

# The OSI-7 Model and Virus-Immune Software Design

Dr. Steve G. Belovich

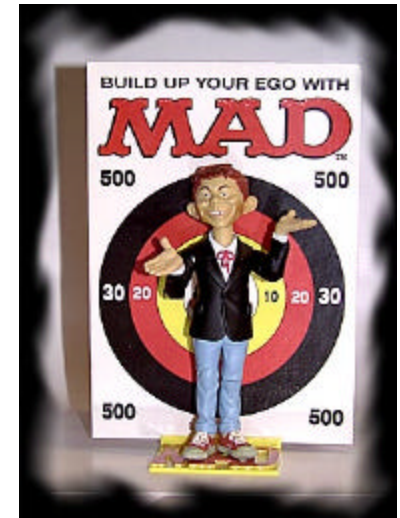


# Cyber Attacks & Viruses

What, Me Worry?

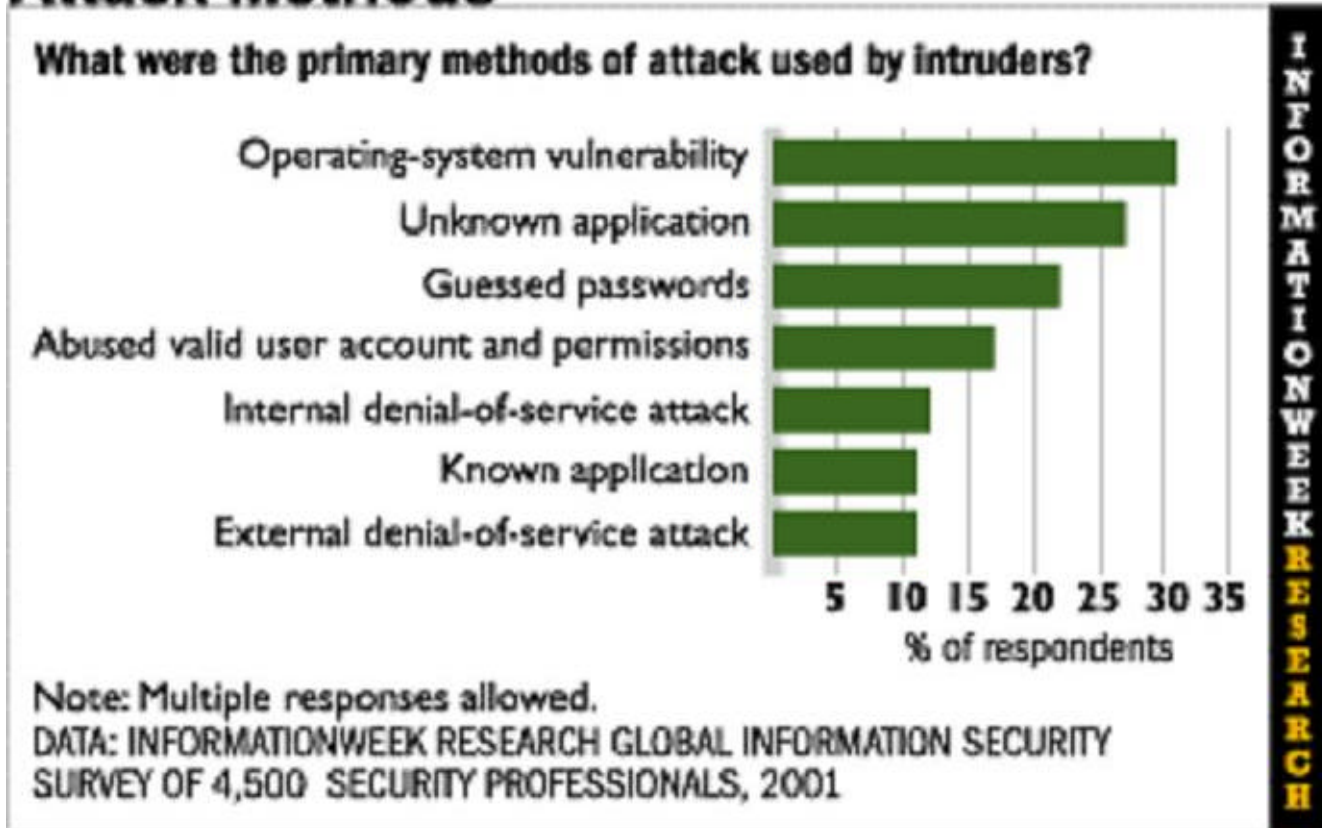
## We Care Because:

- They cause damage.
- Attacks are increasing in frequency & severity.
- Cleanup is expensive.
- Cost of business disruption is a lot higher.
- Cost of data theft is higher still.
- Cost of a “time bomb” is incalculable.



