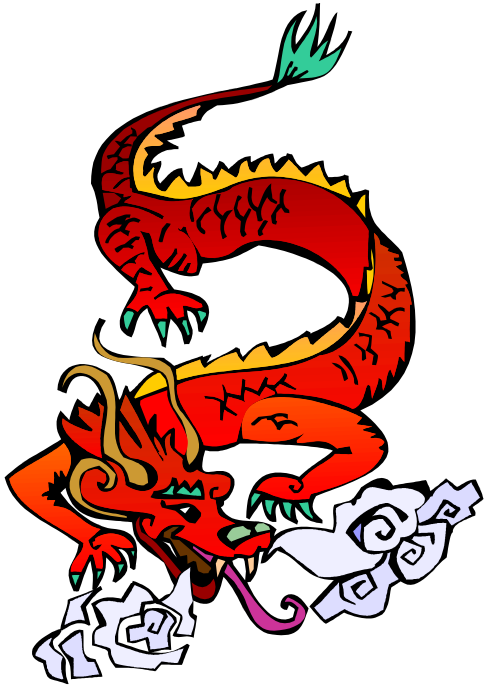
Next Slide

Security History & Architecture

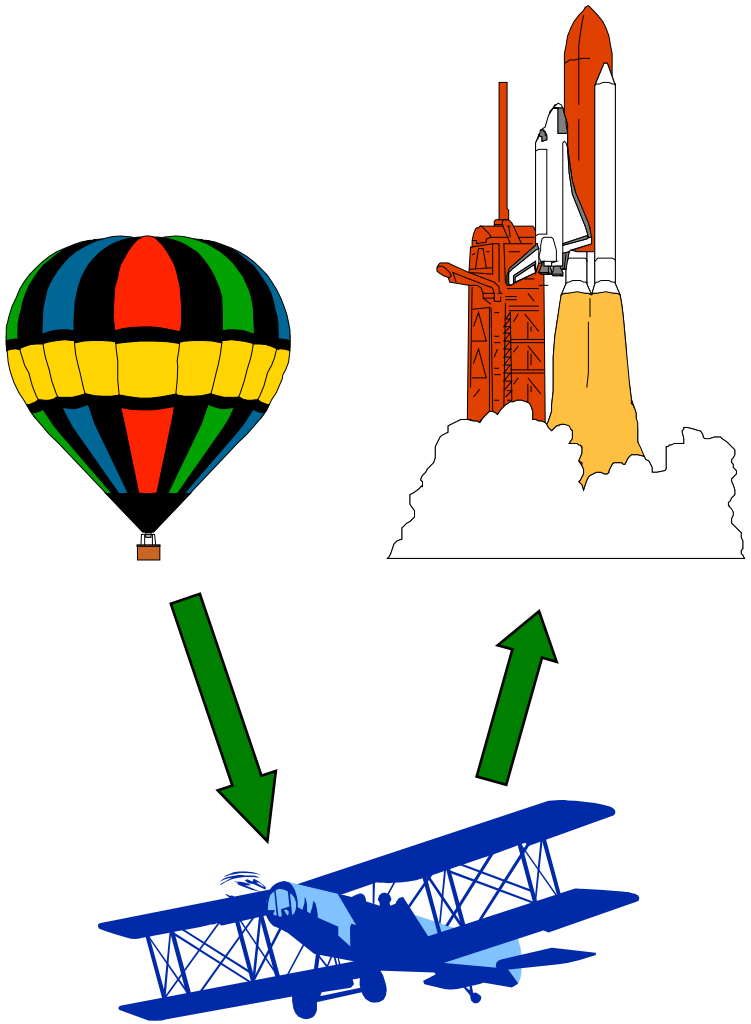
(and other *VERY* scary stories!)



Dr. Steve G. Belovich

IQware™

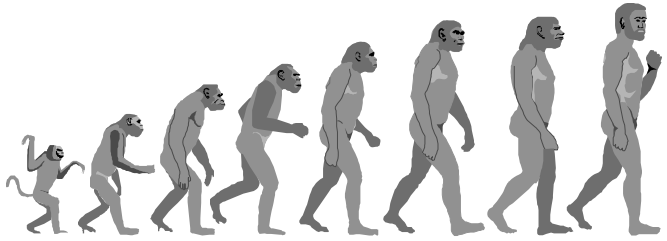
Why IT & Security History?



- It explains why we are in this condition.
- It significantly affects our options today.
- It can help prevent repeat errors.
- You can't get a solution if you don't understand the problem.
- Ignorance is very expensive!

The Evolution of Computing

Primates - 65 to 5 million BC



You Are Here

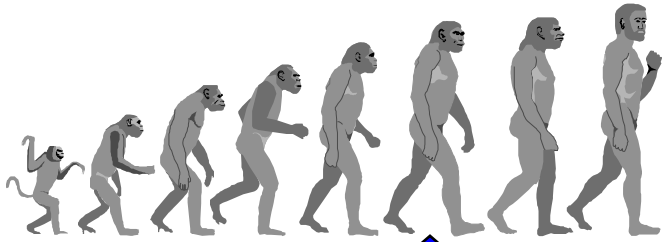
- 1830 - Charles Babbage Analytical Engine”
- 1937 - Howard Aiken Proposed “Automatic Sequence Controlled Calculator”
- August 1944 - “Mark I” completed
- 1946 - Von Neuman Architecture implemented in ENIAC/EDVAC
- 1951 UNIVAC 1
- 1952 IBM 701 - IBM’s first commercial box (19 units sold)

Key facts:

- Getting hardware working was the only issue.
- O/S concepts not there yet.
- Need for security non-existent.

