J. A. "Drew" Hamilton, Jr., Ph.D.

Director, Distributed Analytics & Security Institute
Director, Center for Cyber Innovation
Professor, Computer Science & Engineering

CCI
Post Office Box 9627
Mississippi State, MS 39762

Voice: (662) 325-2294
Fax: (662) 325-7692
hamilton@cci.msstate.edu

Outline
(Understanding, Applying, and Enforcing Software Security) 10%

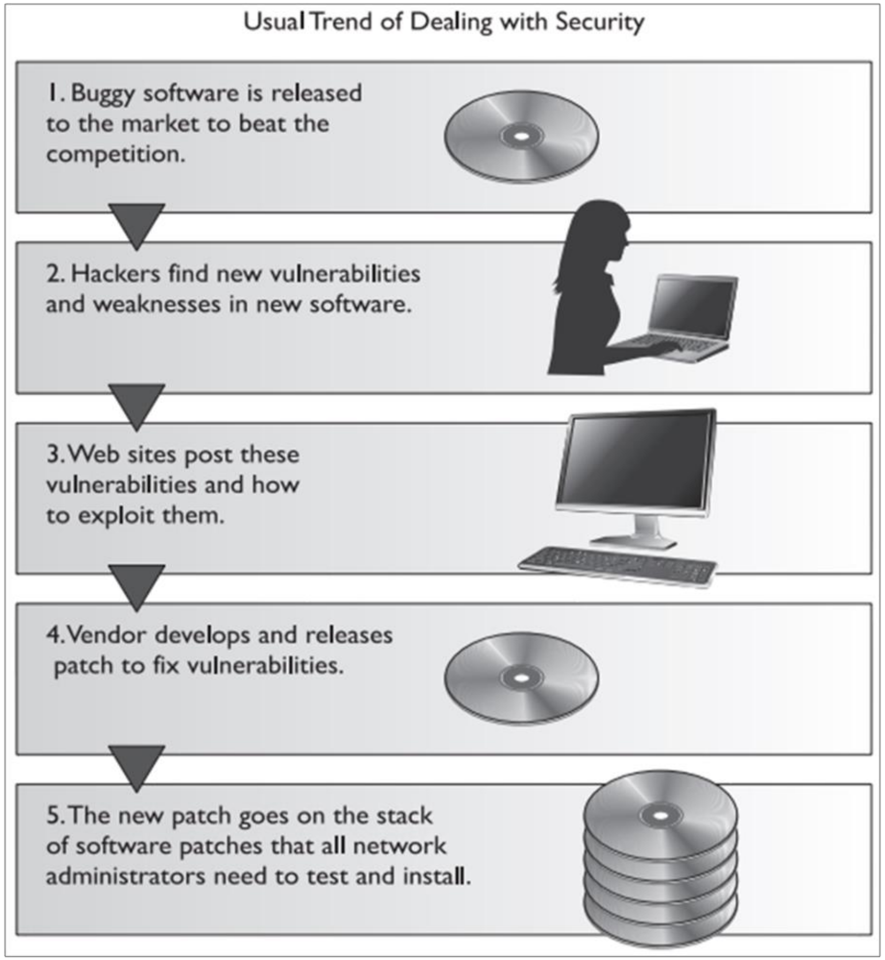
- Security in the software development lifecycle
- Development environment security controls
- Software security effectiveness
- Acquired software security impact

Security in the software development lifecycle

Reference: Dr. Devin Cook, Sandia Labs

Reference: Shon Harris

Usual Software Development



Usual Trend of Dealing with Security

1. Buggy software is released to the market to beat the competition.

2. Hackers find new vulnerabilities and weaknesses in new software.

3. Web sites post these vulnerabilities and how to exploit them.

4. Vendor develops and releases patch to fix vulnerabilities.

5. The new patch goes on the stack of software patches that all network administrators need to test and install.

- **This is the usual trend of software being released to the market and how security is dealt with.**
- **To avoid this problem, it is important to address security from the start of the development process**

Different Environments, Different Security Requirements

- Software controls can be implemented by the OS, by the application, or through the database management controls.
- OS controls mainly control a subject's access to different objects and restrict the actions within the system, but do not restrict the subject's actions within an application.
- Application and database management controls are very focused on preventing security issues within their particular domains.

Operating Systems

- User Space versus Kernel Space
- Process space protection
  - A single process is not allowed to access any of the memory allocated to other processes directly
  - Additionally, no process can directly access the memory marked as "in use" by the operating system
  - Windows NT/2000 and ALL UNIX systems provide process space protection
  - Windows 95/98/ME do NOT provide process space protection
- Palm Pilot example
  - any file stored on an internet-enabled PalmOS device is accessible by any application running on the Palm
- Generally, in an advanced OS, user-level processes need to use kernel services to access devices

Case Studies

- **The Microsoft Fallacy**
  - **Microsoft makes bad software**
  - **Microsoft software is closed source**
  - **Therefore all closed source software is bad**
- **The Java Fallacy**
  - **If we keep fixing holes in a given piece of software, eventually, the software will be completely secure**
  - **From 1996 to 2001, more than 19 serious security flaws found in Java**

General Security Implementation

- Inputs should be checked for correct length, format, and type.
  - Hexadecimal or UNICODE should not be accepted for ASCII input
- Only accept reasonable values, and do not allow bogus entries to be passed to logic calculations
- When a secure application is installed always default to “No Access”
  - Incorrect implementations are to blame for the majority of security issues in a network

Failure States

- If an application fails for any reason, it should return to a safe and more secure state.
    - For example, an OS restarting and returning the user to the login screen
- If an application is running in privileged mode and fails, but is not shut down properly an attacker can possibly use this to access privileged information.

System Development

- Security is important throughout the development lifecycle, you can't just add it in at the very end.
- There are many different development models that utilize different life cycles, but all models contain in some form or fashion:
  - Project initiation
  - Functional design analysis and planning
  - System design specifications
  - Software development
  - Installation
  - Operations and maintenance
  - Disposal

Project Initiation

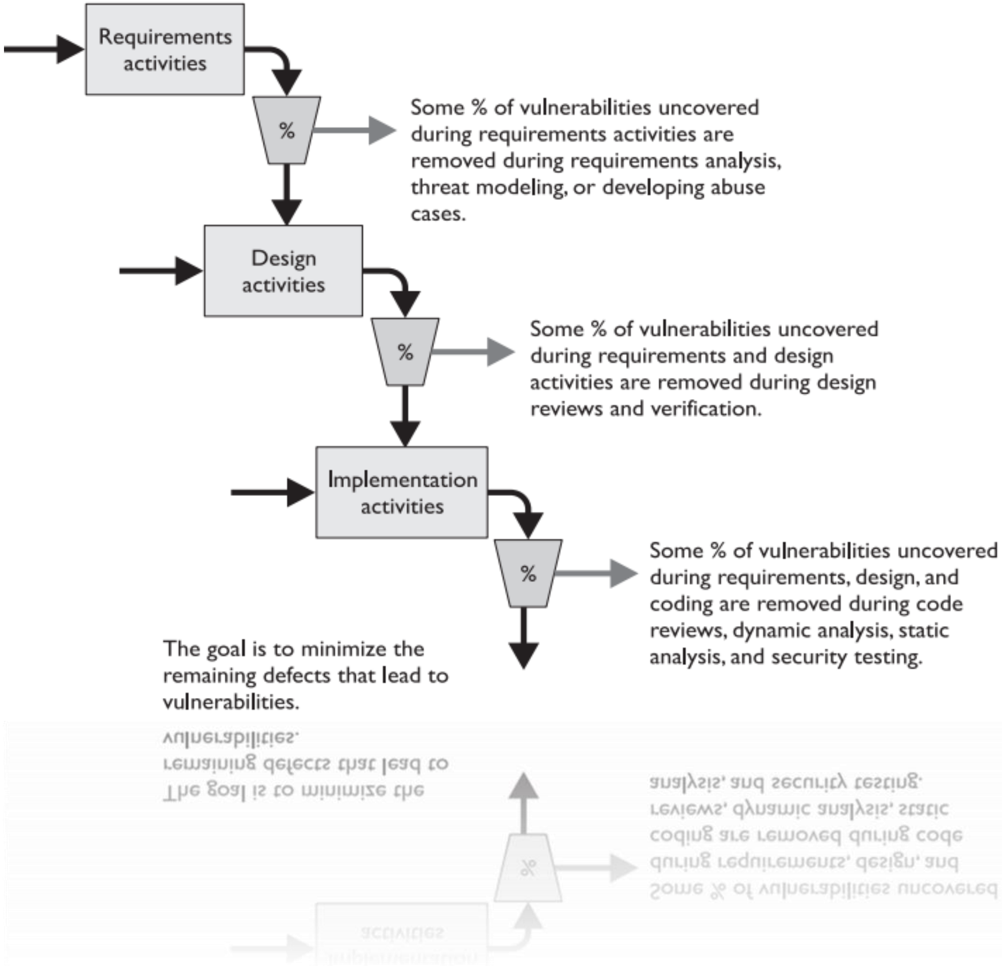
- Phase when everyone involved attempts to understand why the project is needed and what the scope of the project entails
- User needs are identified and basic security objectives of the product are acknowledged
- Risk management processes are established

Risk Management and Analysis

- It is important to analyze and manage risks that the software being developed faces.
- Risks must be identified, categorized, and prioritized based on their severity
- These risks should be addressed in the design and architecture of the software as well as in the functionality it provides

Functional Design Analysis and Planning

- **This phase addresses the functionality required of the product and is captured in the design document.**
- **With regard to security issues, this is where high-level questions are asked.**



Requirements activities

%

Some % of vulnerabilities uncovered during requirements activities are removed during requirements analysis, threat modeling, or developing abuse cases.

Design activities

%

Some % of vulnerabilities uncovered during requirements and design activities are removed during design reviews and verification.

Implementation activities

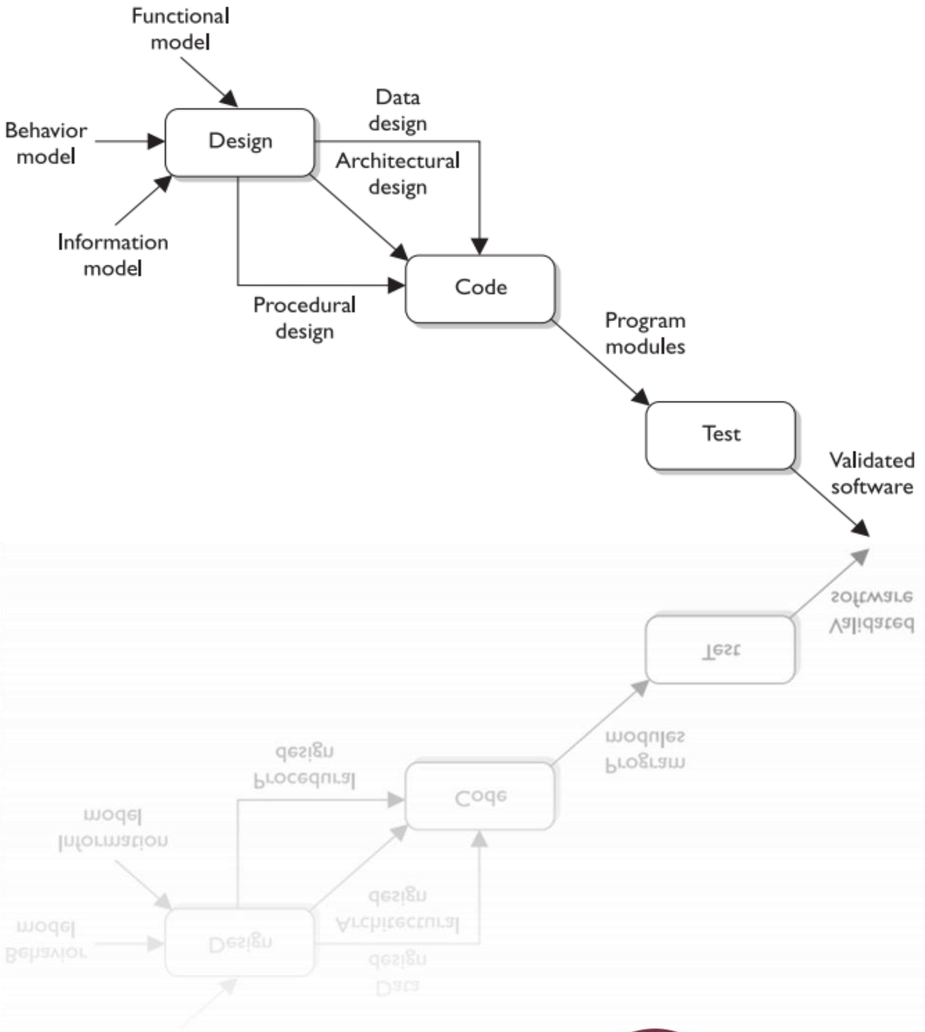
%

Some % of vulnerabilities uncovered during requirements, design, and coding are removed during code reviews, dynamic analysis, static analysis, and security testing.

The goal is to minimize the remaining defects that lead to vulnerabilities.

System Design Specifications

- **Software requirements come from three models:**
  - **Information model- Dictates the type of information to be processed and how it will be processed**
  - **Functional model- Outlines the tasks and functions the application needs to carry out**
  - **Behavioral model- Explains the states the application will be in during and after specific transitions take place**
- **The work breakdown structure (WBS) for future stages is confirmed**



Verification vs. Validation

- Verification determines if the product accurately represents and meets the specifications
- Validation determines if the product provides the necessary solution for the intended real world problem.

Software Development

- In this stage, programmers should code in a way that does not permit software compromises
- Unit testing and formal test should also be done during this phase

Testing Types

- **Unit testing- an individual component in a controlled environment is tested to validate data structure, logic, and boundary conditions**
- **Integration Testing- verifying that the components work together as outlined in design specifications**
- **Acceptance testing- ensuring that the code meets customer requirements**
- **Regression testing- after a change to a system takes place, retesting to ensure functionality, performance and protection**

Separation of Duties

- Different environmental types (development, testing, production) should be properly separated, and functionality and operations should not overlap.
- Developers should not have access to code used in production
- Developers should not be in charge of formal testing

Installation/Implementation

- This phase focuses on how to use and operate the developed system or application.
- Certification and accreditation happen in this phase
  - Certification is the process of reviewing and evaluating security controls and functionality
  - Accreditation is the formal acceptance of the system by management and an explicit acceptance of risk

Operational and Maintenance

- The initial part of this phase includes configuring the new system and inserting it properly into the network and environment.
- Operational assurance is carried out by continually conducting vulnerability tests, monitoring activities, and auditing events
- If major changes occur to the system, product, or environment a new risk analysis may become necessary

Disposal

- **This phase handles the removal of the system**
- **Data may need to be archived, backed up to another system, discarded, or destroyed**

Programming Languages

- A set of instructions and rules that tell the computer what operations to perform.
- Languages have evolved in "generations"
  - 1st Generation: Machine language
  - 2nd Generation: Assembly language
  - 3rd Generation: High-level language
    - Ada, COBOL, BASIC, FORTRAN, Pascal, C, C+, C++, C#, Java
  - 4th Generation: Very high-level language
    - SQL, JavaScript, Perl, SGML (Standard General Markup Language): HTML, XML, SAML, XACML.
  - 5th Generation: Natural language
    - BPEL (Business Process Execution Language), BQEL (Business Query Language)

Programming Languages

- Assembler – program that translates an assembly language program into machine language.
  - Assembly Language → Machine Language.
- Compiler – translates a high-level language into machine language.
  - High-level Language (3rd Gen.) → Machine Language.
- Interpreter – instead of compiling a program at once, the interpreter translates it instruction-by-instruction. It has a fetch and execute cycle.
  - Very high-level Language (4th Gen.) → Interpreter instruction → Machine Language.