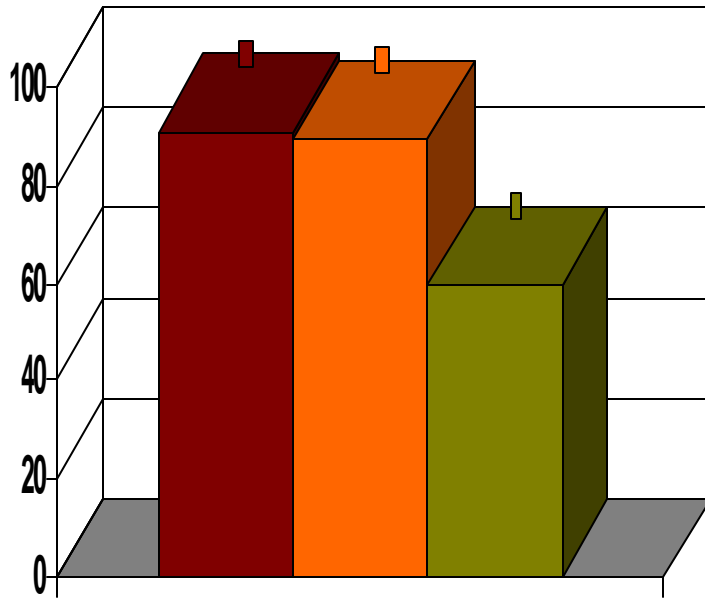
# Cyber Attack Facts

## Attack Methods



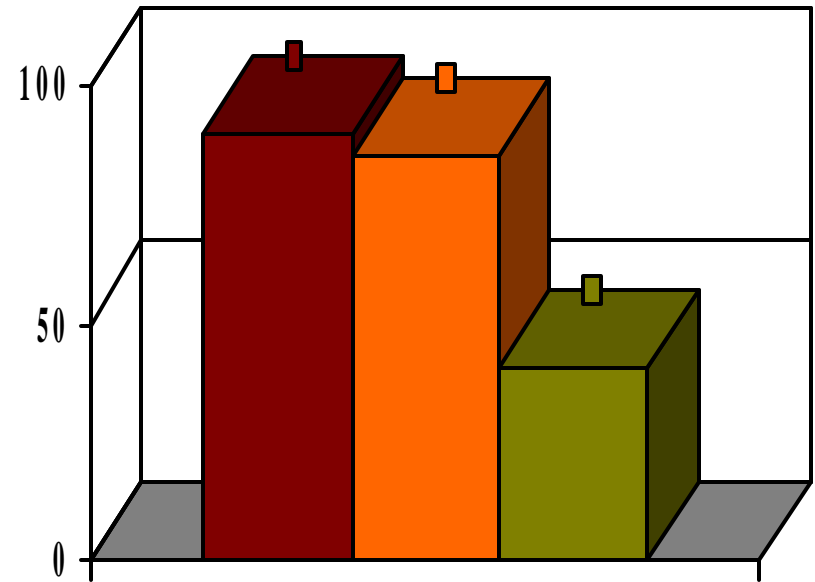
# Cyber Attack Survey

## Defense Methods



■ Anti-Virus ■ Firewall ■ Intrusion Det.

## Defense Effectiveness



■ Breach ■ Damage ■ Ext. Penetration

Source: Computer Security Institute, Computer Crime and Security Survey, April 7, 2002

# Hacker Methods: Footprinting

- Footprinting - discovering an organization's network information:
  - Internet - domain names, network blocks, TCP services, system architecture, IDS, ACLs, banners, routing tables, SNMP, etc.
  - Intranet - protocols in use (e.g., IP, IPX, NetBUI, etc.), network blocks, IP addresses of reachable systems, etc.
  - Remote access - VPNs & related protocols, phone numbers
  - Extranet - Access control mechanism, connection origination/destination.

# Hacker Methods: Scanning

- Scanning - discovering which systems are alive and what they are running.
  - Ping Sweeps - sending ICMP ECHO(type 8) packets to target systems in a range of IP addresses to get ICMP ECHO\_REPLY.
  - Port Scans - connecting to TCP (or UDP) ports on target system to identify services that are running.
  - Active Stack Fingerprinting - sending packets to target system and examining IP stack to detect an O/S specific implementation (e.g., FIN packet probe, TCP initial window size, ACK value, ICMP message quoting, etc.).
  - Passive Stack Fingerprinting - passively monitoring network traffic for same purpose as above.