The Evolution of Computing

Homo-Erectus - 2 million BC



You Are Here

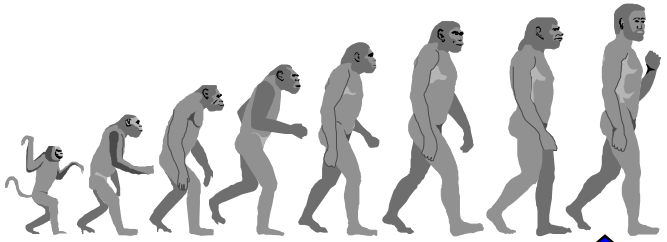
- 1960 - IBM 360 architecture used micro-Programming
- 1960 - Burroughs B5000
- 1964 - IBM 370
- 1973 - Arpanet
- 1975 - Cray 1 (64-bit)

Key facts:

- Hardware more reliable.
- Operating systems evolving.
- Private Networks invented.
- Multi-task O/Ss require task isolation and protection.
- Some basic security concepts formed.
- Small installed based allows lots of experimentation.

The Evolution of Computing

Homo Sapiens - 200,000 BC



You Are Here

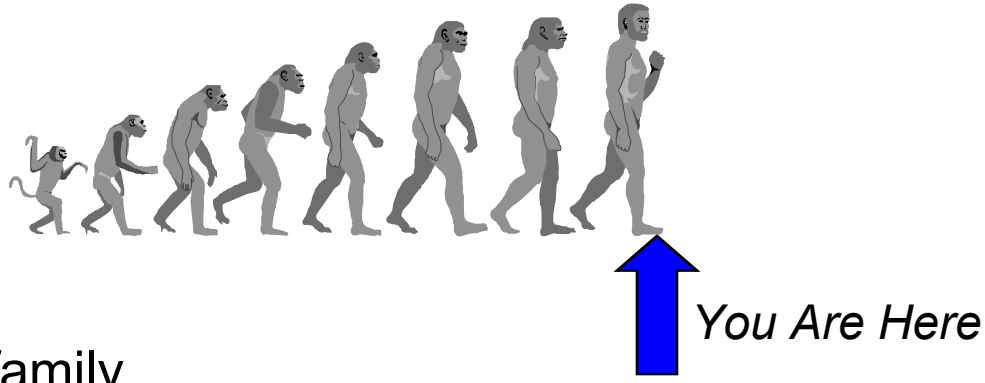
- 1979 - 4361/4381 (44-bit)
- 1981 - IBM 3081
- 1985 - Cray 2
- 1986 - IBM 3090
- 1990 - Internet
- 1996 - IBM Z-series

Key facts:

- Operating systems stable.
- Growing installed base - fewer new architectures.
- Multi-user O/S requires user isolation & protection.
- Public networks invented.
- Remote access requires isolation & protection.

The Evolution of Computing

Agriculture - 10,000BC



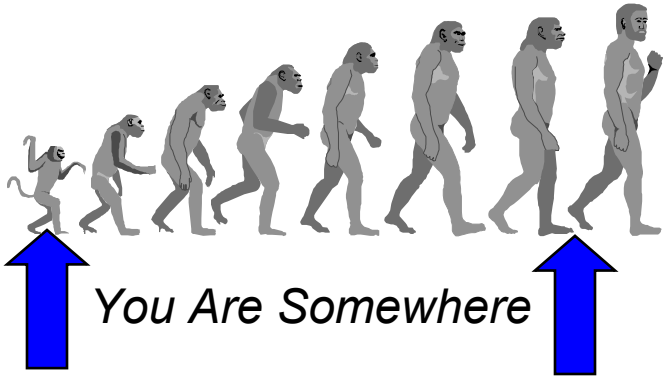
- 1965 - PDP-8/12 family
- 1970 - PDP-11 family (16-bit)
- 1977 - System 34
- 1977 - VAX family (32-bit)
- 1978 - System 38
- 1983 - System 36 - largest installed base of any computer
- 1986 - RS6000
- 1991 - Alpha family (64-bit)
- 1998 - IBM “P series” (64-bit)

Key facts:

- Hardware shrinks.
- Midrange machines introduced.
- Large installed base (200,000-400,000 machines).
- O/S has more features.
- Languages evolving.
- Significant remote access.
- Security matters.

The Evolution of Computing

Special-Purpose Genetic Manipulations



- 1970 - 4004 & 8008 micros
- 1974 - 8085
- 1977 - AMD2900
- 1980 - 8086 family
- 1980 - 68000 family
- 1980s - zillion special-purpose processors
- 1995 - Pentium family
- 2003 - Less than you think!

Key facts:

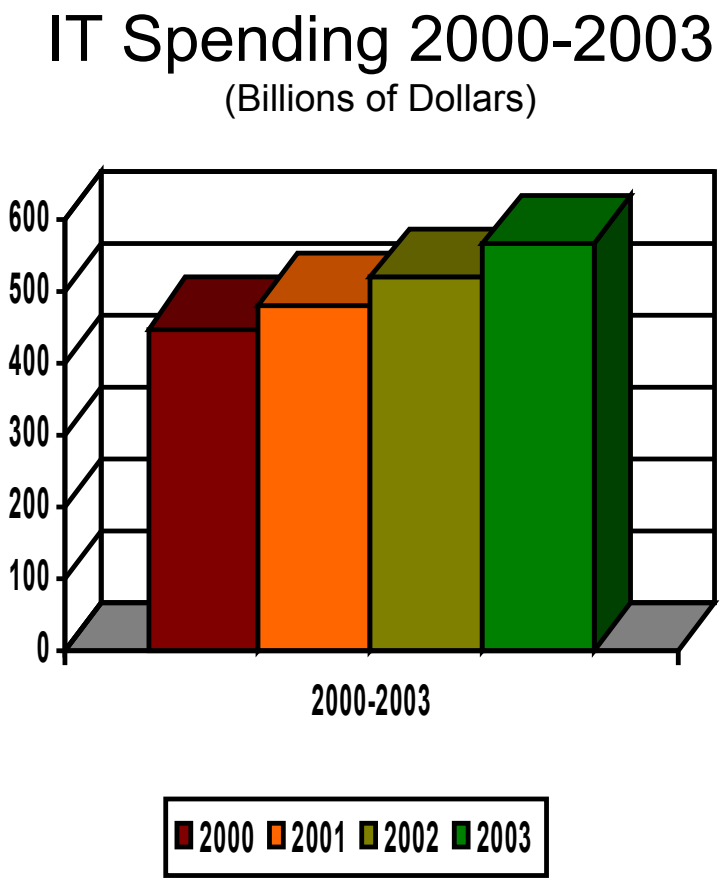
- Hardware really shrinks.
- O/S start out simple and dumb.
- Security deliberately eliminated.
- O/S grows to encompass what used to be applications.
- Cost is the driver; security & performance are secondary.
- Multimedia matters.
- Huge installed base limits change.

