Security and Privacy Meta-Requirements

Eugene H. Spafford
30 August 2001

Basic Infrastructure

- Experimental protocols
- Interconnection of smaller networks
- Commodity software/hardware

- Seven years ago there was no commercial use of the net.
  - 10 years ago, fewer than 100,000 systems
  - 231 systems in 1981
- The workstation is about 17 years old.
- The Intel-based PC is 20 years old.
The Internet Today

- Millions of systems on all 7 continents
- In excess of 400 million users have access
- 220 countries around the world have registered for access
- Population doubling in approximately 10 months for last 11 years
- Volume of traffic doubling approximately every 90 days

Explosion of Storage

- About 200 terrabytes of storage in 1995
- 2000 PCs could hold that much in 2001
  - Cost of less than $1 million
  - Now about 10 exabytes of storage (80% growth per year)
- 50 PCs will hold this much in 2004
- Growth continues
Future Environment: The “Evernet”

- World-wide
- High speed networking
- Cheap (free?), ubiquitous computing
- Widely-deployed encryption
- Truly mobile computing
- Many embedded systems connected
- Billions of users

State of Security: Poor

- Examples abound:
  - DoD reports 22,000 attacks on Pentagon systems in 2000 (over 250,000 through all DoD)
  - 3 Incidents at Microsoft, Oct 2000, Jan 2001
  - Feb 2000, Denial of Service against eBay, Yahoo, Amazon
  - China/US “Cyber-skirmish”
  - Code Red worms, SirCam virus in fall 2001
- CSI/FBI figures
  - Fewer than 20% sites report no unauthorized use
  - Average loss of $1 million per year
Real losses

- 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
  - IIS flaws, with fixes published months earlier
  - 360,000 systems in 14 hours, several billion in damages

(“Brain” took 5 years to do $50 million)
More data

- CERT/CC fielded 21,756 incidents in 2000
  - Growth from 3734 in 1998, 9859 in 1999
- On-going probes (via Intel)
  - 50-60 incidents per day on Internet
  - 10-12 incidents per day on DSL
  - 5-6 incidents per day on dial-up
Typical user

- Less than 1 year online
- No background in computing
- Has major OS, 1 Ghz machine, but uses only 3 applications
- Doesn’t make backups
- Online constantly

In other words, a target

The World in 2004
(at this rate)

- 100,000 computer viruses
  - 99% for one vendor’s software
  - New viruses @ more than 1 per hour
- Most common desktop system
  - Almost 100 million LOC, 4Ghz+
  - 1 security patch announced per day
- Attacks over network exceed 10 per hour
- Losses to business and government will exceed $100 billion per year
Security & Privacy?

- Confidentiality
- Integrity
- Availability
- Auditability
- Control
- Accuracy

- “The right to be let alone”
- Control over what information about you is revealed, and to whom

Critical Concepts

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

Where to Assure

- Policy
- Design
- Development
- Deployment

- Requirements & Specifications
- V & V
- Operation & Maintenance
The Ideal

- Understand the needs of the users
  - Narrow focus better than broad
- Understand basic tenets of security
  - Paucity of programs and experts
- Capture requirements for design and validation
- Design with care using good tools and methods
- Validate & Verify
- Operate with good human-computer interface

The Reality

As metaphor, by Monty Python
(suggested by Steve Chapin)
Do you agree?

“...From a practical standpoint the security problem will remain as long as manufacturers remain committed to current system architectures, produced without a firm requirement for security. As long as there is support for ad hoc fixes and security packages for these inadequate designs and as long as the illusory results of penetration teams are accepted as demonstrations of a computer system security, proper security will not be a reality.”

Preliminary Notes on the Design of Secure Military Computer Systems, Roger Schell, USAF, 1/1/73
Magnitude of the Problem

- There is no perfect code.
- Assume a conservative rate for serious faults
  - 1 error per 1K LoC in unaudited code (20 pages)
  - 1 error per 5K LoC in examined code (100 pages)
- Kernels
  - OpenBSD 2.6
    ~1874K lines, implying ~375 faults
  - HP/UX
    ~2341K lines, implying ~470 faults
  - Linux 2.2.121
    ~1500K lines, implying ~1500 faults
  - Windows 2000
    >30 million lines, implying > 6,000 faults

About 30% are buffer overflows or unchecked data
Over 90% are coding/design flaws.