## Automated tools

- 1) Superscan - [www.foundstone.com/rdlabs/termsfuse.php?filename=superscan.exe](http://www.foundstone.com/rdlabs/termsfuse.php?filename=superscan.exe)
- 2) NetScanTools Pro 2000 - [www.nwpsw.com](http://www.nwpsw.com)

ICMP: Internet Control Messaging Protocol  
TCP: Transmission Control Protocol  
UDP: User Datagram Protocol

# Hacker Methods: Enumeration

- Enumeration - identifying valid information about the following areas:
  - Network Resources and Shares
  - Users and Groups
  - Applications and Banners
- Enumeration techniques are O/S specific, including:
  - Password guessing
  - Eavesdropping on network password exchange
  - Denial-of-Service (DOS)
  - Buffer Overflows

## Buffer Overflow Exploiting

- 1) Phrack 49/14 & 55/15, [www.phrack.org](http://www.phrack.org)
- 2) [www.cultdeadcow.com](http://www.cultdeadcow.com) (general hacking stuff)

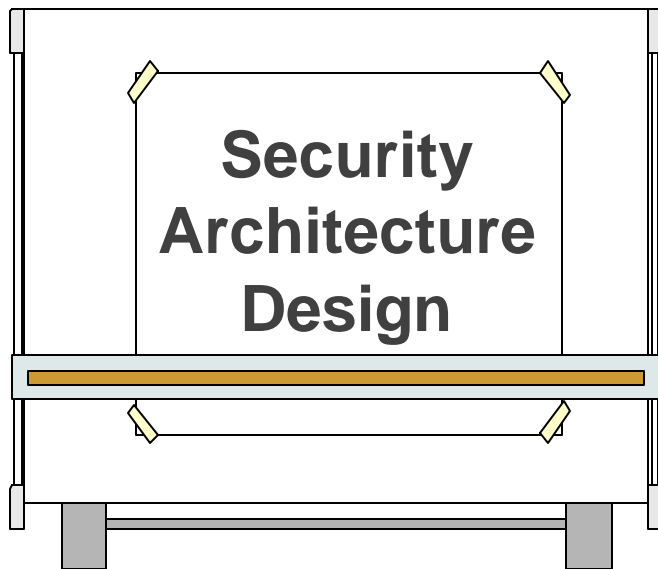
# Critical Observations

- There are a lot of hacker tools & techniques available.
- Websites are devoted to hacking and hacker training.
- IT skill sets are global.
- Anti-virus software helps.
- Firewalls help.
- Intrusion detection systems help.
- All of them can be penetrated!

**There is no simple \$5,000 solution!**

## Why is There No Simple Security Solution?

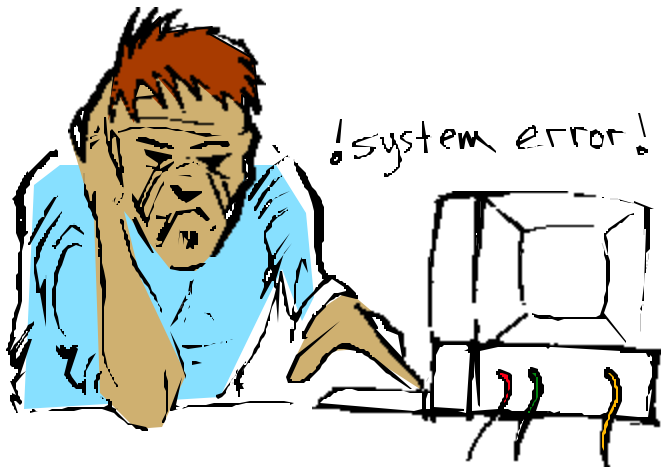
# Software Structure & Security



- Most software “tools” are really “building blocks” - application inherits all security errors and omissions.
- Application also inherits all security errors and omissions of all lower layers (O/S, utilities, etc.)
- Security must be designed into the software from the ground up - it can’t be “bolted on” ex post facto.
- Software **architecture** is critical to security and reliability - the language or development environment is not.

## Why is There No Simple Security Solution?

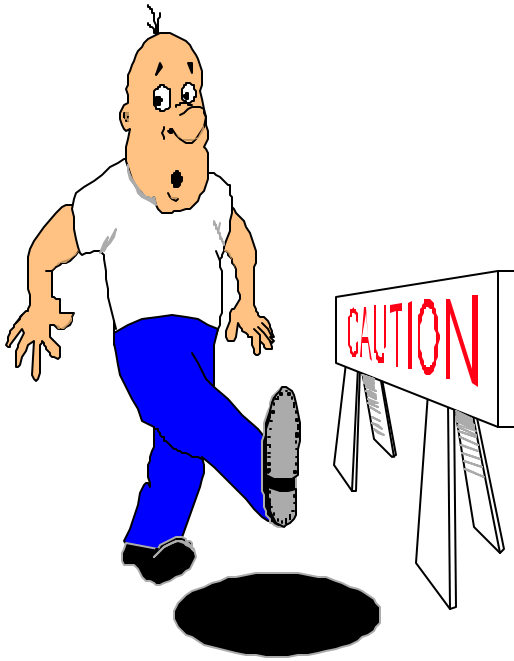
# Software Development & Security



- Software creation is largely a manual process - time consuming and fraught with errors.
- Compilers cannot check for security errors, logic errors and plain stupidity.
- O/S changes to handle new features may cause new security “leaks”.
- O/S change force apps to change because they “see” O/S only through the API (Application Programmer Interface).
- Proper security-oriented changes to desktop O/S could force changes to ***entire installed application software base.***

