pet, others already out
- ARPAnet had 231 nodes
- Usenet created
- First computer virus about to appear
  - Apple II virus in an academic setting
- 100s of thousands of users

Looking Back: ~15 Years Ago

- First Intel/MS computer virus (“Brain”)
- Usenet had $10^5$ nodes
- ARPAnet, NSFnet
- 414 gang hits the newspapers
- Cuckoo’s Egg incident occurring
- Millions of users
Looking Back: ~10 Years Ago

• 100s of computer viruses & worms
• WWW protocol invented
• TCP/IP has $10^6$ nodes
• First security scanner (COPS)
• First general IDs (Tripwire)
• @Large incidents

Looking Back: 5 Years Ago

• Commercial use of the network allowed
• 10,000+ viruses threshold reached
• First Word macro viruses (”concept”)
• First major D-O-S attack
• Initial DNS gold rush
• First root kits
• $10^7$ users
The Internet Today

• Millions of systems on all 7 continents
• Perhaps 500 million users have access
• 220 countries & territories around the world have registered domains
• Online population doubling annually in much of the last decade

Explosion of Storage

• About 200 terabytes of storage in 1995
• 2000 PCs could hold that much in 2001
  – Cost of less than $1 million
  – Worldwide now 10 exabytes (+80% annually)
  – 2002 sales of 8500 petabytes (IDC)
• 50 PCs will hold this much in 2004
But along with the advances....

- Continuing growth in security incidents
  - Viruses & worms
  - Hacking
  - DDOS
  - Fraud
  - Spam
- Attacks are getting more efficient

Example of Evolution

- Melissa, March 1999
  - Word 97, Word 2000
  - $300 million in damages
  - Approximately 4 days, 150,000 systems
- ILOVEYOU, May 2000
  - Outlook
  - As much as $10 billion in damages
  - Approximately 24 hours, > 500,000 systems
- Code Red I, Nimda
  - IIS flaws, with fixes published months earlier
  - 400,000 systems in 14 hours, several billion in damages
- Sapphire/Slammer
  - SQL server flaws, fixed earlier
  - 75,000 systems in 10 minutes, over $1 billion in damage

("Brain" took 5 years to do $50 million)
Software Quality?

- Vulnerabilities reported in common software:
  - 992 in 2000
  - 1506 in 2001
  - 1307 in 2002
- 20% are buffer overflow
  - 75 more so far this year
- Input validation accounts for 47%
  - Includes buffer overflow
- Other design errors represent 26%

Source: Purdue CERIAS Cassandra Service

What the Vendors Say

"There are no bugs in our software! Never, I tell you!"

Source: Microsoft
Security Properties

Traditional
- Confidentiality
- Integrity
- Availability

Other properties
- Auditability
- Accuracy
- Possession
- Non-repudiation
- Pedigree

What is Information Security?

- Policy, Operational, Personnel, Physical security

Network Security

Computer Security

Communications Security
What is Information Assurance?

Disaster Management + Information Security + Software Engineering

Why Assurance?

Security is an unattainable absolute.

We should be seeking high levels of trust, based on sound methods of assurance.

Assurance is an on-going process, not a set of add-on features.
Understanding Assurance

- Assurance requires
  - Limiting what happens
  - Limiting who can make it happen
  - Limiting how it happens
  - Limiting who can change the system
  - Providing recovery mechanisms
- Users don’t tolerate limits well
- But users don’t understand risks
  ...apparently, neither do the developers!

Where to Assure

Policy  Design  Development  Deployment

Requirements & Specifications  V & V  Operation & Maintenance  Disposal
Where to Assure

Policy  Design  Development  Deployment

- Requirements & Specifications
- V & V
- Operation & Maintenance
- Disposal

Current business practice

Security & Software Engineering

- Policy
- Design/Standards
- Implementation
- Testing
- Operation
- Audit
- Patching

- Requirements
- Design
- Implementation
- Testing
- Operation
- Maintenance
Basic Fact

Software quality is necessary (but not sufficient) for trusted systems.

So, what can we learn from information security engineering to guide software engineering?

Observation #1

If we don’t capture security goals in the requirements, then we are not going to get them in the system.

Corollary: security is not an “add-on”
Apocryphal Quote

"If you build software without [requirements and] specifications, it can never be incorrect – it can only be surprising."

Attributed to Brian Kernighan
Using the Wrong Requirements

  - “COTS software is not secure.... It is strongly recommended that COTS products, particularly software, not be used for critical applications.”
- GCN, Sept 11, 2000
  - “The Navy’s next-generation aircraft carrier will use Microsoft Windows 2000 to run its communications systems, aircraft and weapons launchers, and other ship electronics...[Windows] should reduce lifecycle crewing and maintenance costs, as well as procurement costs...”

Observation #2

Few people understand statistics and risk.

It is incumbent on the designers to understand risks ... incorporate risk management into design!
Examples

- Tobacco use
- Seat belt (non)use
- Driving vs. flying

- And one of the few landmarks visible to aliens in UFOs...

Image removed to respect copyright. Sorry.
Observation #3

Our community has been designing solutions for virtuosos and not the general user public

We need to design for the average — or lower — user. Science is different from practice!

Common Users

- Limited depth of skills
- Limited interest
- Unaware of the “big picture”
Corollary

- User interface is important
  - Think of defibrillators
  - Think of cars

Art vs. Trade

- Composers vs. Musicians
- Architects vs. Carpenters
- Sculptors vs. Masons

Not everyone should be entrusted with sharp tools!
Observation #4

We do not understand the psychology of programmers and developers

“Before software can be reusable it first has to be usable.” Ralph Johnson
Personal Hubris

- 70% of drivers believe they are in the top 1/2 of drivers. The same seems to be true of coders.
- Many programmers believe that faults are “accidents” and the exception.
- Speed and cleverness are seen to trump safety.

Sad to say...

- Our information infrastructure would be more secure today if written in Cobol!
- Where would we be today if Unix had been written in Eiffel? APL? PL/1?
Carrots vs. Sticks

Image removed to respect copyright.
Sorry.

Observation #5

We are unnecessarily bound by legacy systems and decisions

“Those who cannot remember the past are condemned to repeat it.”
— George Santayana
Consider...

- Lint was written separate from the C compiler to fit in 32K memory. The C standard still embraces the legacy.
- Testing is often based on uniprocessor architectures
- Most of our research is being driven based on legacy environments designed 10 or 20 years ago!
- Where is the real innovation?
Do We Want to Perpetuate This Approach?

Microsoft Publishes Windows 2000 Bug List!!

Microsoft today published the full bug list for its Windows 2000 operating system. For the first time ever, all known and reported bugs are to be made available to the public. Mr. Hyan Lee of Japan (photographed above) made the mistake of printing the whole list.

Observation #6

Our teaching too often stresses the wrong results.

Do we teach in teams, over a long time, and stress quality over meeting limited requirements?
Who is Writing the Security-Critical Software?

Observation #7

Emergent behavior from unexpected uses is IMPORTANT and should be designed for
Recapitulation

- Software Engineering has been concerned with quality during normal use.
- Information Security has been concerned with preventing abnormal use.

Let’s focus on reliability and quality during any use!

Thus...

Security needs may well be the best thing that has happened to software engineering in years.

But it means retaining an understanding of the present — and a willingness to break with the past.
Thank you!