Software Development Methods

- Waterfall
- Spiral
- Structured Programming Development
- Iterative Development
- Modified Prototype Model (MPM)
- Rapid Application Development (RAD)
- Reuse Model
- Cleanroom
- Component-Based Development
- Extreme Programming
- Joint Analysis Development (JAD)
- Exploratory Model

Computer-Aided Software Engineering

- **Computer-aided software engineering (CASE)- the use of tools to create and manage software.**
  - **First case tools were translators, compilers, assemblers, linkers and loaders**
  - **Graduated into program editors, debuggers, code analyzers, and version-control mechanisms**
- **CASE tools are tools used for one specific phase of the development life cycle**
- **When the automation covers the complete life cycle of a product, the tools are referred to as integrated computer-aided software engineering (I-CASE) tools.**

Definitions

- **Secure Design Methodology- ensures early implementation of security policies and techniques rather than bolting them on as an afterthought**
- **Attack surface analytics- these techniques provide a structured process for analyzing program entry points**
- **Secure Development Methodology- emphasizes constantly analyzing the developed code for flaws and vulnerabilities, instead of putting it off until the software has been completely developed**
  - **This methodology involves code reviews and use of version control technologies**

Security Testing

- Is comprehensive analysis technique that examines programs under artificially created attack scenarios.
- Encompasses both manual and automatic tests
- Automated tests generally use fuzzers, vulnerability scanners, and code scanners
- Manual tests are used for aspects that require human intuition

Change Control

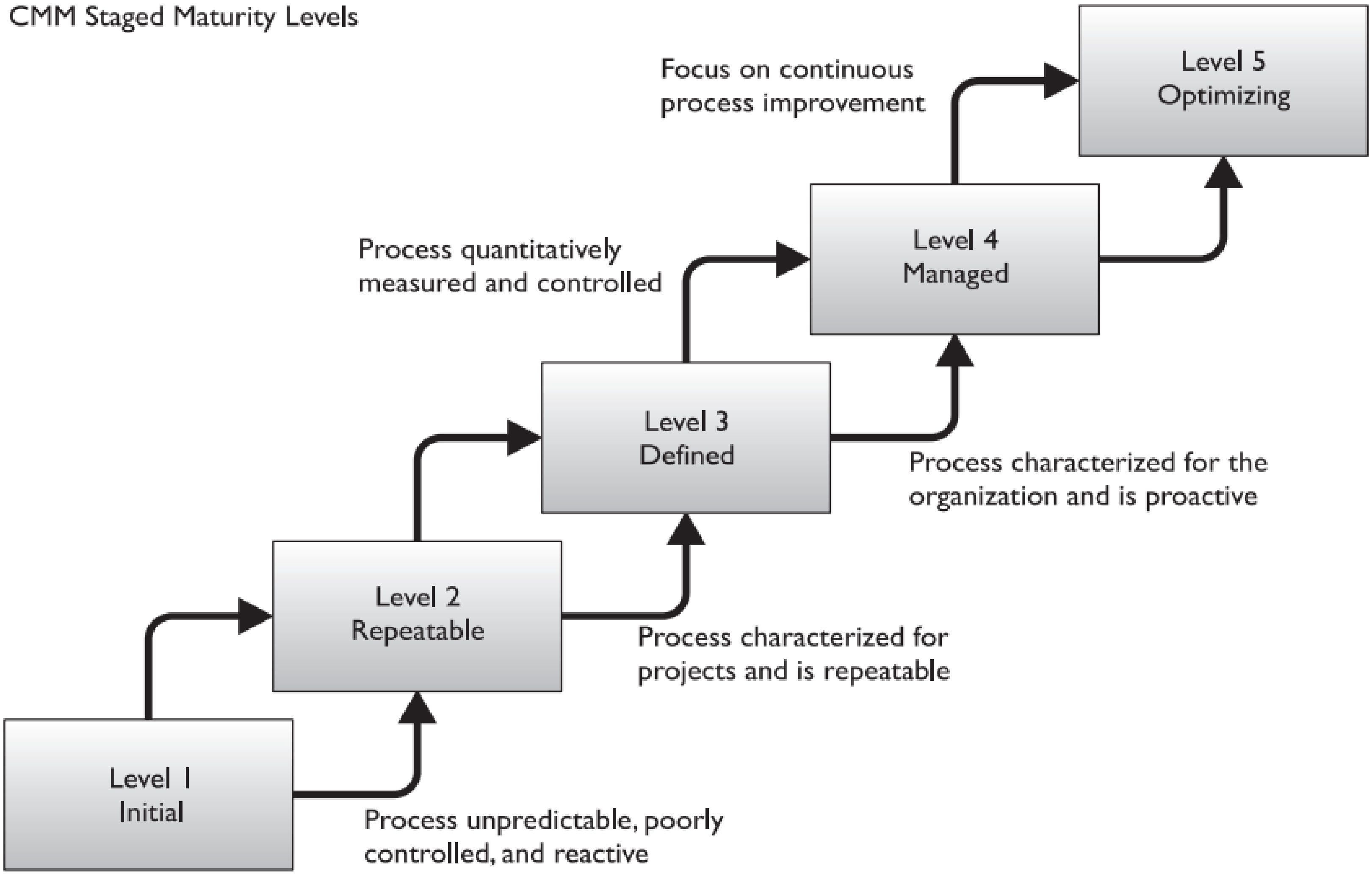
- Is the process of controlling the life cycle of an application and documenting the necessary change control activities
- Changes should be controlled to make sure they are approved, incorporated properly, and do not affect the original functionality in an adverse way
- Changes should not compromise security and management must approve the change before implementation

Capability Maturity Model (CMM)

- Describes procedures, principles, and practices that underlie software development process maturity
- The goal is to standardize the approach to software development so that it is repeatable across many different groups
- Five maturity levels:
  - Initial
  - Repeatable
  - Defined
  - Managed
  - Optimizing

CMM

CMM Staged Maturity Levels



Level 1
Initial

Process unpredictable, poorly controlled, and reactive

Level 2
Repeatable

Process characterized for projects and is repeatable

Level 3
Defined

Process characterized for the organization and is proactive

Level 4
Managed

Process quantitatively measured and controlled

Level 5
Optimizing

Focus on continuous process improvement

Definitions

- Software escrow- a third party keeps a copy of the source code, and possibly other materials, which it will release to the customer only if specific circumstances arise, mainly if the vendor who developed the code goes out of business or for some reason is not meeting its obligations and responsibilities.
- Interpreter- translates one command at a time during execution
- Compilers- translate large sections of code at a time
- Assemblers- translate assembly language into machine language

Application Development Methodology

- Machine language is in a form that the processor can understand and work with directly
- Assembly and high-level languages cannot be understood directly and must be translated, turning it into machine language
- Generations of Languages:
  - Generation one: machine language
  - Generation two: assembly language
  - Generation three: high-level language
  - Generation four: very high-level language
  - Generation five: natural language

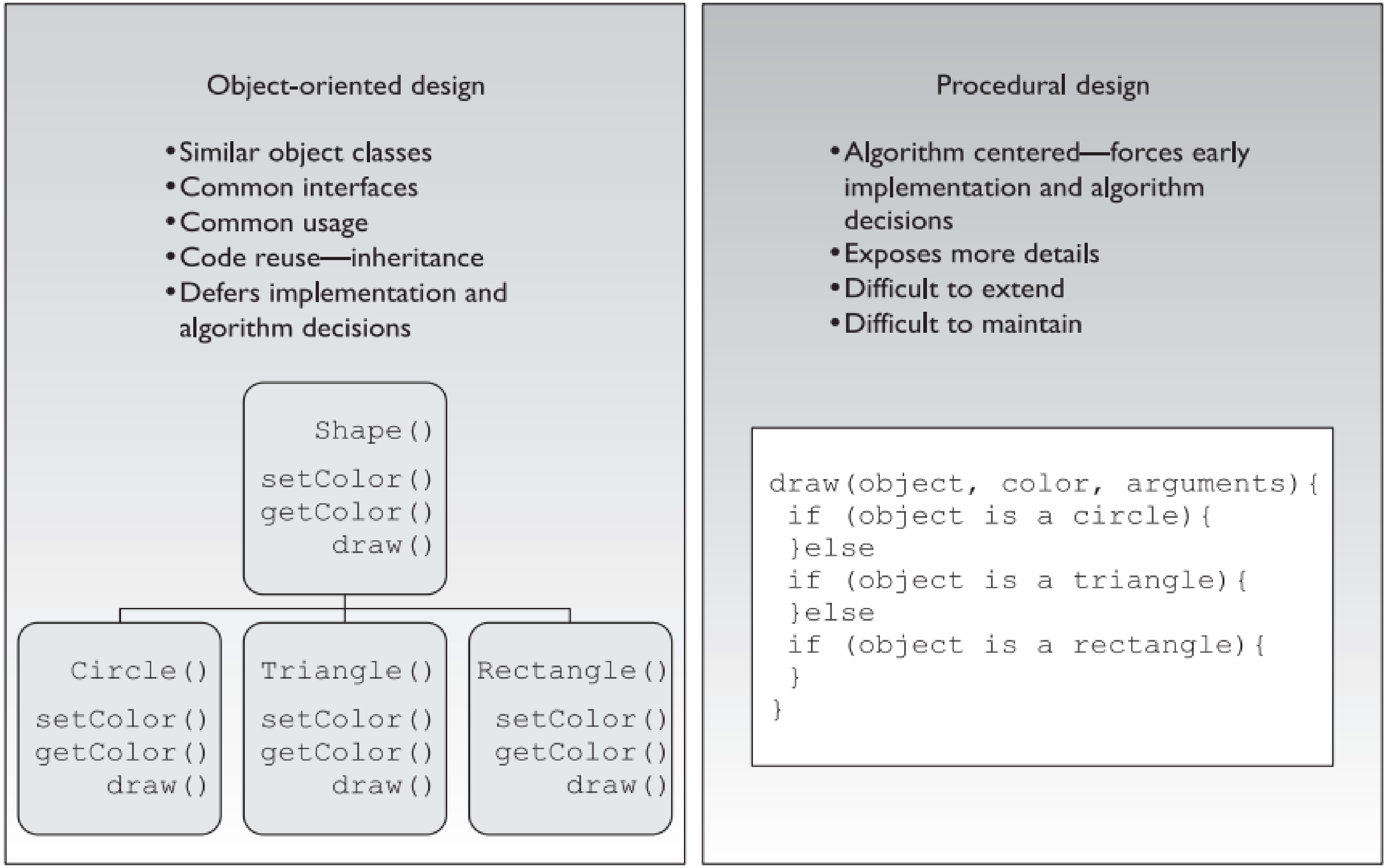
Object-Oriented Concepts

- Object-oriented programming (OOP)-works with classes and objects
- Object- members, or instances, of classes
- Non-OOP applications are written as monolithic entities
- OOP breaks down the code into smaller pieces of classes and objects
- Benefits from OOP
  - Modularity, reusability, naturalness, etc

Object-Oriented Concepts

- Methods- is the functionality or procedure an object can carry out
- The objects encapsulate the attribute values, which means this information is packaged under one name and can be reused as one entity by other objects
- Messages are used to communicate between objects.
- The messages are delivered to the shared portion of the object, the interface API

OOP vs. Procedural



Object-oriented design

- Similar object classes
- Common interfaces
- Common usage
- Code reuse—inheritance
- Defers implementation and algorithm decisions

Shape()
setColor()
getColor()
draw()

Circle()
setColor()
getColor()
draw()

Triangle()
setColor()
getColor()
draw()

Rectangle()
setColor()
getColor()
draw()

Procedural design

- Algorithm centered—forces early implementation and algorithm decisions
- Exposes more details
- Difficult to extend
- Difficult to maintain

```
draw(object, color, arguments){
 if (object is a circle){
 }else
 if (object is a triangle){
 }else
 if (object is a rectangle){
 }
}
```

Object-Oriented Concepts

- **Abstraction is the capability to suppress unnecessary details so the important, inherent properties can be examined and reviewed**
- **Data and operations internal to objects are hidden from other objects, which is referred to as data hiding**
- **If an object does not require much interaction with other modules, it has low coupling**
- **Cohesion reflects how many different types of tasks a module can carry out**
  - **If a module carries out only one task or several similar tasks it has high cohesion**
- **The higher the cohesion and the lower the coupling the better**

Object-Oriented Concepts

- **Polymorphism- two objects can receive the same input and have different outputs**
- **Polymorphism can also take place in the following ex:**
  - **Object A and Object B are created from the same parent class, but Object B is also under a subclass.**
  - **When Object A and Object B receive the same input they would result in different outputs**
- **Object-oriented analysis (OOA) is the process of classifying objects that will be appropriate for the solution**
- **Object-oriented design (OOD) creates a representation of a real world problem and maps it to a software solution using OOP**

Definitions

- **Structured analysis approach- looks at all objects and subjects of an application and maps the interrelationships, communications paths, and inheritance properties.**
- **Data modeling considers data independently of the way the data is processed and of the components that process the data**
- **Software Architecture involves the process of partitioning requirements into individual problems that can be solved by individual software solutions**
- **Data structure is a representation of the logical relationship between elements of data**

Distributed Computing

- Distributed computing means the smaller part (client) of the application can run on different systems and the larger piece (server) of the application runs on a server.
- Three main inter-component communication architectures used today are:
  - Common Object Request Broker Architecture (CORBA)
  - Microsoft COM model
  - EJB (Enterprise Java Beans)

Common Object Request Broker Architecture (CORBA)

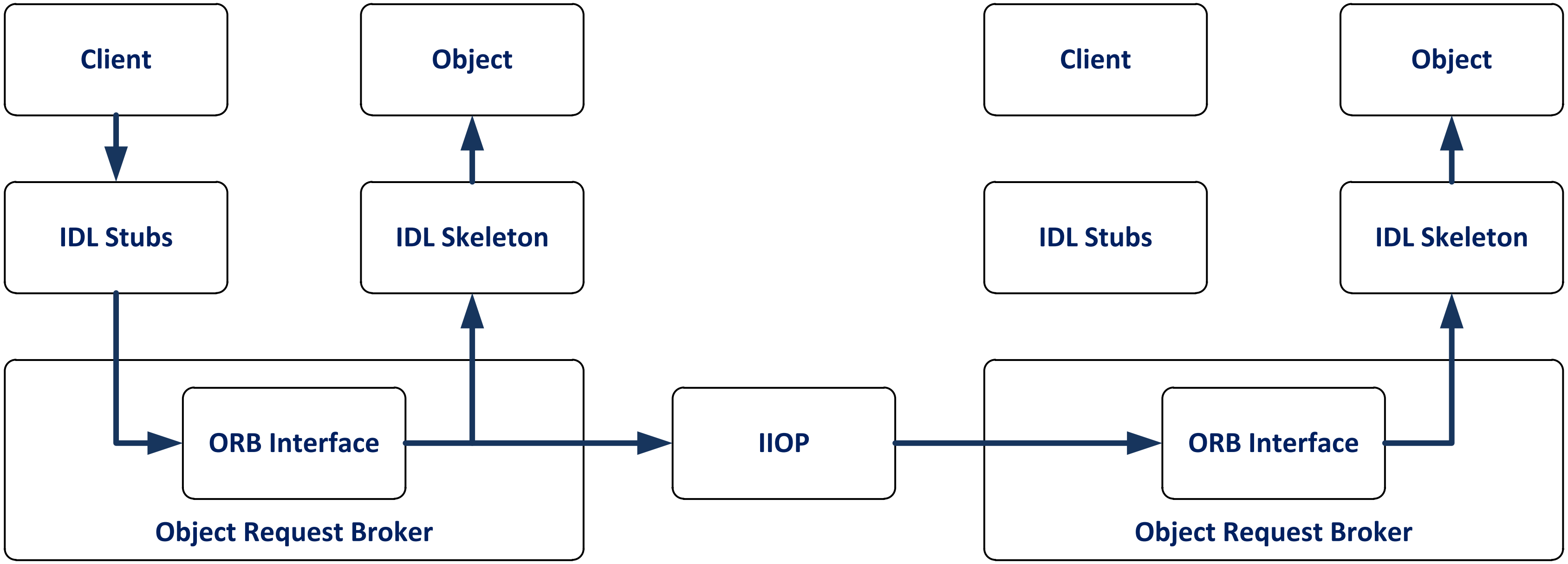
- **Is an open object-oriented standard architecture developed by the Object Management Group (OMG)**
- **This standard defines the APIs, communications protocol, and client/server communication methods to allow applications written in different languages on different platforms to work together**

CORBA

- Contains two main parts:
  - System oriented components (object request brokers [ORBs] and object services)
  - Application-oriented components (application objects and common facilities)
- The ORB manages all communications between components and enables them to interact in a heterogeneous and distributed environment.
- Using CORBA enables an application to be usable with many different types of ORBs.