Security Research History

(What Secure Systems Must Do)

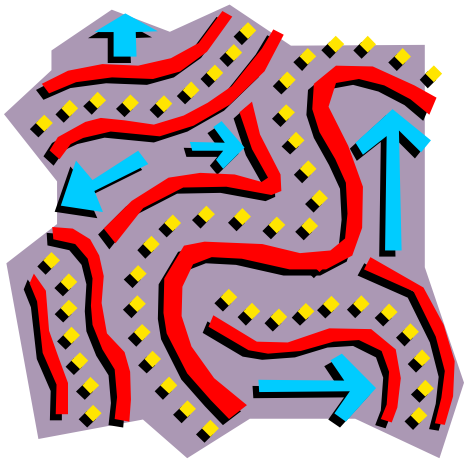
- Control access to information.
- Permit only authorized entities to read, write, create or delete information.
- Must have secure architecture.
- Must obey secure system model.
- Must have proper architecture, coding and deployment of the O/S, the application and all layers in between.
- Discovering security flaws then fixing them one-by-one does NOT work.

Economics 101



- Consumers control desktop market.
- No consequences for lack of adequate security.
- Demand for programming exceeds competent supply.
- Top 10 vendors have 55% of market share & 67% of workforce in packaged software.
- Until 9/11/01, security had had a high cost and low perceived market value.
- Buying bad stuff tells them to make more bad stuff!

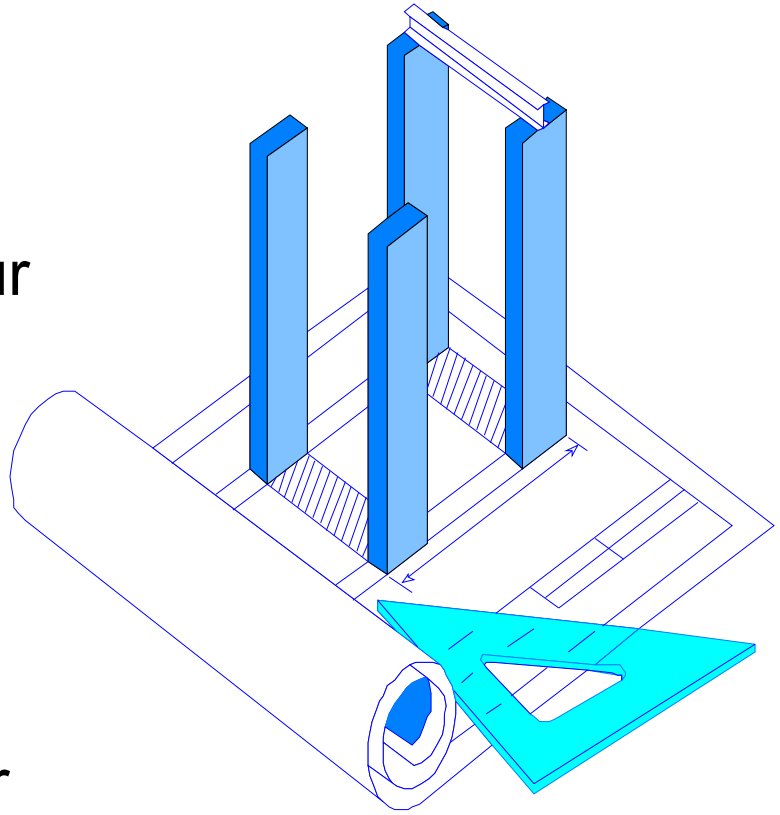
Part 1 Summary



- Long & winding road for computer and IT system evolution.
- Low-end machines are consumer products where cost is king.
- Security was deliberately not included in desktop machines.
- Huge installed base prevents significant desktop re-design.
- Secure systems require the correct architecture - more in Part 2!

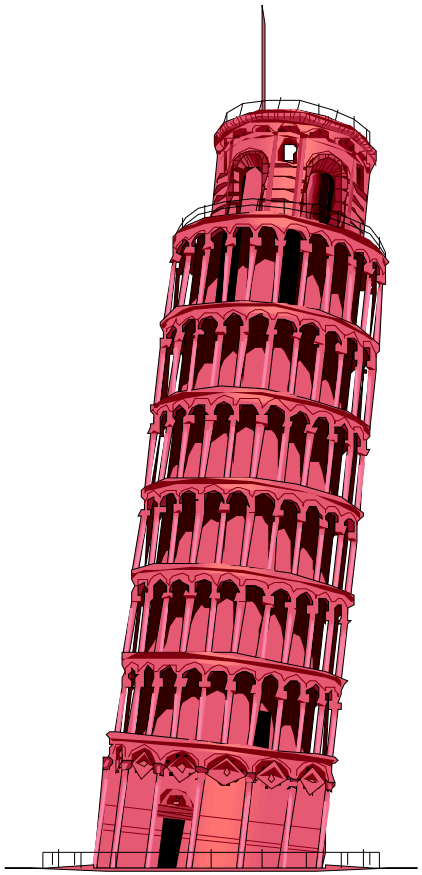
Security Architecture

- IT systems are built over time.
- Software is built in layers.
- Prior IT decisions can limit your choices today.
- Constant change is the rule.
- Must understand layers to direct security efforts properly.
- Must understand which IT operations matter most to your company.