# Why is There No Simple Security Solution?

# Software Security Quality Assurance



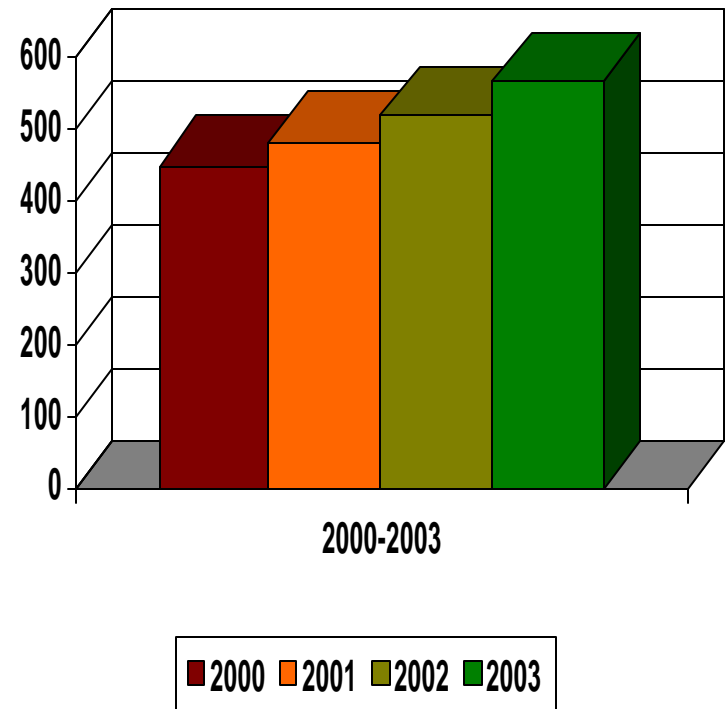
- No uniform quality standards.
- No uniform security standards.
- No uniform testing standards.
- No uniform production standards.
- Not enough research is done in this area.
- Not taught in universities.
- Not enough knowledge available.

## Why is There No Simple Security Solution?

# Economic & Social Forces

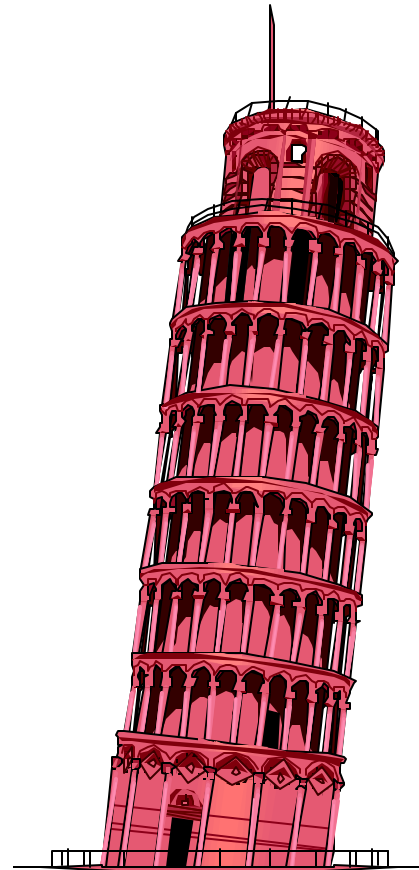
- Demand for programming has far exceeded competent supply.
- Growth rate of IT market is 9%
- Top 10 vendors have 55% of market share & 67% of workforce in packaged software.
- Software industry is not well-understood.
- Market surveys verify that consumers want new features (whether they work well or not) - bug fixes are not important!
- Time-to-market controls what's in the final O/S and app releases.
- Until 9/11/01, security has had a high cost and low perceived market value.
- No significant consequences for bad design or bad perform

IT Spending 2000-2003  
(Billions of Dollars)



# Software's Vertical Structure

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

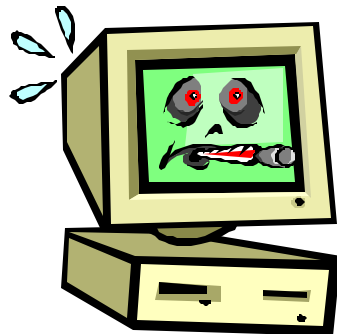


- **Libraries**
- **Books**
- **Chapters**
- **Paragraphs**
- **Sentences**
- **Words**
- **Alphabet**

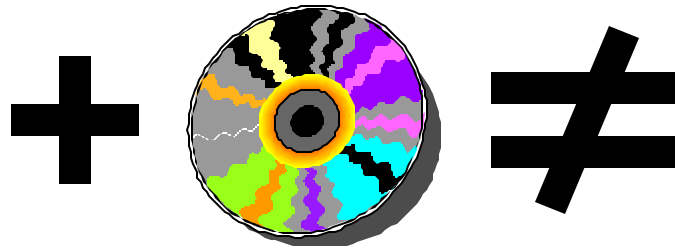
- 1) Layers made by different & competing vendors
- 2) Minimal universal standards between layers
- 3) inter-layer interfaces changing often and without warning

# Security Implications of Software's Vertical Structure

- Security cannot be just “added-on”.
- No “magic button” to add security.
- A mythical security “add-on” will not be compatible with existing IT infrastructure.
- A “magic-CD” that will 100% protect your desktop **without any other changes** is not possible now.