CORBA

Remote Invocation Mechanism



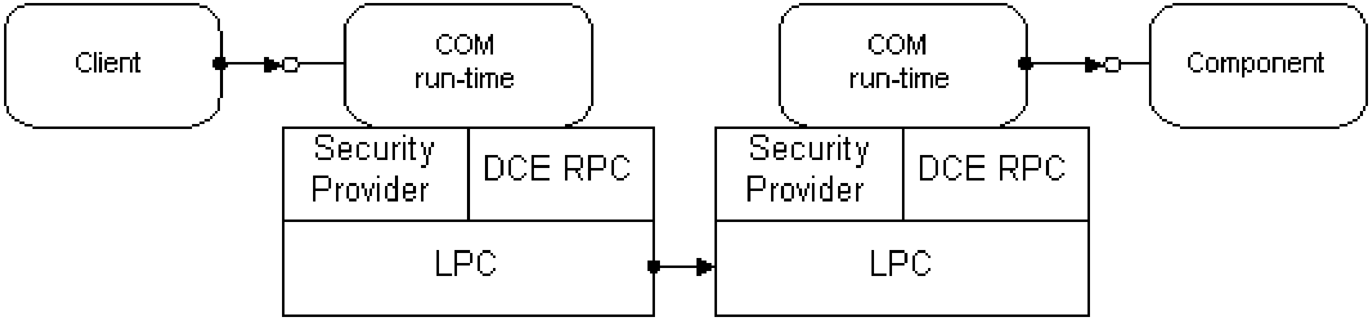
- **CORBA uses Interface Definition Language (IDL) to describe interface requirements.**
- **CORBA uses Internet Inter-ORB Protocol (IIOP) to communicate between Object Request Brokers (ORBs).**

Source: http://www.omg.org/gettingstarted/corbafaq.htm

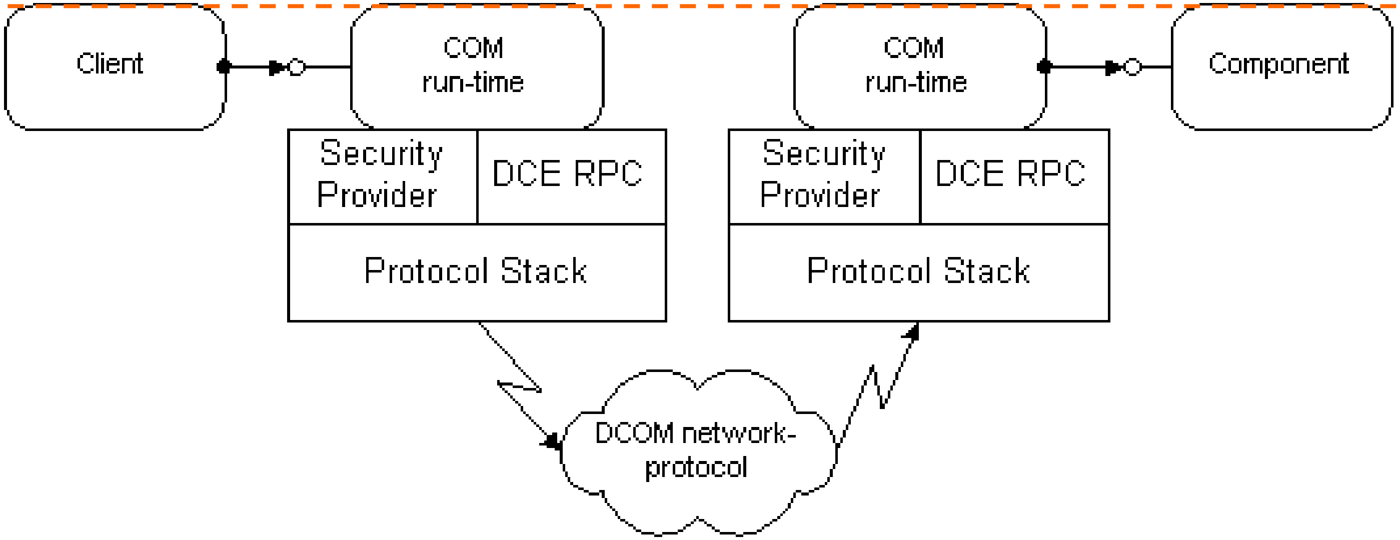
COM and DCOM

- Component Object Model (COM) is a model that allows for inter-process communication within one application or between applications on the same computer.
- Distributed Component Object Model (DCOM) supports the same model for component interaction, and also supports distributed IPC.

Component Object Models



Source: http://technet.microsoft.com/en-us/library/cc722925.aspx



ActiveX

- A loosely defined set of technologies developed by Microsoft. ActiveX is a set of technologies that enables interactive contents for web.
- Elements of ActiveX technologies:
  - ActiveX Controls: interactive objects in a web page that provides user interaction functions.
  - ActiveX Documents: enable user to view non-HTML documents (e.g. Word, Excel, or PPT)
  - ActiveX Scripting Controls: integrated controls for ActiveX controls and/or Java Applets from web browser or server.
  - Java Virtual Machine (JVM): enables web browser (IE) to run Java applets and integrate with ActiveX controls.
  - ActiveX Server Framework: provide web server functions to support the above functions plus objects for database access and online transactions.

Enterprise JavaBeans (EJB)

- Is a structural design for the development and implementation of distributed applications written in JAVA
- EJB provides interfaces and methods to allow different applications to be able to communicate across a networked environment

Object Linking and Embedding (OLE)

- OLE enables a program to call another program (linking) and permits a piece of data to be inserted inside another program or document (embedding)

Simple Object Access Protocol (SOAP)

- SOAP allows programs created with different programming languages and running on different operating systems to interact without compatibility issues
- SOAP actually defines an XML schema or structure of how communication is going to take place
- This communication commonly takes place over HTTP because its ready availability

Distributed Computing Environment (DCE)

- Is a set of management services that sits on the top of the network layer and provides services to the applications above it
- DCE and DCOM offer much of the same functionality
  - DCOM was developed by Microsoft and is much more proprietary in nature
- Both systems use identifiers to uniquely identify users, resources, and components within an environment
  - Globally Unique Identifier (GUID) is used in DCOM
  - Universal Unique Identifier (UUID) is used in DCE

DCE

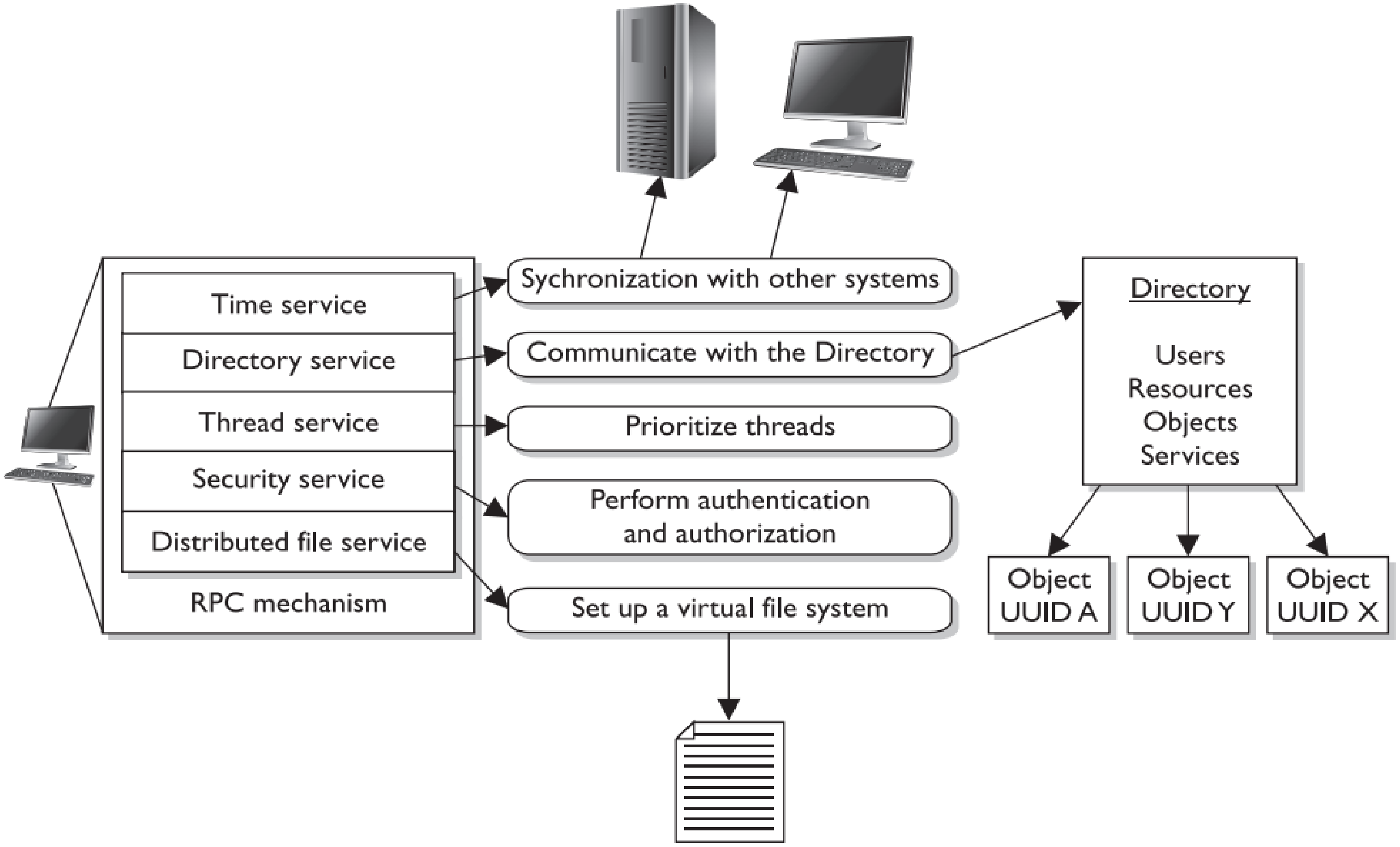


Figure 11-24 DCE provides many services, which are all wrapped into one technology.

Expert Systems

- Expert systems are used to mimic human reasoning and replace human experts
- Expert systems use knowledge bases full of facts, rules of thumb, and expert advice.
- They also use inference engine processing, automatic logical processing, and general methods of searching for problem solutions

Artificial Neural Networks (ANN)

- **ANNs attempt to mimic the brain by using units that react like neurons**
- **ANNs can learn from experiences and can match patterns that regular programs and systems cannot**

Web Security: Threats

- **Vandalism- usually involves replacing the legitimate graphics and titles on web sites with ones modified by the attacker**
- **Financial Fraud- whenever transactions are involved the potential for fraud is present.**
- **Privileged Access- mechanisms for remote access can be abused by attackers**
- **Theft of Transaction Information- target of hackers to steal identities, credit cards, and information to commit other fraud**
- **Denial of Service (DOS) Attacks- technique of overwhelming a system or service with requests that tie up resources until legitimate requests cannot be fulfilled**

Web Security: Defenses

- SSL- will encrypt a user's information, but will also encrypt any attacker's traffic from IDS systems and do nothing to protect the web application itself
- Quality Assurance Process- a quality assurance process is effective in ensuring that the servers are installed and configured properly
- Web Application Firewall- application-layer firewalls that do deep packet inspection
- Intrusion Prevention Systems (IPS)- IPS can actually prevent attacks as opposed to IDS that only detect attacks
  - These can run the risk of degrading the web application's performance

SYN Proxies on Firewalls

- One of the most common DOS attacks are SYN floods.
  - Where attackers send fake SYN requests to servers in an attempt to exhaust the amount of legitimate requests the server can maintain
- SYN proxies have a predefined threshold that, after which the firewall is on guard. If requests continue then the firewall has the option to drop any un-established connections

Specific Threats to the Web Environment

- Information Gathering
- Administrative Interfaces
- Authentication and Access Control
- Configuration Management
- Input Validation
- Parameter Validation
- Session Management

Information Gathering

- Information gathered may allow an attack to infer additional information that can be used to compromise systems
- The best countermeasure is to be aware of the information you are making available and limit it to the minimum amount necessary.

Administrative Interfaces

- Interface for administrators to remotely carry out system maintenance
- The best countermeasure it to disable these interfaces
- However, the second best choice is to use an out of band connection
  - Such as a modem connected to the web server to dial in directly and configure it using a local interface

Authentication and Access Control

- Usually this is done with username and passwords, but usernames and passwords are commonly used across multiple sites
- The best defenses are:
  - Lock out users after multiple failed login attempts
  - Use log files to determine where attacks are originating from
  - Use SSL and HTTPS

Configuration Management

- Is simply the concept of managing the configuration of your systems
- This ensures that the default accounts and passwords are changed or disabled, error messages show the minimum amount of information necessary, and creates a security baseline for the company

Input Validation

- **Input validation is necessary to stop numerous attacks on the web, such as:**
  - Cross Site Scripting (XSS) Attacks
  - Path or directory traversal
  - Unicode encoding
  - URL encoding
  - SQL Injection
- **Best defenses are:**
  - Never trust information coming from the client without first validating it
  - Filter out all “known” malicious requests
  - Implement a strong policy to include appropiate parameter checking in all applications

Parameter Validation

- **Is where the values that are being received by the application are validated to be within defined limits before the server application processes them within the system**
  - **As opposed to input validation, parameter validation is more concerned with environment variables that do not provide the client with a way to modify in the application's interface (such as cookies or hidden values on a web page)**
- **The best defenses:**
  - **Pre-validation- input control verifying data is in appropriate format, and compliant with application specifications, prior to submission to the application**
  - **Post validation- ensuring an application's output is consistent with expectations**

Session Management

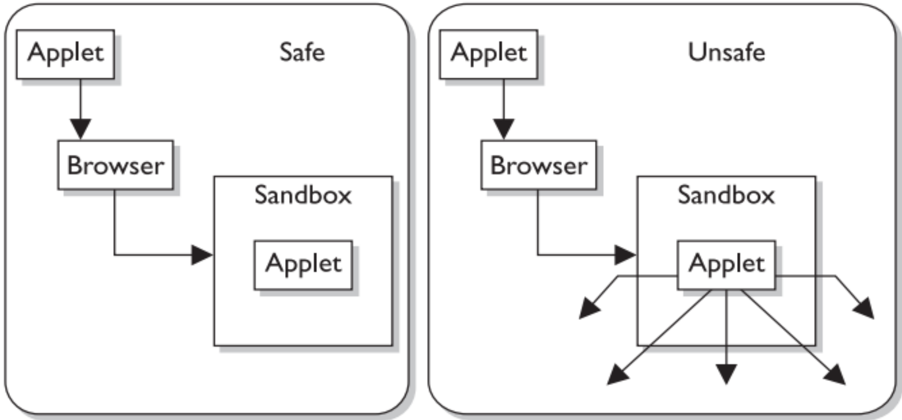
- The most common method for managing sessions is by assigning unique session IDs to every connection
- If attacker's can access or guess the session ID they can convince the server the attacker is the legitimate owner
- To prevent this session IDs should never be transmitted unencrypted and random session IDs should be used to avoid attackers guessing them
- Cookies used to keep state on the connections should also be encrypted