Software Structure

- **Business Applications**
- **Databases**
- **Network communication software, Internet**
- **Languages, Compilers & Tools**
- **Utilities & Libraries**
- **Operating system**
- **ISA (Hardware Layer)**



- **Libraries**
- **Books**
- **Chapters**
- **Paragraphs**
- **Sentences**
- **Words**
- **Alphabet**

1) Layers made by different & competing vendors
2) Minimal universal standards between layers
3) Inter-layer interfaces changing often and without warning
4) Interface features come and go from release-to-release

The OSI 7-Layer Model

Of network-oriented software

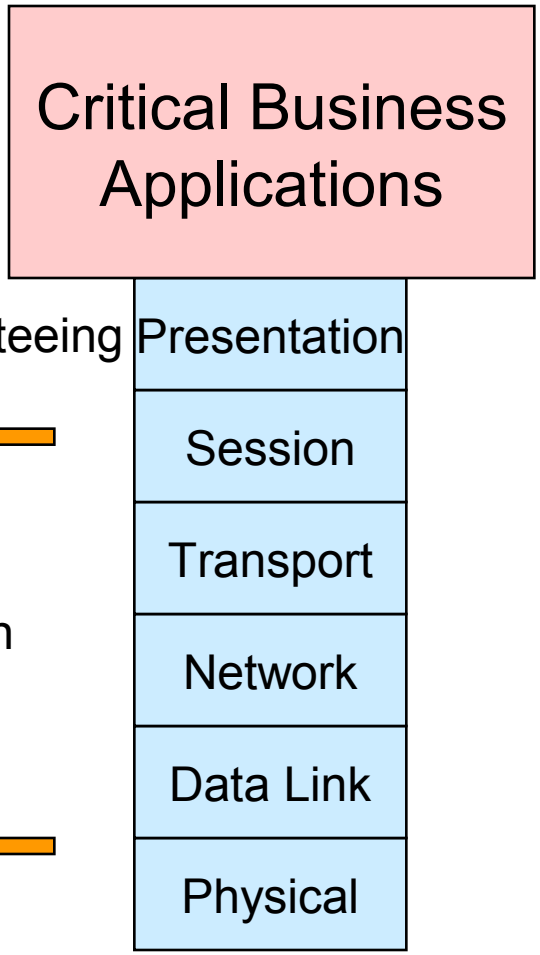
Protection Methods

IQware™

No mechanisms for guaranteeing correct program operation.

Encryption, firewalls, anti-virus software, intrusion detection systems, etc.

Cardkeys, Biometrics, etc.



- The application does the useful work - it is the “payload” of your IT system.
- The 6 underlying logical layers are there to support and enable the application.
- All seven layers must be protected to have 100% security.
- Protecting just the lower layers via firewalls and anti-virus software is not enough - as experience has proven.

Secure System Requirements

- **Policy**
  - **Security Policy** - System must enforce a well-defined security policy.
  - **Marking** - System must associate all objects with access control labels (sensitivity & access modes).
- **Accountability**
  - **Identification** - System must identify individuals and their various authorizations in a secure manner.
  - **Audit Trail** - System must keep & protect audit trail so actions may be traced to responsible party.
- **Assurance**
  - **Evaluation** - System must have hardware/software mechanisms that can be independently evaluated to assure that policy & accountability are enforced.
  - **Continuous Protection** - System must continuously protect trusted mechanisms that enforce policy & accountability from tampering.

Secure System Classification
(DOD)

- **D - Minimal Protection**
- **C - Discretionary Protection**
  - **C1 - Discretionary security protection** - separates users & data, uses credible controls to enforce access limitations on an individual basis.
  - **C2 - Controlled Access Protection** - users individually accountable for their actions, security audit trail, resource isolation.
- **B - Mandatory Protection**
  - **B1 - Labeled Security Protection** - security policy model, keeps integrity of sensitivity labels, sensitivity labels must be held in all major system data structures, demonstration of reference monitor implementation.
  - **B2 - Structured Protection** - formal security policy model, discretionary & mandatory access control enforcement extended to all subjects & objects, separation of critical & non-critical system elements, stringent configuration management controls, covert channels are addressed, relatively resistant to penetration.
  - **B3 - Security Domains** - Reference monitor mediates all accesses of subjects to objects, be 100% tamperproof, TCB (trusted Computing Base) only contains security-relevant code & data structures, system engineered for minimal complexity, security-relevant events are signaled, system recovery is required, highly resistant to penetration.
- **A - Verified Protection**
  - **A1 - Verified Design** - Functionally same as B3, full mathematical verification of design.