Insecure System

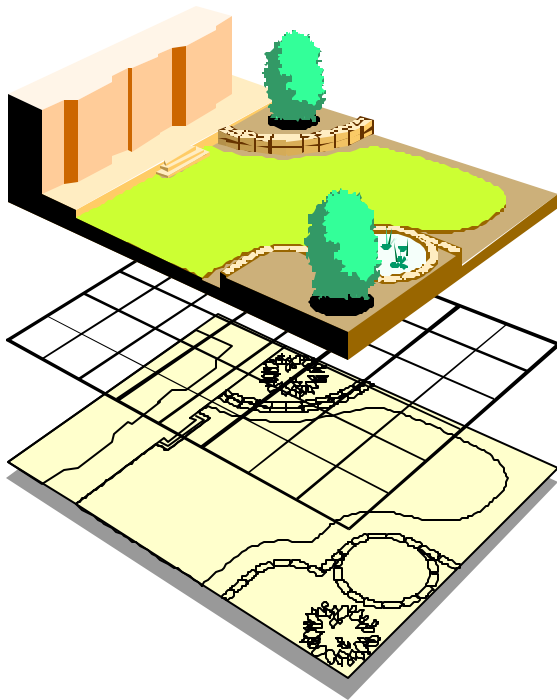


“Magic Software”



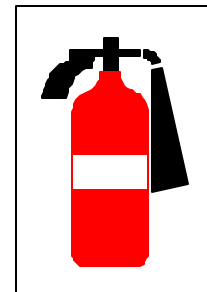
Secure System

# Security Conclusions

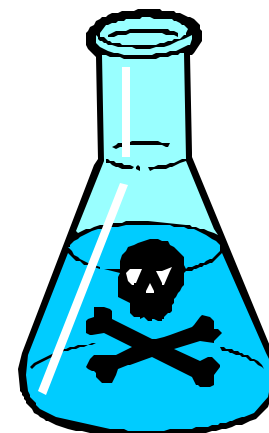


- Market is demand-driven.
- Market demand starting to provide incentives for better security design.
- Going over each line of code won't fix the security problem.
- **Must redesign the O/S, and application and network software architecture.**
- **Security is more of an economic issue than a technical one.**

# Common Security Questions



- How can I protect my PC?
- How can I prevent viruses from damaging my IT systems?
- How can I clean up the damage from viruses?
- How can I prevent malicious attacks from stealing data
- How can I prevent malicious attacks from planting a time bomb?

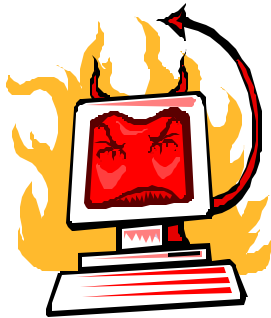


# Very Useful, Achievable Objectives

- Protect data assets
- Provide continuity of operations
- Make IT systems tamper-proof



# Partly Useful but Unachievable Objectives

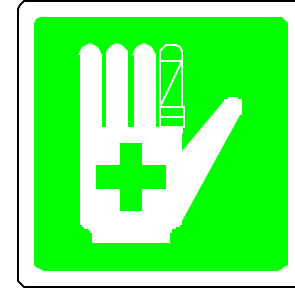


- Prevent cyber attacks
- 100% “Cyber-shield” each desktop
- Prevent all human errors
- Prevent “insider attacks”

# Firewalls & Anti-Virus Software

## What They Can Do

- Firewalls help protect desktops against network transmitted viruses.
- Anti-virus software scans and finds instances of known viruses.



## What They Cannot Do

- Prevent damage from new viral strains.
- Clean up damage from cyber attacks.
- Prevent critical data loss from cyber-thieves.
- Prevent damage from operator error(s)
- Ensure proper operation of application software
- Make the IT system tamper-proof.
- Provide 100% security.



# The OSI 7-Layer Model

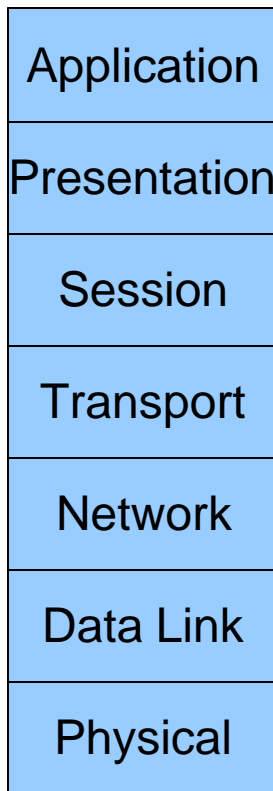
Of network-oriented software

## Protection Methods

Essentially none. No mechanisms for guaranteeing correct program operation.