Mobile Code

- Code that can be transmitted across a network, to be executed by a system or a device on the other end
- This mobile code can be used to run malicious or comprising actions if a website is compromised
- On a web browser, the best defense is disallowing various scripting or active web components

Java Applets

- **Applets are small programs that run in a user's browser**
- **Java is platform independent because it creates intermediate code, byte code, which is not processor specific. This is then translated to machine code by the Java Virtual Machine (JVM)**
- **When an applet is executed it should run in a sandbox to limit its interactions with the host machine, however, bad guys have learned how to break out of the sandbox and write malicious applets**



ActiveX Controls

- Is a Microsoft technology composed of OOP technologies and tools based on COM and DCOM
- ActiveX practices security by informing users where the program comes from so that the user can decide to trust or not to trust it
- ActiveX technology also provides security levels and authentication, letting users control the security of the components they download
- Since they are downloaded to a user's hard drive they have much greater access to the user's system compared to Java Applets

Malicious Software (Malware)

- **Several types of malicious code or malware exist:**
  - **Viruses**
  - **Worms**
  - **Trojan Horses**
  - **Logic Bombs**

Malware

- These are the six main elements of malware, although it is not necessary for all six to be present:
  - Insertion- installing itself on the system
  - Avoidance- methods to avoid detection
  - Eradication- removes itself after the payload is executed
  - Replication- makes copies and spreads itself to other victims
  - Trigger- an event to initiate the payload
  - Payload- carries out the attack (deletes files, installs a backdoor, etc)

Viruses

- Is a small application, or string of code, that infects applications
- A virus’ main function is to reproduce, and it requires a host application to do this. If it does not have a host application then it cannot replicate.
- Macro virus- is a virus written in a macro language (Word Basic, Visual Basic, VBScript, etc) and is platform independent
  - They infect and replicate in templates and within documents
  - They are common since they are relatively easy to write and Office products are common

Viruses

- Boot sector viruses- are viruses that infect the boot sector or overwrite the sector with new information
  - Some have a portion of their code in the boot sector and hide the rest within the hard drive in areas marked off as bad sectors
- Compression viruses append themselves to executables on the system and compress themselves by using the user’s permissions
  - When the executable is run, the system automatically decompresses the virus and runs the malicious code

Viruses

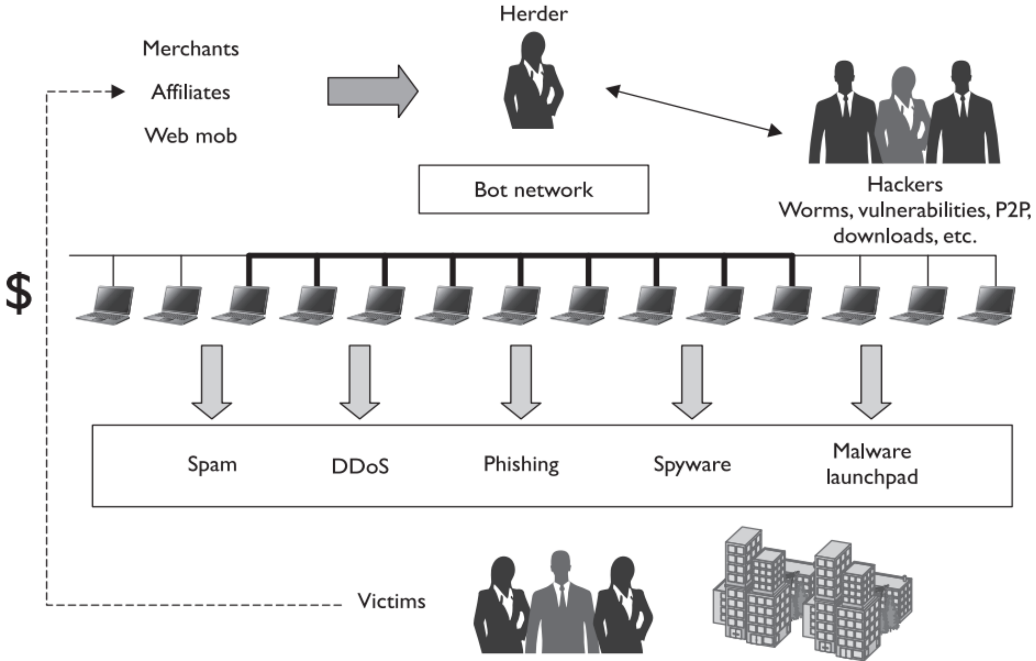
- Stealth virus hides the modifications it has made to files or boot records
  - This is usually accomplished by feeding monitoring functions false data
- Polymorphic virus produces varied but operational copies of itself
  - This is done in hopes of tricking the antivirus scans
  - Noise (bogus instructions) are often included to hide their signatures
- Multipart virus infects both the boot sector and executable files
- Self-garbling virus attempts to hide from antivirus software by garbling its own code
  - A small portion of the virus decodes the garbled code when activated

Viruses

- Meme virus are not actual computer viruses but types of e-mail messages that are continually forwarded around the internet (chain letters, e-mail hoax virus alerts, religious messages, pyramid selling schemes, etc)
  - These are replicated by humans not software
- Script virus carry out payloads written in a scripting language (Jscript, VBscript, etc) when viewed on a web page or e-mail
- Tunneling virus attempts to install itself under an antivirus program
  - When the AV sends out a command for information the virus intercepts the call and responds that the system is clean

Botnets

- Bots are a type of malware that carries out functionality for its master, who is the author of the code.
- Computers infected with dormant code are called *zombies*
- Botnets are commonly used to send spam, viruses, pornography, or attack code to help ensure the attacker's IP is not found



Worms

- **Worms are different from viruses in that they can reproduce on their own without a host application and are self contained programs**
  - **The definition between worms and viruses are continually merging and becoming blurred**

Logic Bombs and Trojan Horses

- Logic bombs execute a string of code or program when a certain event happens or a date and time arrive
- Trojan horses are programs disguised as another program
  - They generally perform a useful function in addition to their malicious purpose
- Remote Access Trojans (RATs) are malicious programs that run on systems and allow intruders to access and use a system remotely

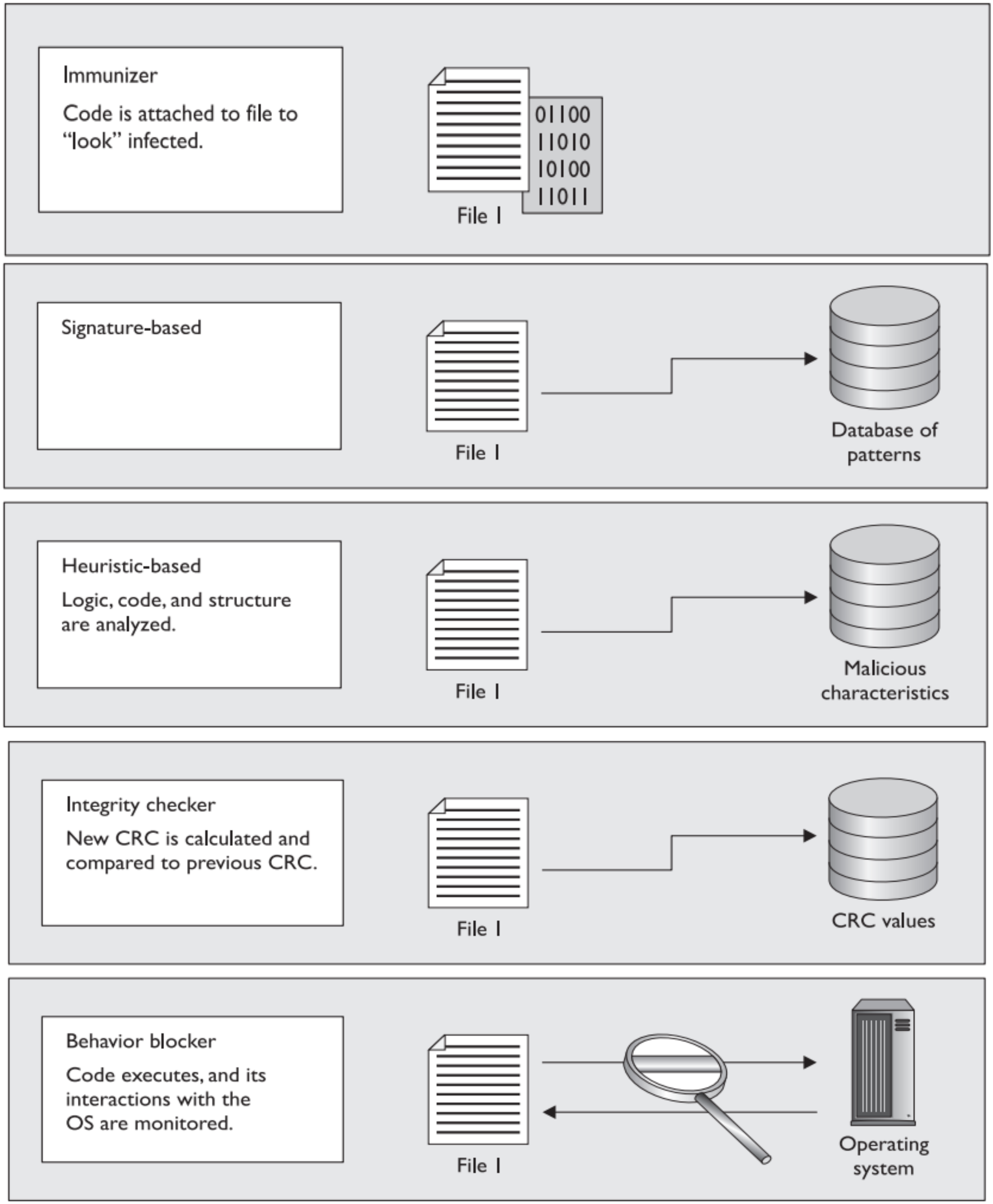
Antivirus Software

- Signature based detection (also called fingerprint detection) uses signatures to detect malicious code
  - Signature is a sequence of code that was extracted from the virus itself
  - With this sort of antivirus there is usually a delayed response time as the virus is studied
- Heuristic detection analyzes the whole structure of the malicious code, evaluates the coded instructions and logic functions, and looks at the type of data within the virus or worm
- Immunizer attaches code to the file or application, which would fool a virus into “thinking” it was already infected

Antivirus Software

- Dynamic analysis is allowing a piece of the suspicious code to run in a virtual machine
- Static analysis is only reviewing information about a piece of code
- Behavior Blocking allows the suspicious code to execute within the OS unprotected and watches for suspicious interactions with the OS
  - If any suspicious activities are found the user is notified and the process is terminated
  - Some newer anti-viruses offer the ability to roll back to a state before the infection took place
- Heuristic detection and behavior blocking are both effective against "zero day" attacks, signature based detection cannot detect new malware

Antivirus Detection Methods



Immunizer

Code is attached to file to "look" infected.

Signature-based

Heuristic-based

Logic, code, and structure are analyzed.

Integrity checker

New CRC is calculated and compared to previous CRC.

Behavior blocker

Code executes, and its interactions with the OS are monitored.

Spam Detection

- Spam unsolicited junk e-mail
- Bayesian filtering- carries out a frequency analysis on each word and then evaluates the message as a whole to determine whether or not it is spam

Anti-Malware Programs

- A company should have a policy on what anti-virus and anti-spyware software should be installed and configured
- Users should know who to contact if a virus is discovered
- Scanning software can also be integrated at network entry points
  - Mail server, proxy server, firewall, etc

Patch Management Process

- **It is important to have a process to implement patches to software since they often times will remove or change functions**

Steps of patch management process:

- **Infrastructure- physical media and the process that supports the patch**
- **Research- making sure the files are free from corruption and aren’t trojan horses**
- **Assess and Test- test for any unexpected effects a patch may have**
- **Mitigation (“Rollback”)- the ability to roll back any changes from unforeseen consequences**
- **Deployment (“Rollout”)- start with a pilot group and gradually rollout to the rest of the company**
- **Validation, Reporting, and Logging- auditing of what, where, when, and how the patches were deployed**

Patching

- Patch management can be restricted by failures in the systems or infrastructure involved in the distribution of patches, or from failures in the patch method itself
- It is good practice to keep an up-to-date inventory of all the software, hardware, and configuration info

Denial of Service (DoS)

- These attacks can be performed by sending malformed packets to a system that does not recognize the format and thus does not know how to process it
- DoS attacks can interrupt service or completely deny legitimate users access
- Normally DoS attacks target bandwidth, but they can also use up other resources such as processes, file system quotas, memory, or CPU utilization

Smurf

- This attack requires three players: the attacker, the victim, and the amplifying network
- The attacker spoofs to make an ICMP ECHO REQUEST packet seem to originate from the victim and then broadcasts it to the network
- The victim computer is overwhelmed as each computer replies
- Countermeasures:
  - Disable direct broadcast functionality
  - Configure perimeter routers to reject as incoming messages any packets that contain internal source IP addresses
  - Allow only the necessary ICMP traffic into and out of an environment

Fraggle

- Is similar to smurf, but instead of using ICMP it employs the User Datagram Protocol (UDP)
- The attacker broadcasts a spoofed UDP packet to the amplifying network, which in turn replies to the victim's system
- Countermeasures:
  - Disable direct broadcast functionality at perimeter routers to make sure a certain network is not used as an amplifying stage
  - Packets that contain internal source IP addresses should not be accepted
  - Allow only the necessary UDP packets into and out of the environment

SYN Flood

- A SYN flood is a form of denial-of-service attack in which an attacker sends a succession of SYN requests to a target's system.
- The victim will commit the necessary resources to set up this communications socket, and it will send its SYN/ACK message, waiting for the ACK messages in return.
- Countermeasures:
  - Decrease the connection-established timeout period (This only lessens the effects of a SYN attack)
  - Increase the size of the connection queue in the IP stack
  - Install vendor-specific patches, where available, to deal with SYN attacks

Teardrop

- Is an attack where a packet cannot be correctly reformed and causes the system to freeze or reboot
- Countermeasures:
  - Install the necessary patch or upgrade the operating system
  - Disallow malformed fragments of packets to enter the environment
  - Uses a router that combines all fragments into a full packet prior to routing it to the destination system

Distributed Denial of Service(DDoS)

- Is the logical extension of the DoS attack that gets more computers involved in the act
- The DDoS attack uses hundreds or thousands of computers to request services from a server or server farm until the system or web site is no longer functional
- Countermeasures:
  - Use perimeter routers to restrict unnecessary ICMP and UDP traffic
  - Employ a network-based IDS to watch for this type of suspicious activity
  - Disable unused subsystems and services on computers