Secure System Classification

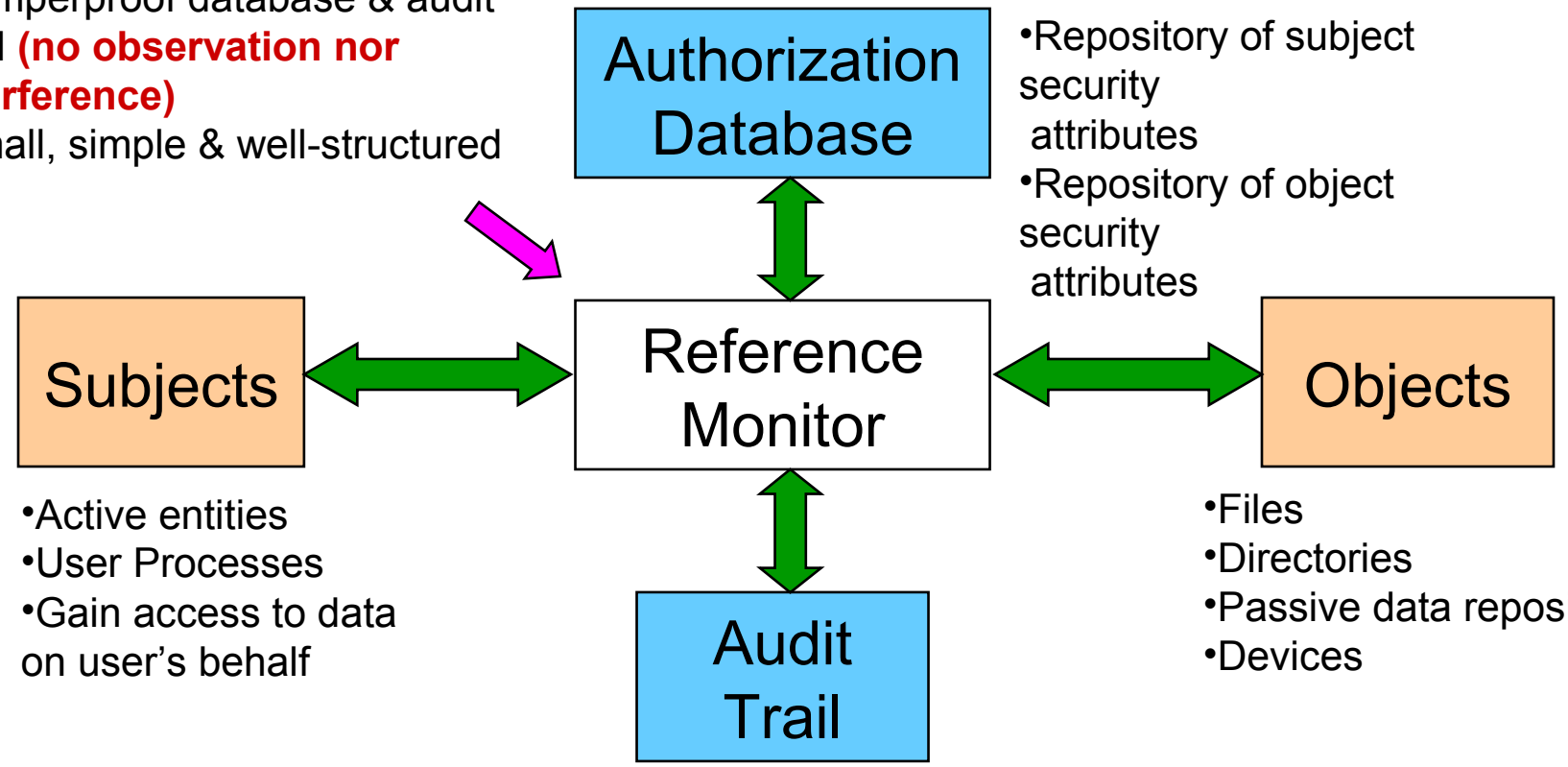
(NIST - National Institute of Standards & Technology)
(NIAP - National Information Assurance Partnership)
ISO 15408

- **EAL-1 - Functionally Tested** - independent testing of selected features.
- **EAL-2 - Structurally Tested** - independent testing of selected features using limited developer design data.
- **EAL-3 - Methodically Tested & Checked** - independent testing using limited developer design data, selective developer result confirmation, evidence of develop search for obvious vulnerabilities.
- **EAL-4 - Methodically Designed, Tested & Reviewed** - independent testing using low-level vendor design data, search for vulnerabilities, development controls, automated configuration management.
- **EAL-5 - Semiformally Designed & Tested** - independent testing of all of the implementation (TOE), formal model, semiformal conformance to design specs, vulnerability assessment for attackers with moderate potential.
- **EAL-6 - Semiformally Verified Design & Tested** - independent testing of 100% of TOE, modular & layered approach to design, structured presentation, vulnerability assessment for attackers with high potential, systematic search for covert channels.
- **EAL-7 - Formally Verified Design & Tested** - same as above, but all models, specs & presentations are formal, TOE is tightly focused on security functionality, amenable to formal analysis, design complexity must be minimized.

The Reference Monitor

(A Secure System Architecture)

- Enforces security policy
- Mediates all attempts by subjects to access objects
- Tamperproof database & audit trail **(no observation nor interference)**
- Small, simple & well-structured



- Repository of subject security attributes
- Repository of object security attributes

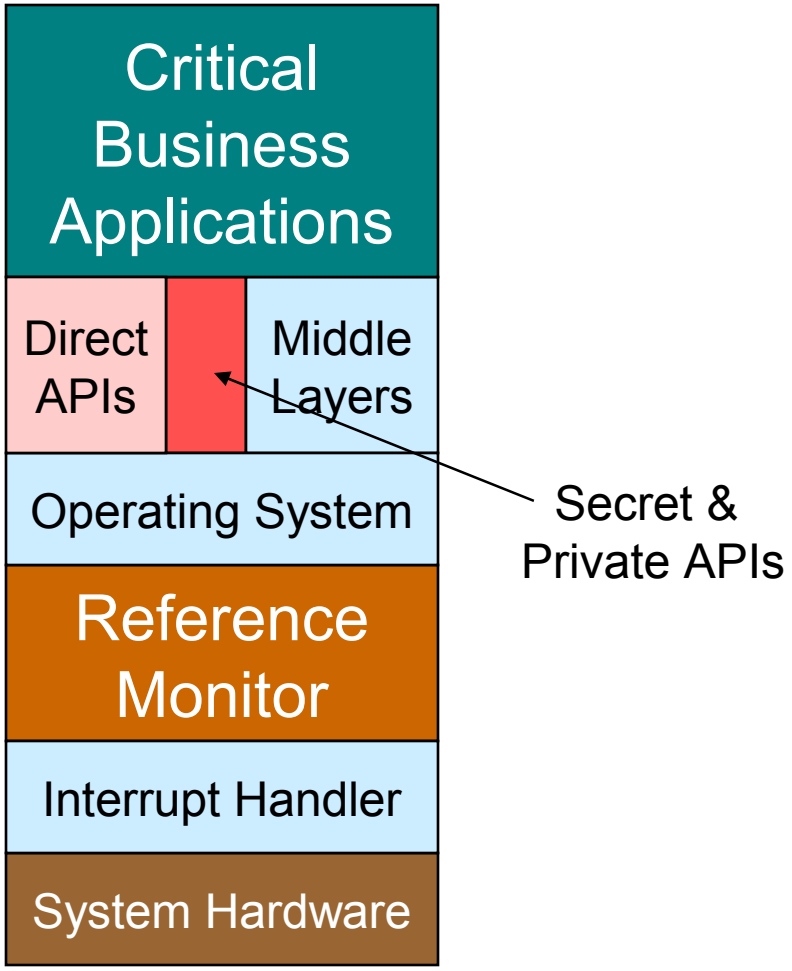
- Active entities
- User Processes
- Gain access to data on user's behalf