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

Concrete, M16s, MIRVs, etc.



## Design, Operation & Maintenance Responsibility

Vendor, user

ACM, IEEE, vendor, W3C, user, etc.

ISP, phone company, user, etc.

Computer manufacturer, user

# Security Research Results

- Security has been important (to “techies”) for about 40 years.
- Research on security which started in the 1960s (and through the 1980s) led to two important conclusions:
  - Discovering security flaws then fixing them one-by-one will NOT work.
  - Effective security can only be achieved by **architecting** a system in accordance with a secure system model.



***Putting out fires  
doesn't work!***

# System Security Objectives



- Secure systems control access to information.
- Only properly authorized people are allowed to read, write, create or delete information.
- Only properly authorized processes are allowed to read, write, create or delete information.

# What Secure Systems Must Do

## ■ 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

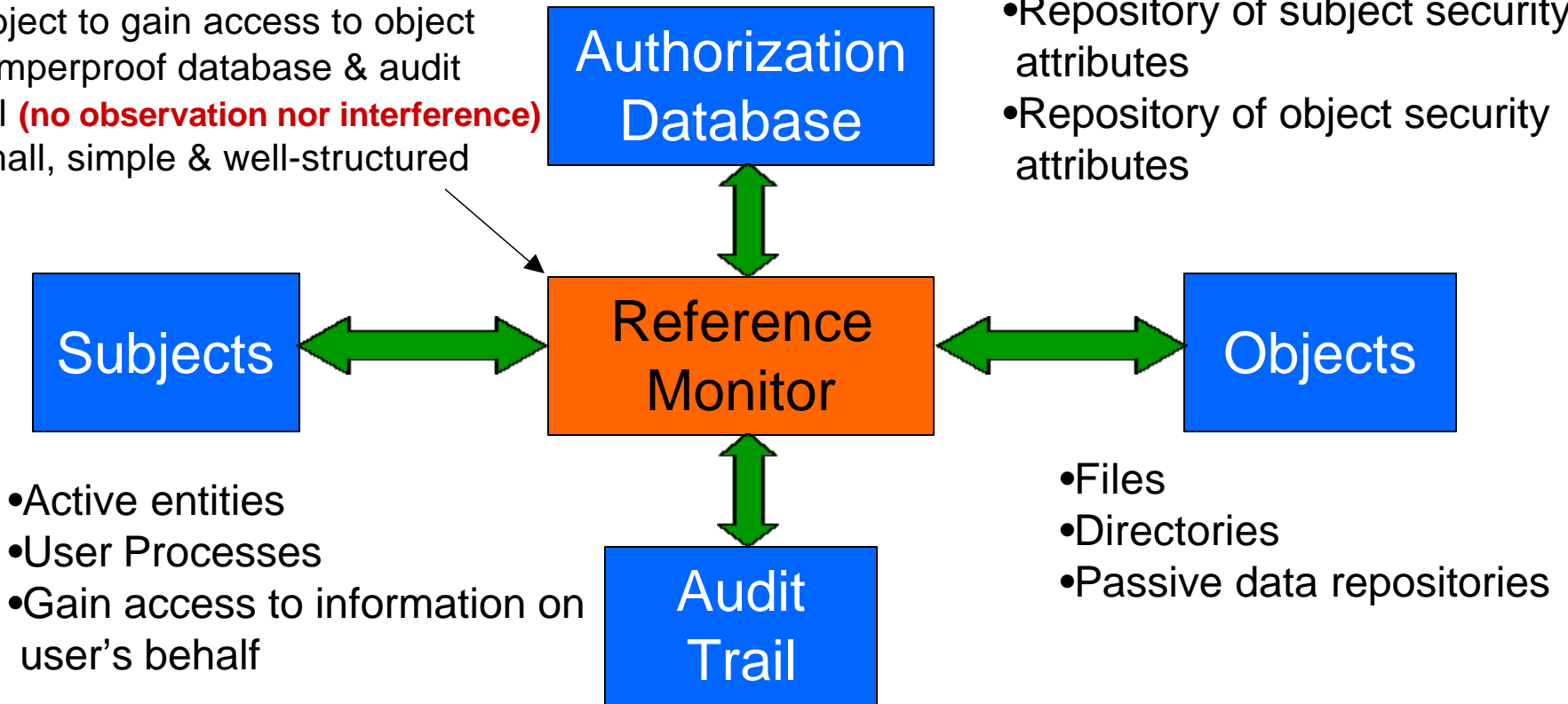
- **EAL-1 - Functionally Tested** - independent testing of selected features.
- **EAL-2 - Structurally Tested** - independent testing of selected features using limited developer design data.
- **EAL-3 - Methodically Tested & Checked** - independent testing using limited developer design data, selective 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 every attempt by subject to gain access to object
- 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 information on user's behalf