Development environment security controls

Reference: Dr. Devin Cook, Sandia Labs

Reference: Shon Harris

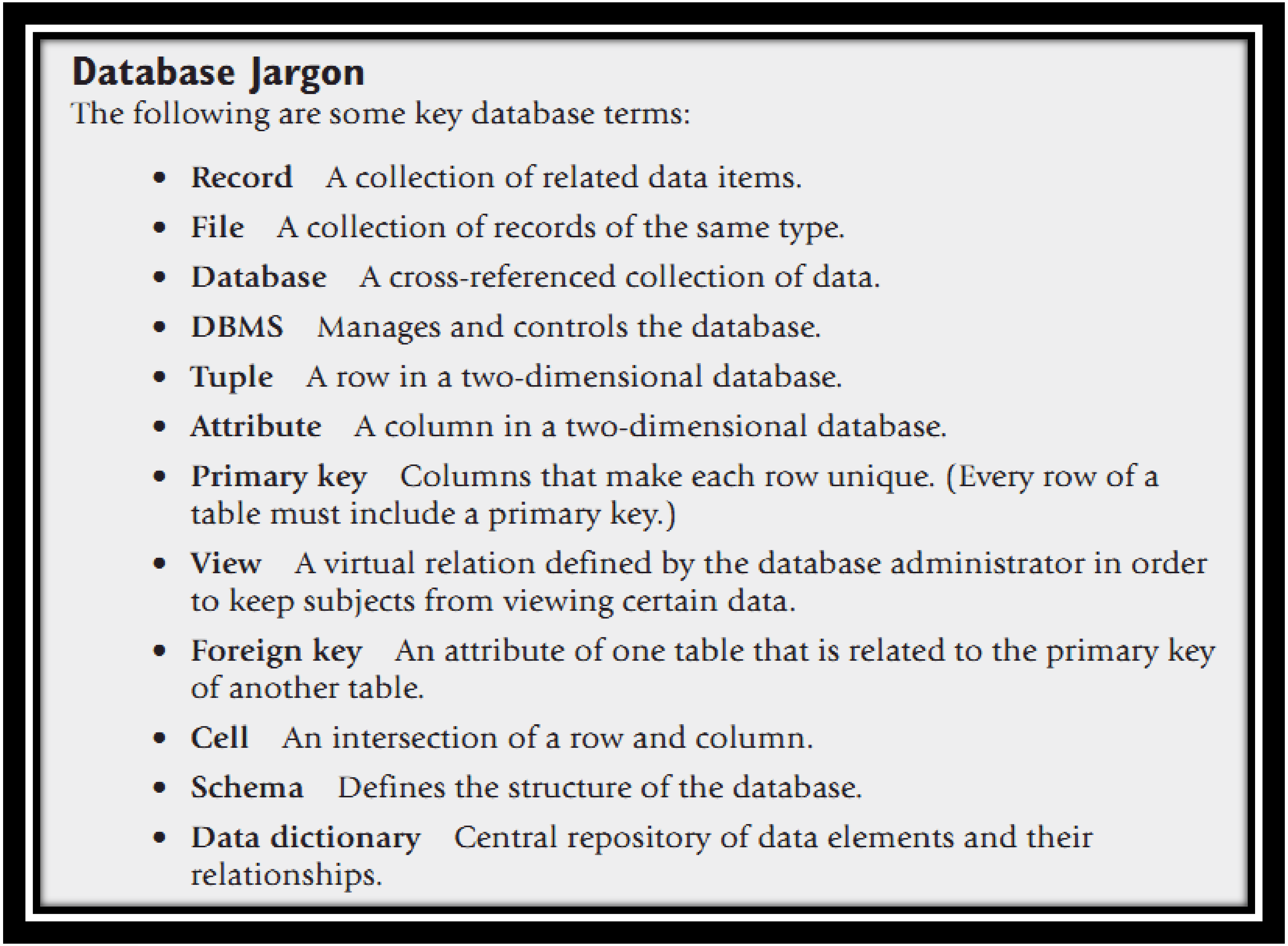
Security Controls for Software Environment

- For CISSP Exam, countermeasures are also called “security controls”…
  - Security Controls for Buffer Overflows
  - Memory Protection
  - Covert Channel Controls
  - Cryptography
  - Password Protection Techniques
  - Inadequate Granularity of Controls
  - Control and Separation of Environments
  - Time of Check/Time of Use (TOC/TOU)
  - Social Engineering
  - Backup Controls
  - Malicious Code/Malware Controls
  - Virus Protection Controls
  - Mobile Code Controls
  - Sandbox
  - Programming Language Support
  - Access Controls

Reference: *Official (ISC)^2 Guide to The CISSP CBK*, H. Tipton, et. al., (ISC)^2 Press, Auerbach Publications, 2007.

Database Management Definitions

- **Database- a collection of data stored in a meaningful way that enables users and applications to access, view and modify data as needed**
  - **A cross-referenced collection of data**
- **Database Management System (DBMS)- the software that controls the access restrictions, data integrity, redundancy, and the different types of manipulation available for a database.**
  - **Manages and controls the database**
- **SQL- Structured Query Language**



Database Jargon

The following are some key database terms:

- **Record** A collection of related data items.
- **File** A collection of records of the same type.
- **Database** A cross-referenced collection of data.
- **DBMS** Manages and controls the database.
- **Tuple** A row in a two-dimensional database.
- **Attribute** A column in a two-dimensional database.
- **Primary key** Columns that make each row unique. (Every row of a table must include a primary key.)
- **View** A virtual relation defined by the database administrator in order to keep subjects from viewing certain data.
- **Foreign key** An attribute of one table that is related to the primary key of another table.
- **Cell** An intersection of a row and column.
- **Schema** Defines the structure of the database.
- **Data dictionary** Central repository of data elements and their relationships.

Types of Databases

- Relational
- Hierarchical
- Network
- Object-oriented
- Object-relational

Relational Database Model

- Uses attributes (columns) and tuples (rows) to contain and organize information
- Uses tables to present information
- Uses primary keys to link data to a unique value

Hierarchical Data Model

- Combines records and fields that are related in a logical tree structure
- Parents can have one child, many children, or no children.
- The tree contains branches, and each branch has a number of leaves (or data fields).
- Useful for mapping one-to-one relationships

Network Database Model

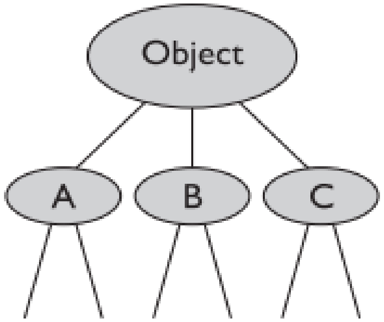
- Built upon a hierarchical data model
- Each element can have multiple parents and child records
- This forms a redundant network-like structure instead of a strict-tree structure
- Uses records and sets
  - Records contain fields, which may lay out in a hierarchical structure
  - Sets define the one-to-many relationships between the records

Object-Oriented Database

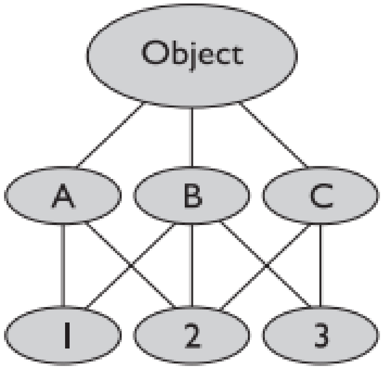
- Designed to handle a variety of data (images, video, etc)
- Has classes that define attributes and procedures of its objects
- An object-oriented database management system (ODBMS) is more dynamic in nature than a relational database because objects can be created when needed and the data *and* procedure go with the object when it is requested

Object-relational Database (ORD)

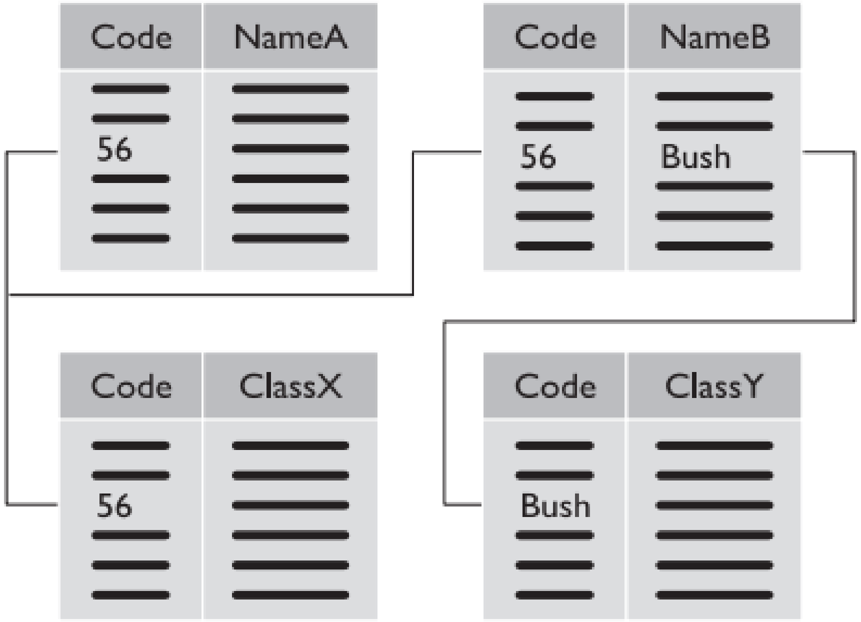
- ORD (AKA object-relational database management system [ORDBMS]) is a relational database with a software front end that is written in an object oriented programming language.
- A benefit of ORD is that it contains the necessary procedures to get what is needed by an application from the database.



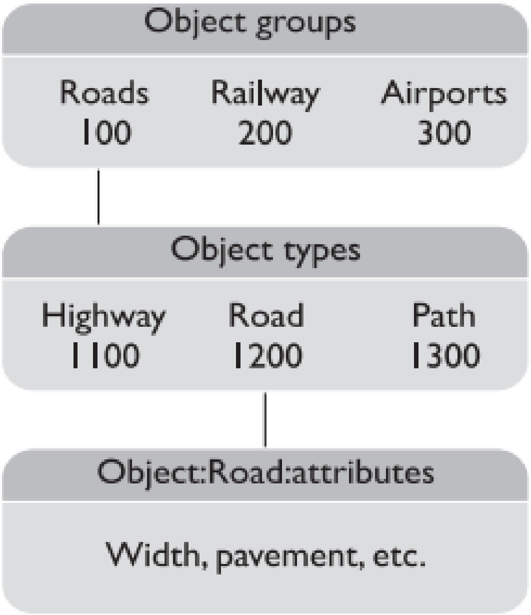
Hierarchical model



Network model



Relational model



Object-oriented model

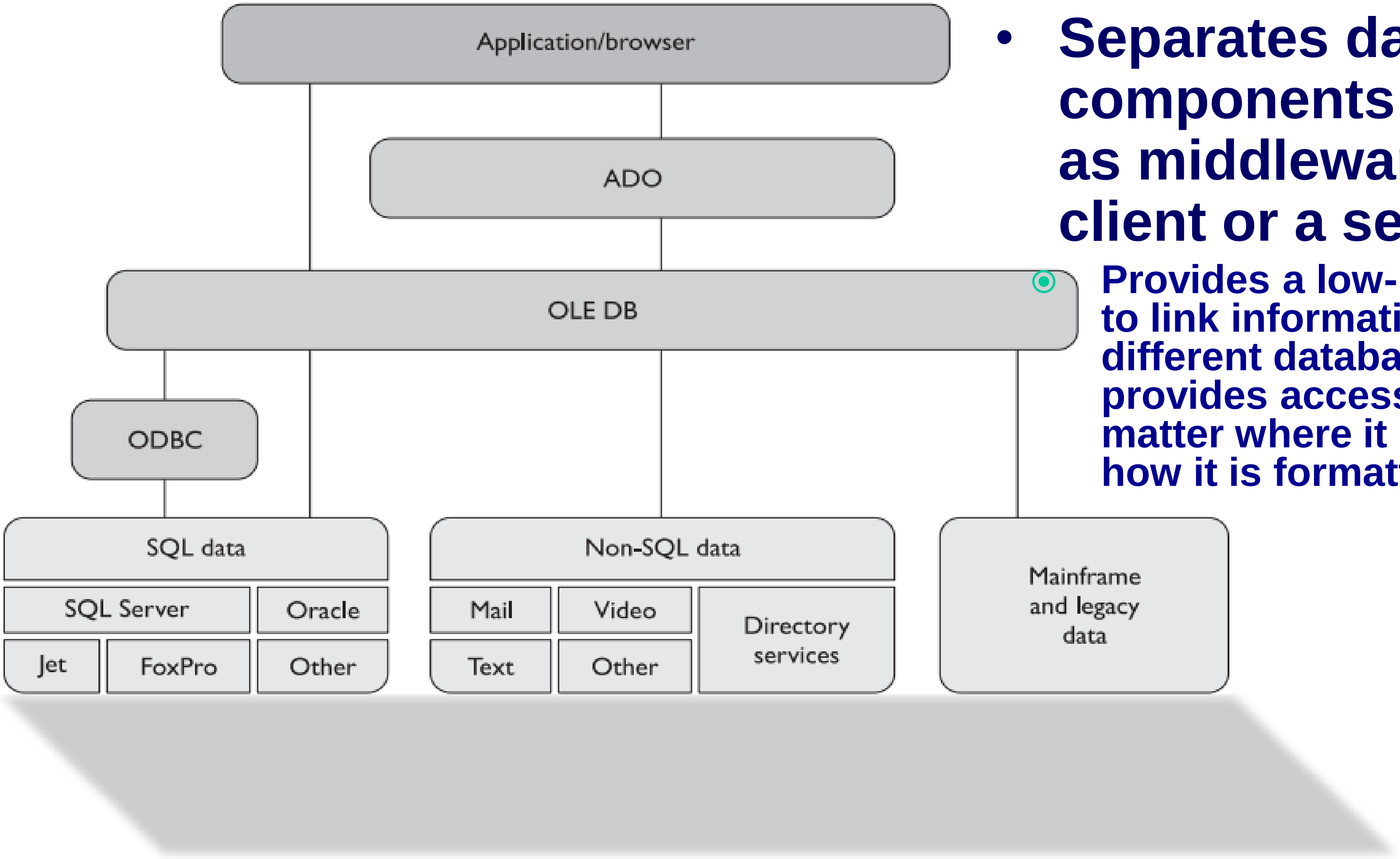
Database Programming Interfaces

- Open Database Connectivity (ODBC)
- Object Linking and Embedding Database (OLE DB)
- ActiveX Data Objects (ADO)
- Java Database Connectivity (JDBC)

Open Database Connectivity (ODBC)

- An application programming interface (API) that allows an application to communicate with a database either locally or remotely.
- ODBC tracks down the necessary database specific driver for the application and translates it into a database command as needed.

Object Linking and Embedding Database (OLE DB)



- **Separates data into components that run as middleware on a client or a server.**
  - **Provides a low-level interface to link information across different databases and provides access to data no matter where it is located or how it is formatted.**

OLE DB Characteristics:

- It’s a replacement for ODBC
  - Extends support to a wider variety of non-relational databases, such as object databases and spreadsheets that do not implement SQL
- COM-Based
  - Provides applications with uniform access to data stored in diverse data sources
  - Due to this is limited to use by Microsoft Windows based client tools.
- Developers access OLE DB services through ActiveX Data Objects (ADO)
- Allows different applications to access different types and sources of data

ActiveX Data Objects (ADO)

- An API that allows applications to access back-end database systems.
- ADO uses the OLE DB interface to connect with the database and can be developed with many different scripting languages.

ADO Characteristics:

- High-level data access programming interface to an underlying data access technology (such as OLE DB)
- It’s a set of COM objects for accessing data sources, not just database access
- Allows a developer to write programs that access data, without knowing how the database is implemented
- SQL commands are not required to access a database when using ADO.