- Files
- Directories
- Passive data repositories
- Devices

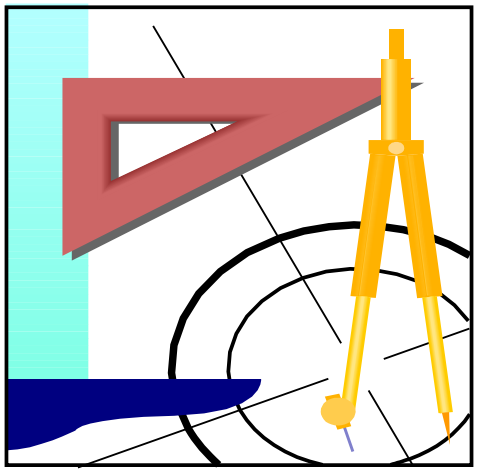
- Record of all security-related events

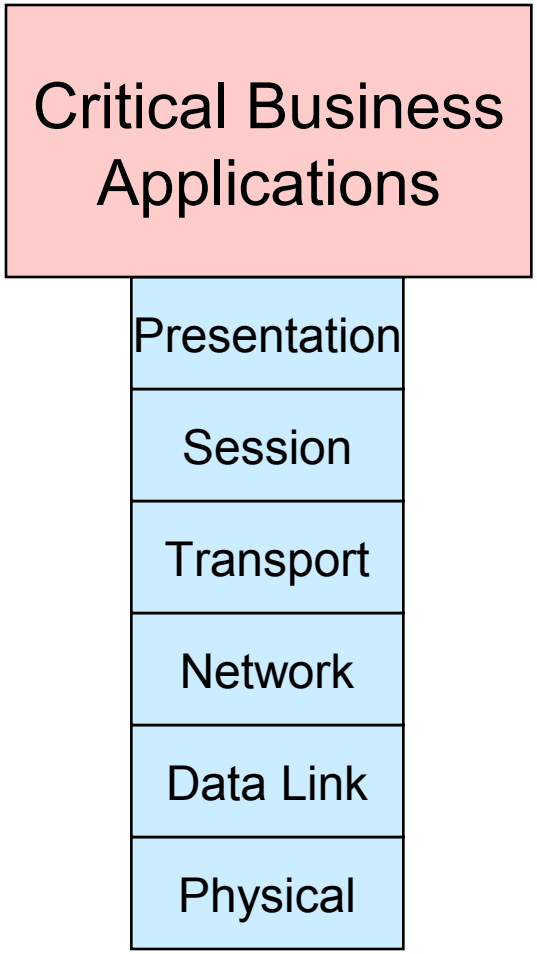
Where the RM is Used

- Reference Monitor (RM) must be part of O/S at lowest layer.
- RM must always control subject-object access.
- O/S cannot bypass RM at any time.
- RM cannot be “turned off”.
- All APIs must go through RM.
- Cannot “bolt on” the RM ex post facto.



Part 2 Summary



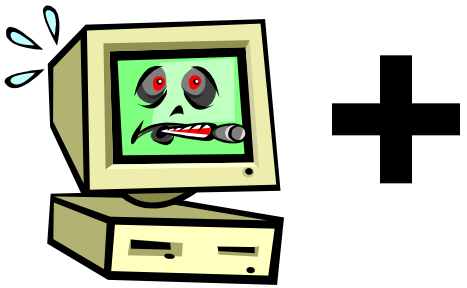


- Software & IT systems are architected in layers.
- Software model shows why firewalls, Anti-Virus and Intrusion Detection systems will never be 100% effective.
- Reference Monitor - or equivalent - is required for a secure O/S.
- Applications must also adhere to proper security design requirements.
- Must use RM properly for success.
- Secure systems require the correct implementation - more in Part 3!

Security Implementation

Concept Summary

- There is no PC prophylactic!
- Security requires proper architecture and cannot be just “added-on”.
- Huge installed based gives little incentive for radical change.
- Simply going over each line of code - a famous Microsoft quote - won’t fix the security problem.
- Non-secure platforms are ubiquitous.
- Have to

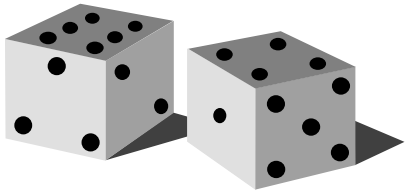


Insecure System

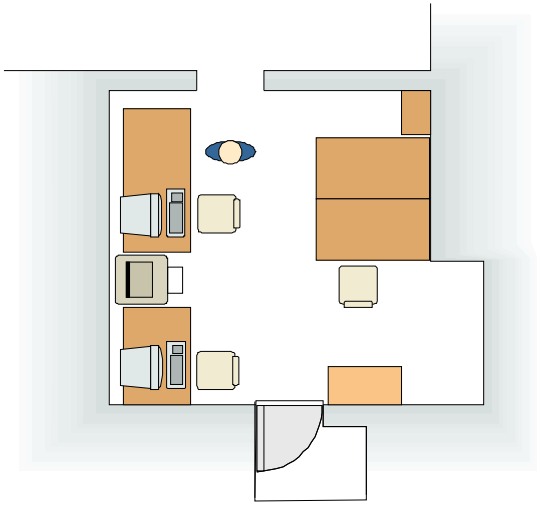
“Magic Software”