- Files
- Directories
- Passive data repositories

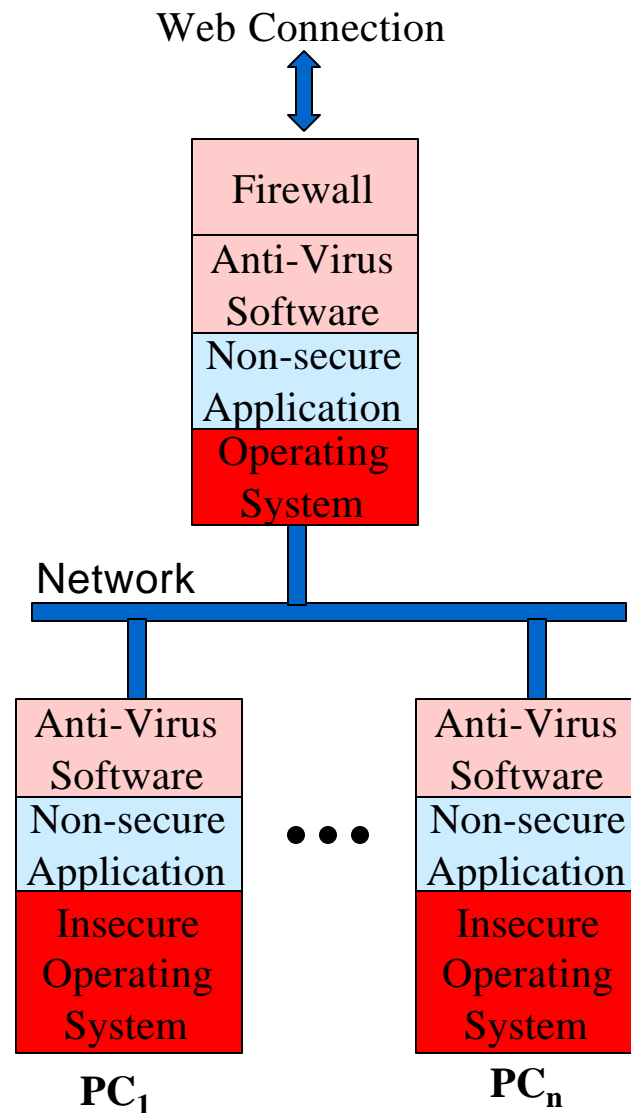
- Record of all security-related events

# So, How Do We Fix This?

(Things to Avoid)

- Avoid the traditional IT architecture - the risk exposure is too great.
- Protecting the desktop does not equate to protecting your business operations 24 x 7.
- Avoid deploying critical software on non-secure platforms. This minimizes risk exposure.
- Avoid deploying critical functionality on non-secure platforms. Critical functionality can be separate from the software that supports and/or delivers it.