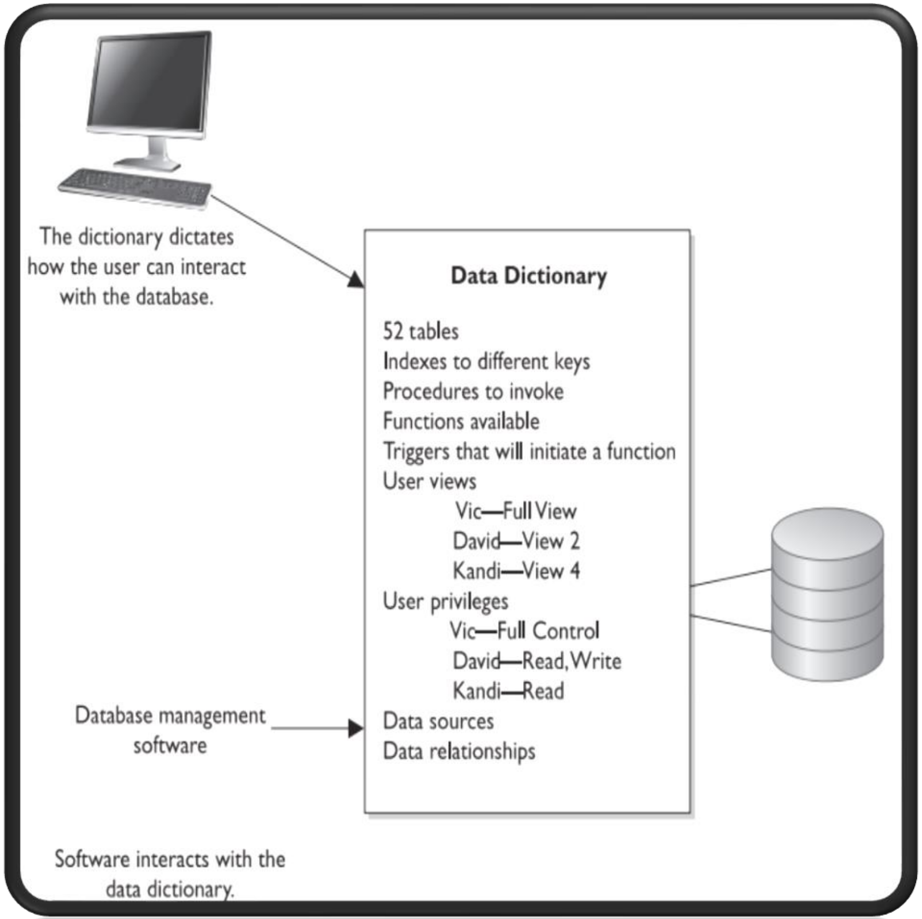
Java Database Connectivity (JDBC)

- An API that allows a Java application to communicate with a database.
- The application can use it to bridge through ODBC or directly to the database.
- Characteristics:
  - Same functionality as ODBC, but specifically designed for use by Java database applications
  - Has database-independent connectivity between Java platform and wide range of databases
  - Enables Java programs to execute SQL statements

Components of a Relational Database

- Data Definition Language (DDL)- defines the structure and schema of the database
  - Structure could mean table size, key placement, views, and data element relationships.
- Data Manipulation Language (DML)- contains all the commands that enable a user to view, manipulate, and use the database
  - Such as view, add, modify, delete, etc
- Query Language (QL)- enables users to make requests of the database
- Report Generator- produces printouts of data in a user-defined manner

Data Dictionary



The dictionary dictates how the user can interact with the database.

Data Dictionary

52 tables
Indexes to different keys
Procedures to invoke
Functions available
Triggers that will initiate a function
User views
Vic—Full View
David—View 2
Kandi—View 4
User privileges
Vic—Full Control
David—Read, Write
Kandi—Read
Data sources
Data relationships

Database management software

Software interacts with the data dictionary.

- This is a central collection of data element definitions, schema objects, and reference keys.
  - Schema Objects can be tables, views, indexes, procedures, functions, and triggers
- It is a tool used to centrally manage parts of a database by controlling data about data (referred to as *metadata*) within the database
- The DBMS creates and reads the data dictionary to determine if schema objects exist and if users have proper access rights to view them

Primary Key vs. Foreign Key

- **The primary key is an identifier of a row and is used for indexing in relational databases.**
  - **Primary keys must be unique to properly represent a row as one entity.**

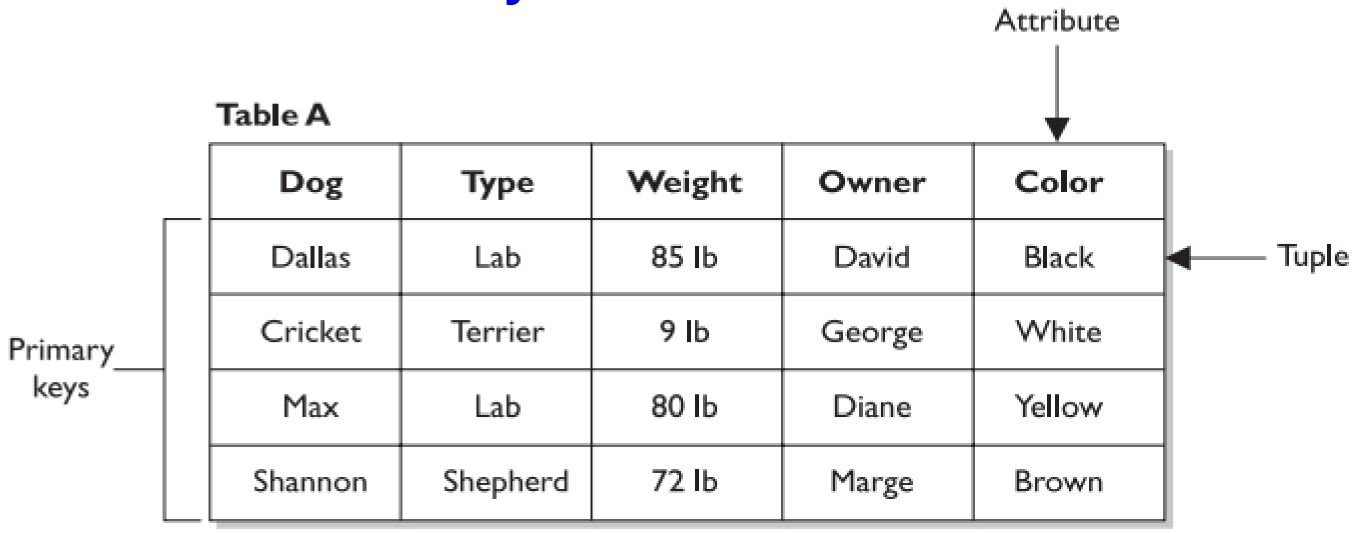


Table A

Dog	Type	Weight	Owner	Color
Dallas	Lab	85 lb	David	Black
Cricket	Terrier	9 lb	George	White
Max	Lab	80 lb	Diane	Yellow
Shannon	Shepherd	72 lb	Marge	Brown

Attribute
Tuple
Primary keys

Primary Key vs. Foreign Key

- **A foreign key is an attribute in one table has a value matching the primary key of another table and there is a relationship set up between the two of them.**
  - **The foreign key does not necessarily have to be the primary key of its table.**

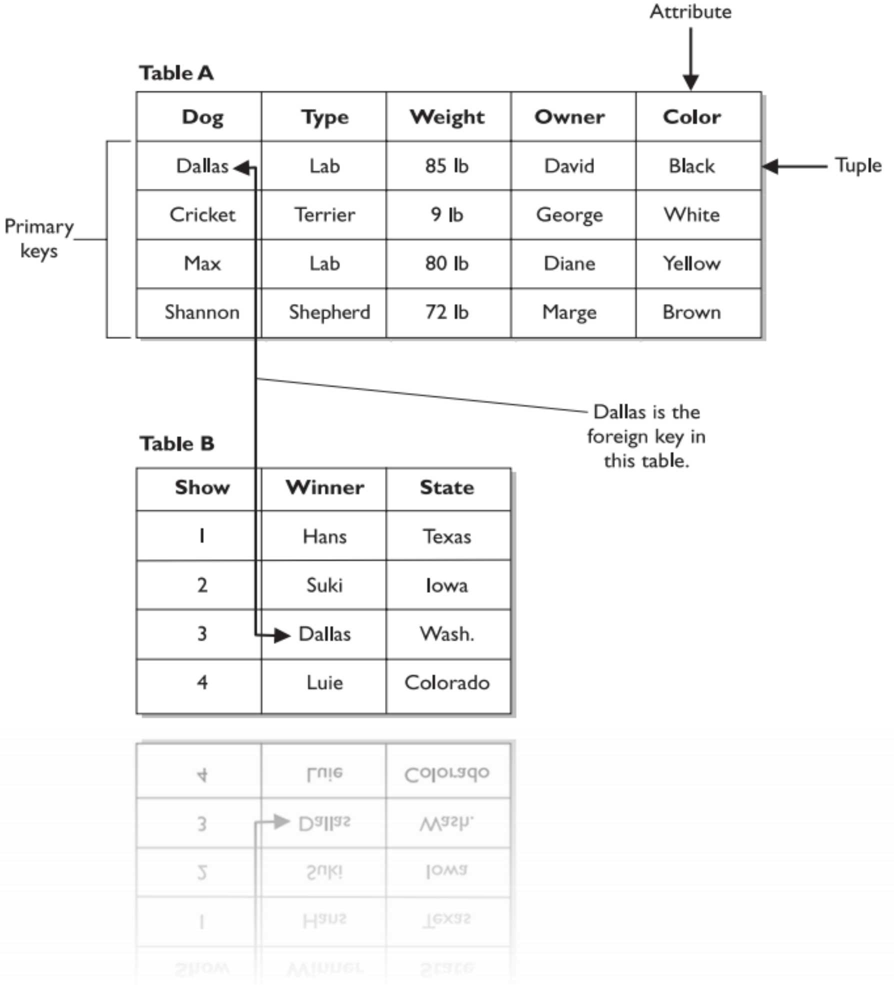


Table A

Dog	Type	Weight	Owner	Color
Dallas	Lab	85 lb	David	Black
Cricket	Terrier	9 lb	George	White
Max	Lab	80 lb	Diane	Yellow
Shannon	Shepherd	72 lb	Marge	Brown

Table B

Show	Winner	State
1	Hans	Texas
2	Suki	Iowa
3	Dallas	Wash.
4	Luie	Colorado

Integrity

- Concurrency issues come up when there is a piece of software that will be accessed at the same time by different users and/or applications.
- One method to avoid concurrency issues is locking, which prevents users from accessing and modifying data being used by someone else.
- In database software there are three main types of integrity services:
  - Semantic Integrity
  - Referential Integrity
  - Entity Integrity

Integrity

- **Semantic integrity makes sure structural and semantic rules are enforced.**
- **These rules pertain to data types, logical values, uniqueness constraints, and operations that could adversely affect the structure of the database.**

Integrity

- A database has referential integrity if all foreign keys reference existing primary keys.
- Entity integrity guarantees that the tuples are uniquely identified by primary key values.
  - For the sake of entity integrity, every tuple must contain one primary key.
  - Without a primary key, the tuple cannot be referenced by the database.
- A database must not contain unmatched foreign key values, all must be matched to an existing primary key

Operations to keep integrity

- A rollback is an operation that ends the current transaction and cancels the current changes to the database.
- A commit operation completes a transaction and executes all changes just made by the user.
  - If a commit cannot be completed a rollback is performed so that no partial changes are executed
- Save points are used to make sure that if a system failure occurs, or if an error is detected, the database can attempt to return to a point before the system crashed or malfunctioned.

Operations to keep integrity

- Checkpoints are very similar to savepoints. When the database software fills up a certain amount of memory a checkpoint is created, which writes the data to a temporary file.
- Two-phase commit means when a change is submitted by a user, the different databases initially store these changes temporarily. It asks for a “pre-commit” from all the involved databases, and if it gets the affirmative then the monitor sends out a commit command to each database.

Database Security Issues

- **Aggregation is the act of combining information from separate sources. The combination of the data forms new information which the user does not have the necessary rights to access.**
  - **The combined information has a sensitivity greater than that of the individual parts.**

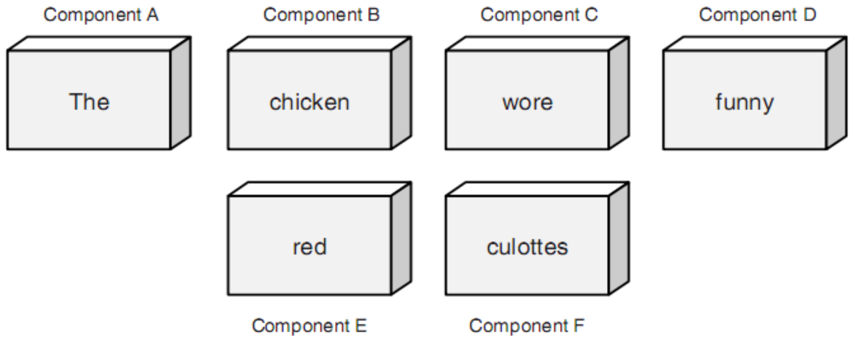


Figure 11-9 Because Emily has access to components A, C, and F, she can figure out the secret sentence through aggregation.

Database Security Issues

- Inference is the intended result of aggregation.
  - This is when the user deduces the fully story from the pieces learned of through aggregation.
  - This is indicative of data at lower security indirectly portraying data at a higher level.
- Content-dependent access control- is based on the sensitivity of the data.
  - The more sensitive, the less people who can access it
- Context-dependent access control- the software determines what should be allowed based upon the state and sequence of the requests.

Database Security Issues

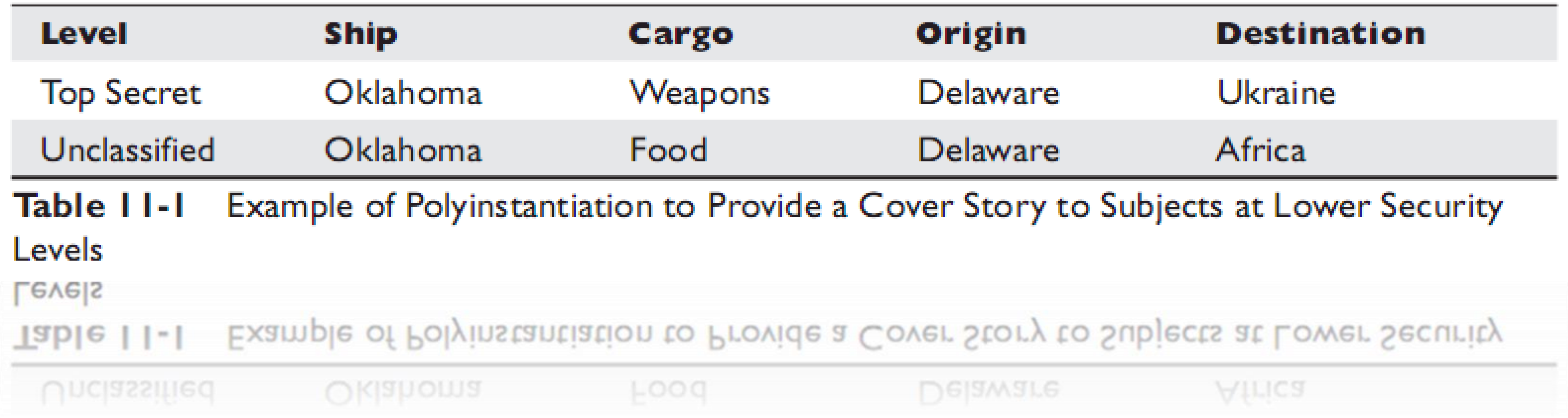
- Common attempts to prevent inference attacks are cell suppression, partitioning the database, and noise and pertubation.
- Cell suppression- a technique used to hide specific cells that contain information that could be used in inference attacks.
- Partitioning- involves dividing the database into different parts so that is harder to find the connecting pieces of information.
- Noise and perturbation- a technique for inserting bogus info in the hopes of misdirecting an attacker or confusing the matter enough that the inference attack is not fruitful.