Secure System

Operational Risk Analysis

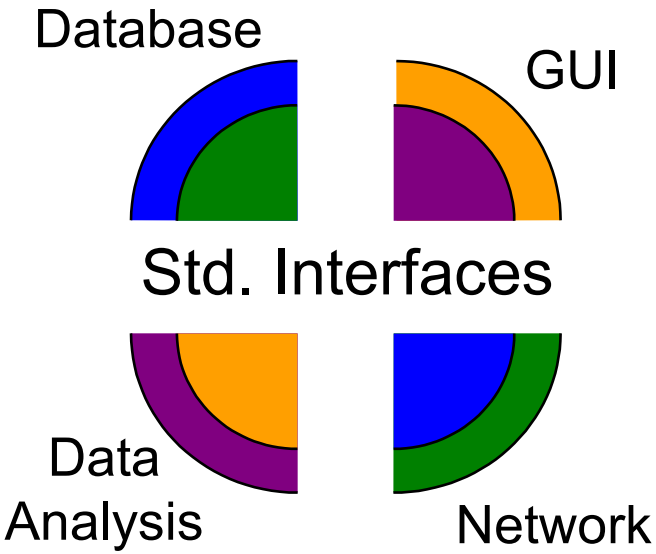


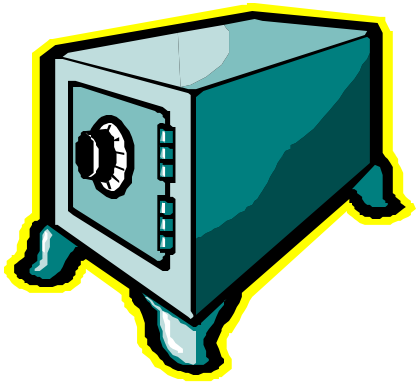
- Analyze existing IT operations.
- Identify critical ops that must run 24x7.
- Identify ops that organization can live without for a short time.
- Identify input & output data set locations for these ops.
- Identify which apps do these ops now.
- Identify how these apps are architected and deployed.



Deployment Guidelines

- Deploy critical apps only on secure platforms.
- Ensure that critical functionality is performed only by critical apps running on secure platforms.
- Segregate the data presentation, data analysis, network communication and database portions of your applications.
- Setup all applicable security controls for these apps and O/S.
- Force password management.

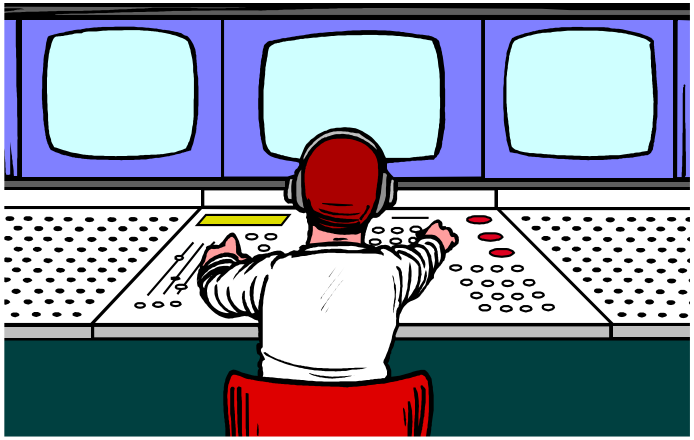


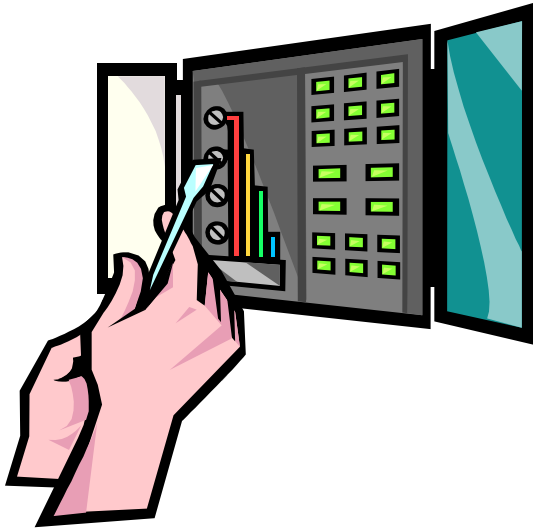


Deployment Example

Security Monitoring at NASA

- Critical operations:
  - Reading cameras
  - Reading other sensors
  - Analyzing data & drawing conclusions
  - Main operator interface
- Less critical operations:
  - General user interface
- Critical aps:
  - I/O drivers for sensors
  - Data analysis
  - Main operator interface