## Traditional, Non-secure IT Architecture

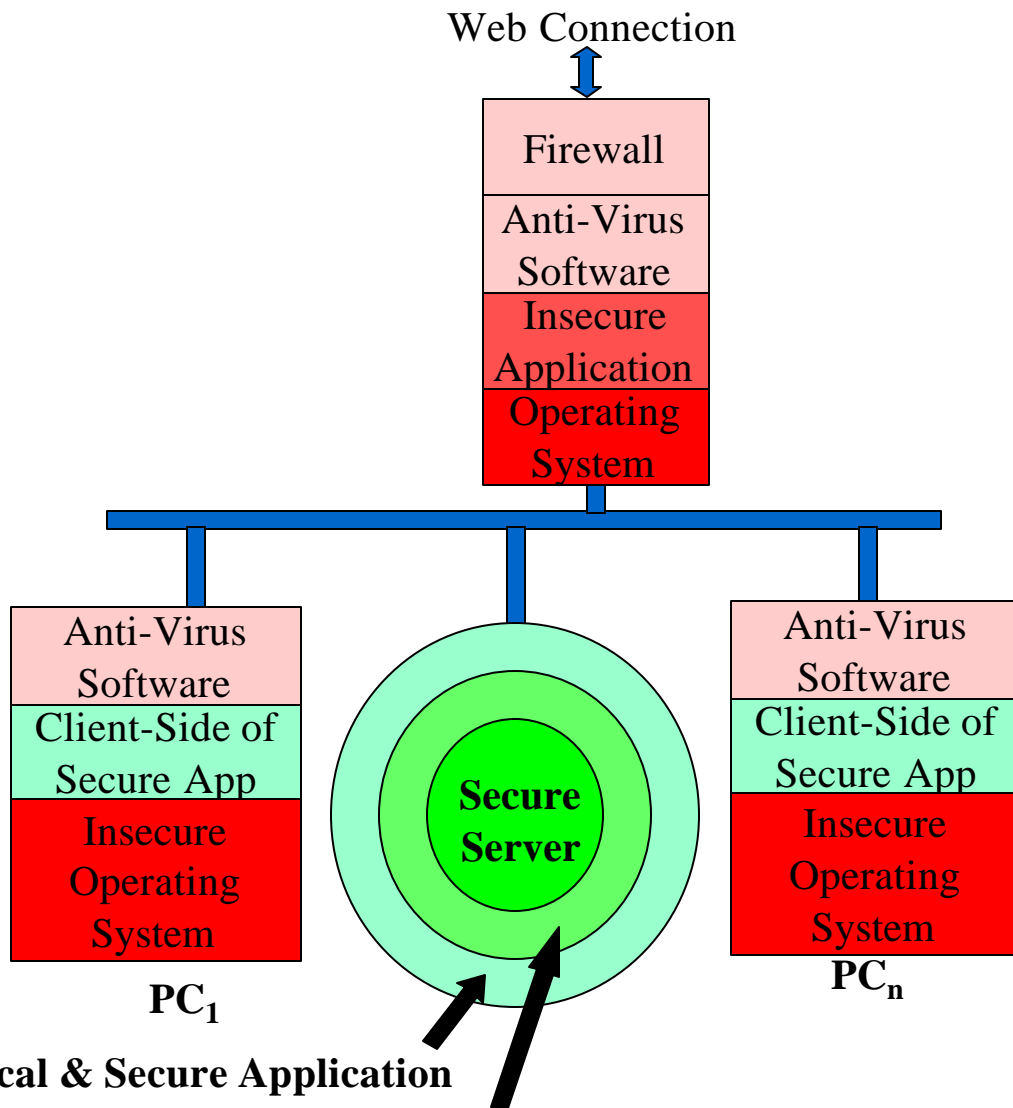


# So, How Do We Fix This?

(Things to Do)

- Migrate to this secure IT architecture (patent-pending).
- Deploy critical software on secure platform(s). This eliminates the risk exposure.
- Your critical business and operational functionality will be protected - which is what matters.
- Critical functionality will be separated from the software that supports and/or delivers it.
- Flexibility will be maximized since the desktop boxes can be upgraded at any time while preserving business operations.

## SmartData's Secure Architecture



Server-Side of Critical & Secure Application

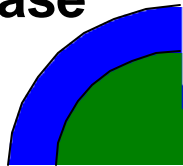
Secure Operating System (Reference Monitor)

# Secure System Summary

- Identify your critical applications and focus on them (not all applications are equally critical).
- Then, use a secure O/S for the critical applications which obeys the secure system model (Reference Monitor, etc.).
- Re-architect and/or re-deploy critical applications (client-server style) to be consistent with the secure system model and security policy.
- Re-architect and/or re-deploy critical applications in accordance with the secure IT system architecture.
- **Doing all of this will yield a virus-immune software system and ensure continuity of operations, 24 x 7 - this is what matters!**
- **This is quite feasible - and it has been done - at a reasonable cost!**

# More Suggestions

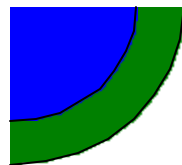
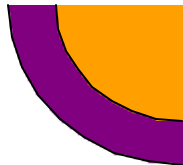
Database



GUI



Multi-Vendor-Controlled  
Interfaces

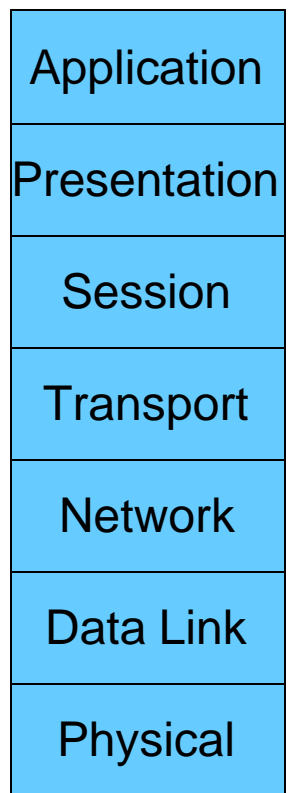
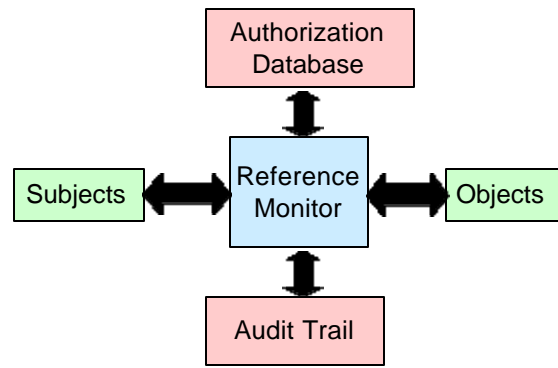


Network  
Interface

Information  
Processing

- Use a “trusted computing base” (TCB) - this assures security at the lowest layer.
- Use the TCB to implement the application security policy - via the appropriate APIs.
- Create a segregated system with a “thin” client - this reduces vulnerability from non-secure clients.
- Use the right tool for each section - this optimizes performance.
- Use only multi-vendor-controlled interfaces - this prevents rapid obsolescence & increases portability.

# Final Thoughts



- 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.
- Architect critical apps consistent with Reference Monitor structure.
- Deploy apps carefully, using all available security controls.