Database Views

- **Database Views- permitting one group, or a specific user, to see certain information while restricting another group from viewing it altogether.**
  - **This can be done with discretionary access controls (DAC) or mandatory access controls (MAC) as discussed in chapter 4.**
    - **DAC is based on identity, authentication and authorization**
    - **MAC then user security clearances and data classificiation levels would be used**

Polyinstantiation

- **Polyinstantiation enables a table to contain multiple tuples with the same primary keys with each instance distinguished by security levels.**
  - **This avoids telling lower-level users that something of higher clearance is available**



Level	Ship	Cargo	Origin	Destination
Top Secret	Oklahoma	Weapons	Delaware	Ukraine
Unclassified	Oklahoma	Food	Delaware	Africa

Table 11-1 Example of Polyinstantiation to Provide a Cover Story to Subjects at Lower Security Levels

Online Transaction Processing (OLTP)

- **OLTP provides mechanisms that watch for problems and deal with them appropriately when they do occur.**
- **It is generally used when databases are clustered to provide fault tolerance and higher performance.**
- **OLTP's main goal is to ensure that transactions happen properly or they don't happen at all.**

ACID Test

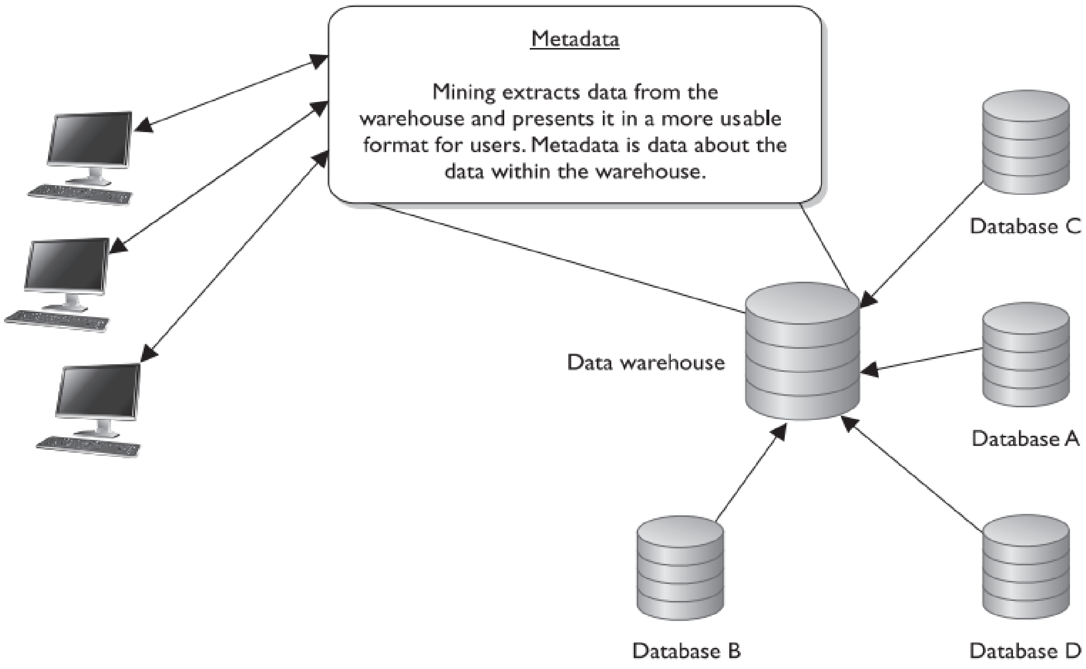
- Since OLTP records transactions as they occur, it usually updates more than one database in a distributed environment. This type of complexity can introduce many integrity threats so the database software should implement the characteristics of the ACID test.

ACID Test

- **Atomicity- divides transactions into units of work and ensures that all modifications take effect or none take effect.**
- **Consistency- a transaction must follow the integrity policy developed for the particular database and ensure all data are consistent in the different databases.**
- **Isolation- transactions execute in isolation until completed, without interacting with other transactions. Modification results are not available until the transaction is complete.**
- **Durability- Once verified on all systems, the changes are committed and cannot be rolled back**

Data Warehousing and Data Mining

- **Data warehousing- combines data from multiple databases into a large database for the purpose of providing more extensive information retrieval and data analysis.**
- **Data mining- the process of massaging data held within a data warehouse to provide more useful information to users.**
- **Metadata- data about data in the warehouse.**


Metadata
Mining extracts data from the warehouse and presents it in a more usable format for users. Metadata is data about the data within the warehouse.
Data warehouse
Database C
Database A
Database B
Database D

Knowledge Discovery in Database (KDD)

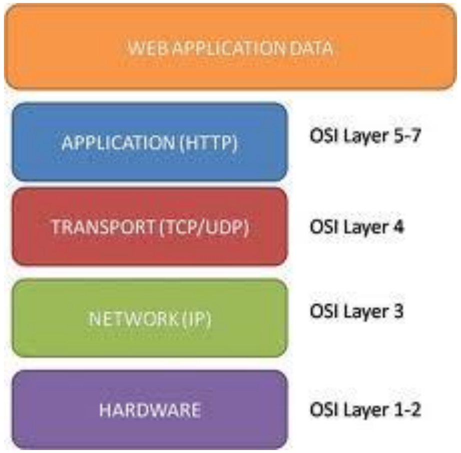
- Knowledge Discovery in Database (KDD)- a combination of techniques to identify valid and useful patterns
- These approaches are used:
  - Classification- groups together data according to shared similarities
  - Probalistic- identifies data interdependencies and applies probabilities to their relationship
  - Statistical- identifies relationships between data elements and uses rule discovery

Software security effectiveness

Reference: Joe Krull

Application Security Fundamentals

- Application security includes measures taken throughout an application's life-cycle to prevent exceptions in the security policy of an application or the underlying system (vulnerabilities) through flaws in the design, development, deployment, upgrade, or maintenance of the application.*
- The primary focus is on Layer 7 of the OSI Model
- AppSec should be part of an organization's or vendor's Software (or System) Development Life-Cycle (SDLC)
- A key component of application security should be for developers and their managers to be aware of basic AppSec requirements, common threats and effective countermeasures
- AppSec knowledge and maturity is significantly lower today than traditional network security

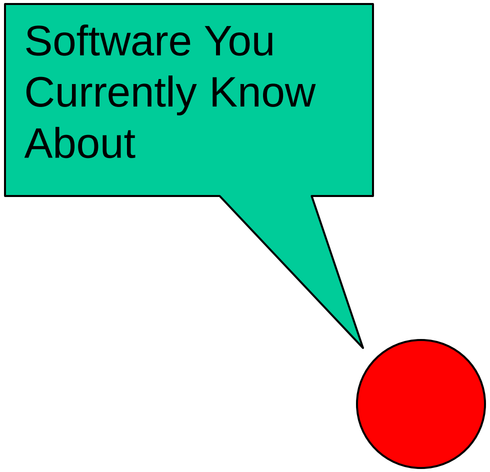


Risks Associated With Vulnerable Applications

- Unauthorized access to sensitive customer or company data
- Theft of sensitive data to conduct identity theft, credit card fraud or other crimes
- Defacement of websites; strong potential for brand damage
- Manipulation of data impacting data integrity, quality and organization’s reputation
- Redirection of users to malicious web sites; phishing and malware distribution
- Denial of service; availability of data
- Attackers can assume valid user identities
- Access to hidden web pages using forged URLs
- Attacker’s hostile data can trick the interpreter to execute unintended commands

What Is Your Software Attack Surface?

To assess application security, many organizations focus on obvious software resources, but overlook their overall inventory of applications and code from less obvious sources when they analyze their assets.



Software You Currently Know About

What's Normally In This Category?

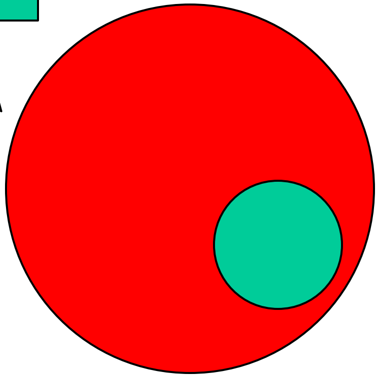
- Critical legacy systems
- Notable web applications

Why Do These Usually Merit Consideration?

- Lots of monetary or brand value flows through them
- Compliance requirements (e.g. PCI, HIPAA, FFIEC, etc.)
- Formal SLAs with customers
- You've had one or more previous security incidents (or near misses)

What Is Your Software Attack Surface – Part 2?

Add In the Rest of the Web Applications Your Organization Actually Develops and Maintains



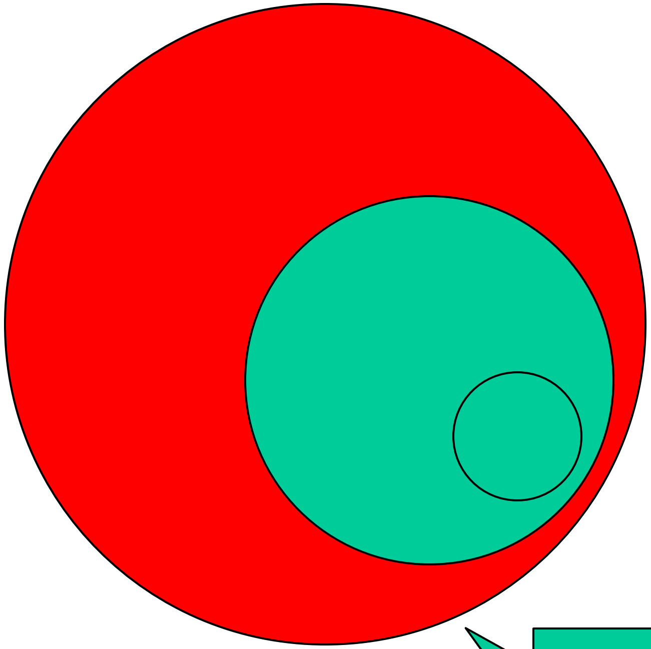
What's Normally in This Category?

- Line of business applications
- Event-specific applications, e.g. holiday apps, sales support, open enrollments

Why Could You Miss Them in Your Analysis ?

- Lack of knowledge, overlooked or forgot they were there
- Line of business procured through non-standard channels
- Added through a merger or acquisition
- Believed to be retired but still active

What Is Your Software Attack Surface – Part 3?



Add In the Software You Bought from Somewhere

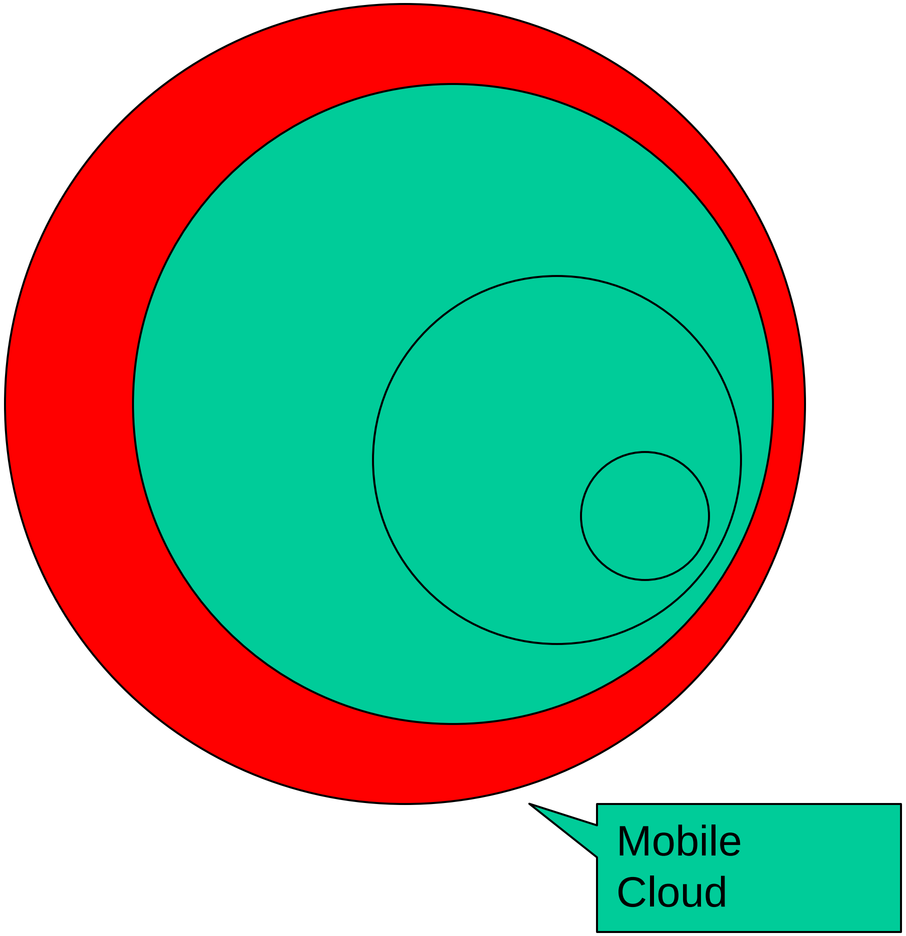
What's Normally in This Category?

- Less known or utilized line of business applications
- Support applications
- Infrastructure applications

Why Could You Miss Them in Your Analysis?

- Automated scanners are good at finding web applications. Non-web, not so much.
- Contract language or un-validated assumptions that the application vendor has security "covered"

What Is Your Software Attack Surface – Part 4?



What's Normally in This Category?

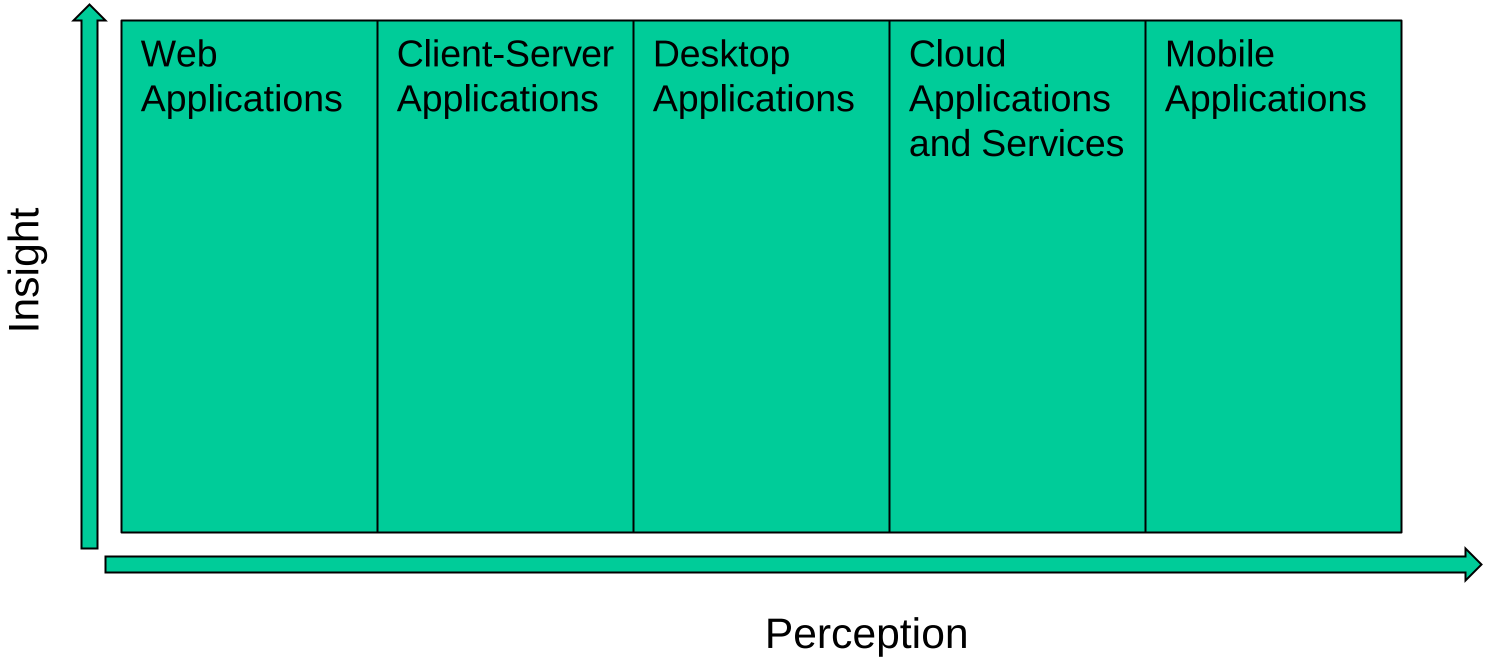
- Support for line of business functions
- General marketing and promotion
- Financial analysis applications
- Software as a Service (SaaS)
- Mobile applications
- User procured software

Why Could You Miss Them in Your Analysis?