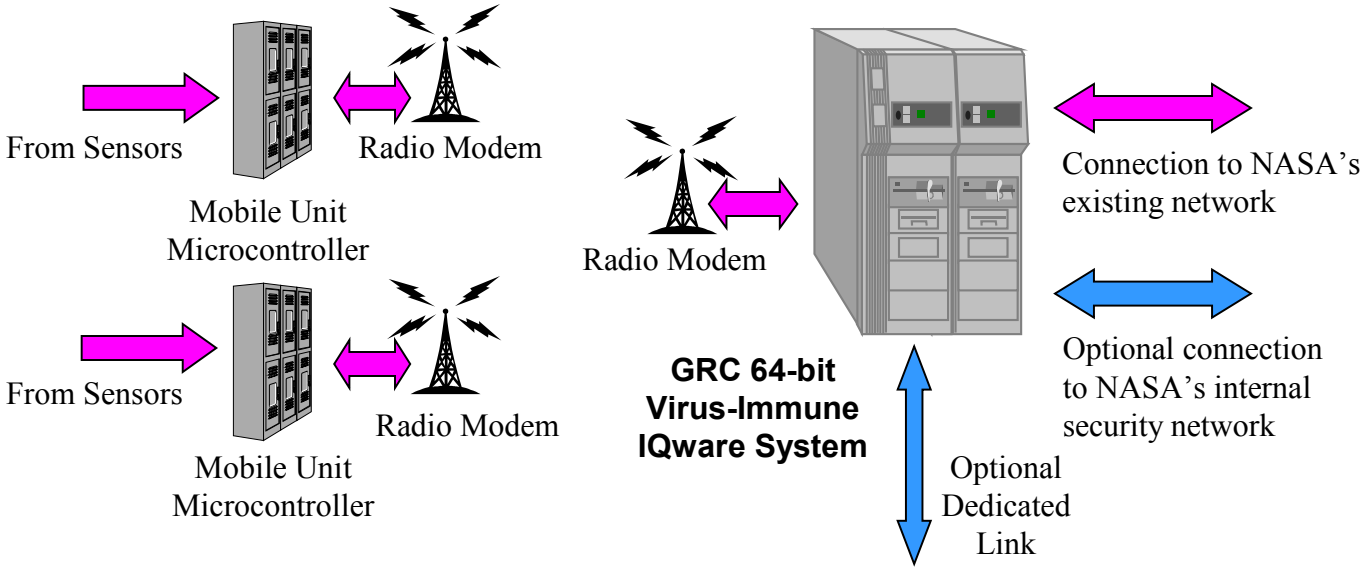




Deployment Architecture

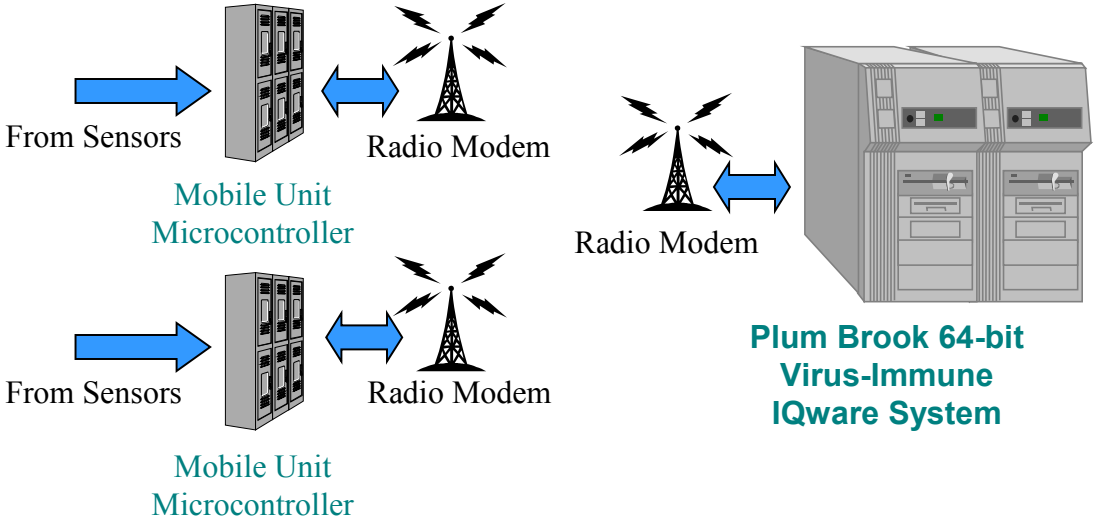
GRC Mobile Unit Sensor List

Flow Rate
Temperature
Conductivity
C(organics)
pH
Various security sensors



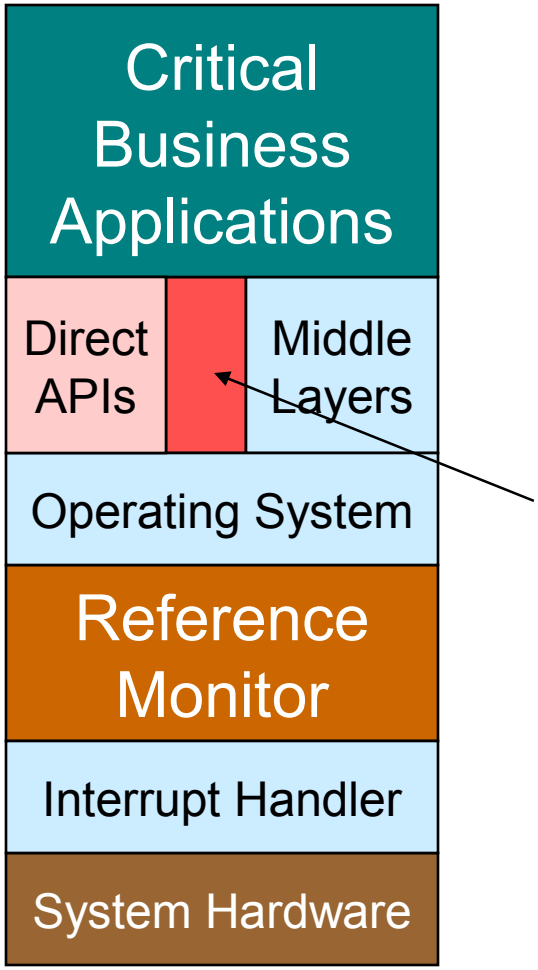
Plum Brook Mobile Unit Sensor List

Flow Rate
Temperature
Conductivity
C(chlorine)
pH
Various security sensors



Notes:
Radios use proprietary non-standard protocol.

Part 3 Summary



- Software has a vertical structure so ensure that your apps are on a solid base.
- Use the right tool for the job:
  - Use secure system for critical information.
  - Use secure system for critical applications.
- Deploy critical aps on secure platforms
- Use secure coding techniques
- Deploy apps carefully, using all available security controls.