Source: Securityfocus.com
Understanding the User

- Users span a range from expert to fool
  - Most fools think they are experts
- Requirements and policies vary widely
  - Policy for a university different from a military agency
- However, to maximize market, vendors build for the most general, simple case

Some Basic Problems, Restated

- Internet Time and current market
  - Demand
  - Personnel
- User expectations
- Lack of consequence and liability
  - Embedded “Easter eggs”
  - ILOVEYOU 14 months after Melissa
- Changing user base
- Poor software engineering and QA techniques
Let’s look at requirements specifically...

Apocryphal Quote

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

Brian Kernighan
Missing Requirements

- Quality of Service
- Privacy
- Integrity support
- Auditability
- Testability
...many more

Complicated by not having good metrics.
The result is choices based on cost or speed instead of safety, privacy and security.

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...”
Worth Repeating

- Least privilege
- Economy of mechanism
- Complete mediation
- Open design
- Separation of privilege
- Least common mechanism
- Psychological acceptability

J. H. Saltzer & M. D. Schroeder 1975

Structure with least privilege

- Concept is to limit:
  - Protection domains
  - Access
- No superuser
- Fine-grained ACLs, real capabilities or similar
- Role-based authentication
- Confinement
- ...These don’t match a desktop-based model!
Build with economy of mechanism

- Privileged code should be
  - Small
  - Simple
  - Easy to verify
  - Security built in instead of added on
- Use proven methods

...Conflicts with feature-rich design

Ensure complete mediation

- Every access checked
- Security kernel approach or real capabilities

...Conflicts with speed and performance, and doesn’t map well to WWW.
Build to “open design”

- Should not depend on the design being kept secret
  - Note: *Not* the same as “no security through obscurity”
- Primarily applies to cryptographic modules
- (Open design is *not* the same as open source!)

…”NIH” syndrome complicates

Note about open source

- S/COMP
- Trusted VMS
- CMWS Ultrix and SunOS

...all are closed source systems

The key is quality, and that depends on training, methodology, and control. NOT OS vs. CS
Use separation of privilege

- Access should require more than one authorization
- Superuser is a bad concept as is single user/all rights
  - Macro viruses, ILOVEYOU worm, etc
- Model of administrator is also the security officer violates this principle

... Doesn’t map to the desktop computer

Use least common mechanism

- To reduce information flow
- To reduce race conditions
- Reuse of verified code is good, up to a point

...Plays havoc with backwards compatibility
Seek psychological acceptability

- Easy to use
  - Should be as easy to use as not to use
- False alarms should be avoided
- Frequent changes and updates are bad
- Should not require great expertise to get correct or use

...Doesn’t match user population

A Comment on Patches

- Fixes for flaws that require an expert to install are not a good fix.
- Fixes that break something else are not a good fix.
- Frequent fixes may be ignored.
- Goal should be design, not patch
Additional Requirements

- Testability
- Service
- Auditability
- Identity
- Data pedigree
- Data aging

Build testable software

- Require test points in critical code
  - Should be useful for revisions
- Self-checking code
Ensure minimum service

- Critical functions should get assured service
- Load shedding should be designed in

Construct audit capabilities

- All critical components should contribute meaningful audit
- Identify what, how, who, when
- Audit should be timely
- Audit should be protected
Protect identity privacy

- Establish authorization without identification
- Establish authentication without identification
- Use random identifiers
  - Not SS#
- Use user-selectable authenticators
  - Not mother’s maiden name!

Keep data pedigree

- Where did data come from?
- Who entered the data?
- Who changed the data?
- Who validated the data?
- Expire the data
What can we do?

- Understand that there is no “average user”
- Understand balance between features and security
- Use known good methods of software engineering
- Employ better testing
- Understand policy differences.
- **Build in assurance from the start**
  - Establish sound requirements for security & privacy

Users need to be better consumers

- 28-30 million lines of code for an OS !?
- Consumers need to start demanding quality and security instead of new features.
- Security & QA needs to be explicit part of every design and measured for the consumer
- Hacking into systems is not security (“penetrate and patch” is not design)
The Assurance Problem

Technology

Personnel

Time

Regulation

Consumer demand

Collateral costs

Liability

Need to Motivate Vendors

- Security & assurance can be a sales differentiator
- Stop depending on add-ons
- Lawyers are getting involved
- Vendors need to apply well-known QA methods
- Hiding behind laws like UCITA should antagonize the public

http://www.acm.org/usacm/IP
Closing thought

“There is more to life than increasing its speed.”
Ghandi

Thank you!
Questions?