- Decentralized procurement
- Ineffective security policies
- Use of prohibited software
- Lack of awareness

Attack Surface: The Security Officer's and Auditor's Perspective

As perception of the problem of attack surface grows, the scope of the problem increases – or, the more you know, the more you need to assess

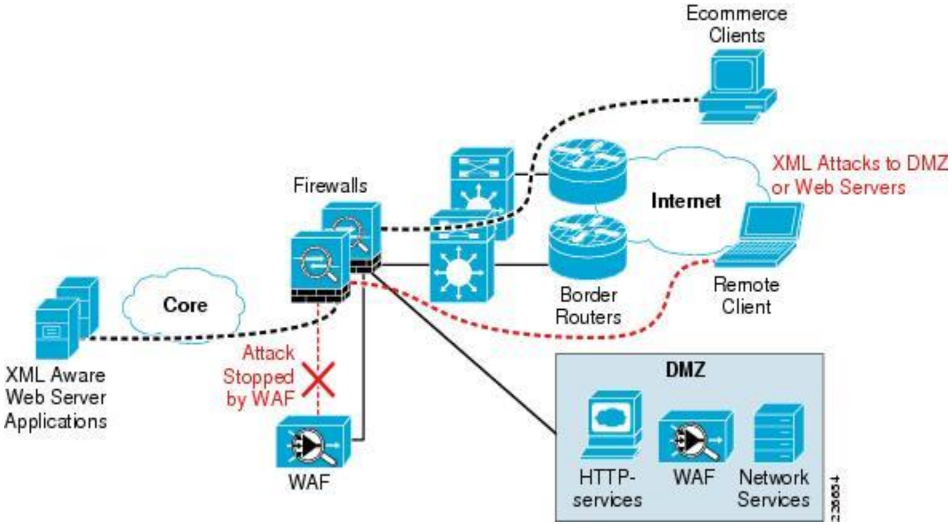


Value and Risk Are Not Equally Distributed

- **Some Applications Matter More Than Others**
  - **Value and character of data being managed**
  - **Value of the transactions being processed**
  - **Cost of downtime and breaches**

- **Therefore All Applications Should Not Be Treated the Same**
  - **Allocate different levels of resources to assurance**
  - **Select different assurance activities**
  - **Also must often address compliance and regulatory requirements**

Myth #1 – I Don't Need AppSec Because My Network is Secure



Technical Rationale

Non-Technical Rationale

Mean Time to Fix (MTTF)

- A 2013 industry study from White Hat Security revealed that the "Mean Time to Fix" for web application flaws categorized as "serious" averaged 193 days across all industries.
- In the same study, for one industry (Education) the figure jumped to 342 days of exposure
- In a similar study from Veracode, 70% of 22,430 applications submitted to their testing platform in 2012 contained exploitable security vulnerabilities
- How would you report to your management that a "serious" and likely exploitable vulnerability was present on your primary public facing web site or a 3rd party hosted portal for more than six months?
- What compensating control or controls do you think you could explain to placate management that a serious vulnerability could not be exploited?
- Verizon's 2013 Breach Report says 90% of attacks last year were perpetrated by outsiders and 52% used some form of hacking. How does this help you explain application risk?

Myth #2 – An Automated Scanner Can Find All The Application Vulnerabilities That Exist

- There is no “silver bullet” for identifying application security vulnerabilities. There are different classes of tools ranging from static code scanners that assess the code to dynamic scanners that analyze logic and data flow. Generally, 30% to 40% of vulnerabilities can be identified by scanners; the remainder are uncovered by other means.
- Manual testing allows an informed and experienced tester to attempt to manipulate the application, escalate privileges or get the application to operate in a way it was not designed to do.

What Goes Into An Application Test?

Application security goes well beyond simply running a scanning tool. For critical or high value applications, or those that process sensitive data, thorough testing may actually include a combination of several methods.

- **Unauthenticated Automated Scan**
- **Authenticated Automated Scan**
- **Automated Binary Analysis**
- **Blind Penetration Testing**
- **Manual Source Code Review**
- **Manual Binary Analysis**
- **Informed Manual Testing**
- **Automated Source Code Scanning**

AppSec – What Can You Do and Why?

Information Security Professionals

- Promote AppSec awareness in your organization
- Confirm that application security testing is part of your overall security program
- Demand that all applications developed by 3rd parties be tested and remediated prior to being placed in production
- Get all developers and their managers trained on AppSec
- Obtain and review the SDLC from a security perspective

IT Auditors

- Influence your Chief Audit Executive to include AppSec in the organization's annual risk assessment
- Increase your relevance and value to your organization by identifying risks associated with poorly coded applications
- Conduct a simple initial audit to assess what controls are in place
- Conduct a subsequent audit to determine the effectiveness of those controls; measure MTTF

Tools and Resources

- Open Software Assurance Maturity Model (OpenSAMM)
  - A freely available open source framework that organizations can use to build and assess their software security programs
    - www.opensamm.org

- The Open Web Application Security Project (OWASP)
  - Worldwide not-for-profit organization focused on improving the security of software. Source of valuable free resources
    - www.owasp.org
- Open Source or Low Cost Application Security Scanners
  - OWASP Zed Attack Proxy (ZAP), w3af, Mavituna Netsparker, Websecurify, Wapiti, N- Stalker, SkipFish, Scrawlr, Acunetix, and many more to do basic discovery work

The OWASP Top 10

- **A1 Injection**
- **A2 Broken Authentication and Session Management**
- **A3 Cross-Site Scripting (XSS)**
- **A4 Insecure Direct Object References**
- **A5 Security Misconfiguration**
- **A6 Sensitive Data Exposure**
- **A7 Missing Function Level Access Control**
- **A8 Cross-Site Request Forgery (CSRF)**
- **A9 Using Components with Known Vulnerabilities**
- **A10 Unvalidated Redirects and Forwards**

Example AppSec Audit Work Program

Software Assurance Maturity Model (SAMM) Scorecard

				Level 1	
				Activity	
Business Functions	#	Security Practices/Phase	Maturity Level	A	B
Governance	1	Strategy & Metrics	0.5	🟥	🟩
	2	Policy & Compliance	0.5	🟥	🟩
	3	Education & Guidance	0	🟥	🟥
Construction	4	Threat Assessment	0	🟥	🟥
	5	Security Requirements	0.5	🟥	🟩
	6	Secure Architecture	0	🟥	🟥
Verification	7	Design Review	0.5	🟥	🟩
	8	Code Review	0	🟥	🟥
	9	Security Testing	0	🟥	🟥
Deployment	10	Vulnerability Management	1	🟩	🟩
	11	Environment Hardening	1	🟩	🟩
	12	Operational Enablement	0	🟥	🟥

Example AppSec
udit Work Progran

SAMM Valid Maturity Levels	
0	Implicit starting point representing the activities in the Practice being unfulfilled
1	Initial understanding and ad hoc provision of Security Practice
2	Increase efficiency and/or effectiveness of the Security Practice
3	Comprehensive mastery of the Security Practice at scale

Legend

🟩	Objective Activity was met.
🟥	Objective Activity was not met.

Acquired software security impact

Dr. Devin Cook, Sandia Labs

Dr. Patrick Pape, MSU

Certification Test & Evaluation (CT&E)

- **Knowing the vulnerabilities of a system allows an organization to implement security features to protect those assets.**
- **Security features of the system can be**
  - **Technical (e.g. software or network devices)**
  - **Non-technical (e.g. physical protections)**
- **Evaluation of the security features is necessary to know how well they protect and what they do not protect.**

Certification Tools

- Simply installing security features to protect a system is not enough.
- Testing is required to ensure the security feature works correctly to protect against the vulnerabilities for which it was installed

Product Assurance – NSTISSP no. 11

- National Policy Governing the Acquisition of Information Assurance (IA) and IA-Enabled Information Technology Products that protect national security information.

- Effective 1 July 2002, all COTS IA and IA-Enabled products must be evaluated by
  - International Common Criteria Mutual Recognition Arrangement
  - NIAP Evaluation and Validation Program (CCEVS)
  - NIST FIPS validation program
- The evaluation/validation of COTS IA and IA-enabled products will be conducted by accredited commercial laboratories, or the NIST.

- All GOTS IA or IA enabled products must be evaluated by NSA or an NSA approved process.

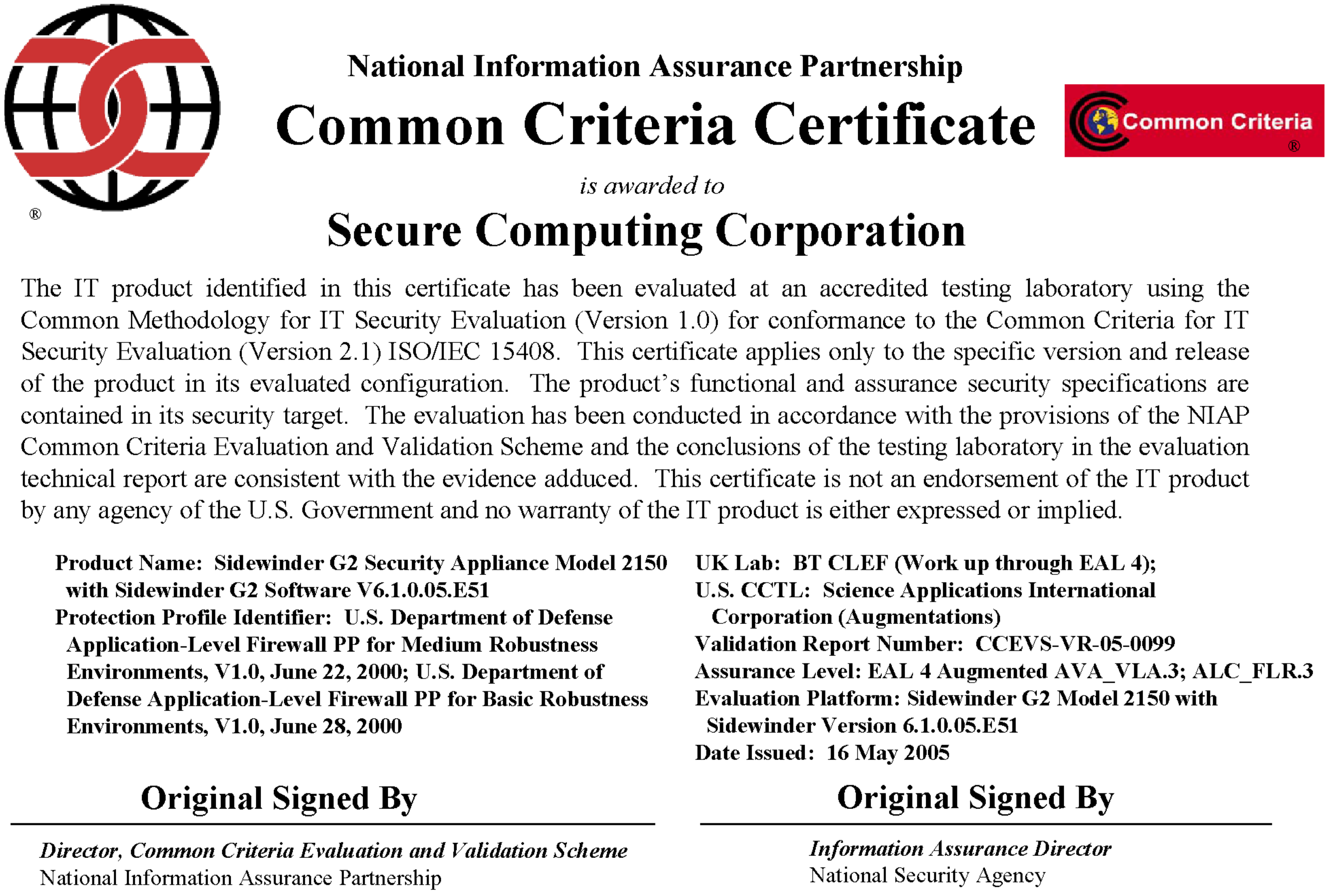
Protection Profiles (generic) & Security Targets (specific)

Protection Profile

- Introduction
- TOE Description
- Security Environment
  - Assumptions
  - Threats
  - Organizational security policies
- Security Objectives
- Security Requirements
  - Functional requirements
  - Assurance requirements
- Rationale

Security Target

- Introduction
- TOE Description
- Security Environment
  - Assumptions
  - Threats
  - Organizational security policies
- Security Objectives
- Security Requirements
  - Functional requirements
  - Assurance requirements
  - *TOE Summary Spec.*
- *Protection Profile Claims*
- Rationale



National Information Assurance Partnership

Common Criteria Certificate

is awarded to

Secure Computing Corporation

The IT product identified in this certificate has been evaluated at an accredited testing laboratory using the Common Methodology for IT Security Evaluation (Version 1.0) for conformance to the Common Criteria for IT Security Evaluation (Version 2.1) ISO/IEC 15408. This certificate applies only to the specific version and release of the product in its evaluated configuration. The product's functional and assurance security specifications are contained in its security target. The evaluation has been conducted in accordance with the provisions of the NIAP Common Criteria Evaluation and Validation Scheme and the conclusions of the testing laboratory in the evaluation technical report are consistent with the evidence adduced. This certificate is not an endorsement of the IT product by any agency of the U.S. Government and no warranty of the IT product is either expressed or implied.

Product Name: Sidewinder G2 Security Appliance Model 2150 with Sidewinder G2 Software V6.1.0.05.E51
Protection Profile Identifier: U.S. Department of Defense Application-Level Firewall PP for Medium Robustness Environments, V1.0, June 22, 2000; U.S. Department of Defense Application-Level Firewall PP for Basic Robustness Environments, V1.0, June 28, 2000

UK Lab: BT CLEF (Work up through EAL 4);
U.S. CCTL: Science Applications International Corporation (Augmentations)
Validation Report Number: CCEVS-VR-05-0099
Assurance Level: EAL 4 Augmented AVA_VLA.3; ALC_FLR.3
Evaluation Platform: Sidewinder G2 Model 2150 with Sidewinder Version 6.1.0.05.E51
Date Issued: 16 May 2005

Original Signed By

Director, Common Criteria Evaluation and Validation Scheme
National Information Assurance Partnership

Original Signed By

Information Assurance Director
National Security Agency

Contracting For Security Services

- As noted in Advertising Age -- vendors “sell stuff”.
- Does your agency have external partners or vendors?
- How do you turn a vendor into a partner?
  - Assuming you do not have unlimited funding….
- Evaluating vendor specifications.
- Integrating vendor products into information assurance architectures.

Facilities Planning

- To meet security requirements, facilities must support types and levels of protection necessary for equipment, data, information and applications to meet IS security policy.

Life Cycle System Security Planning

- Definition
  - Information Lifecycle Management comprises the policies, processes, practices, and tools used to align the business value of information with the most appropriate and cost effective IT infrastructure from the time information is conceived through its final disposition. Information is aligned with business processes through management policies and service levels associated with applications, metadata, information, and data.

    Storage Networking Industry Association http://www.snia.org
- Activities
  - Creation and Receipt
  - Distribution
  - Use
  - Maintenance
  - Disposition

System Security Architecture

- The security architecture consists of the mechanisms, inventions, and decisions intended to fulfill an application's security requirements.
- The security architecture diagram is part of the system architecture and is a low-level architecture diagram work product that documents the major security components and the application mechanisms and their relationships.
- Its objective is to document the security mechanisms of the application.
- It is validated against the system requirements and verified against security regulations and instructions.

Security Architecture and Continuity of Operations CONOPS

- facilitate proper and efficient security identification, authentication, authorization, administration in response to the access and use of information resources.
- are compliant with State laws and Federal regulations
- allow portability across platforms
- support multiple service delivery channels
- ensure that security requirements and associated risks are adequately evaluated
- support the integration of new technologies while maintaining appropriate security protection and requirements
- address and support multiple layers of protection, including network level, operating system, application level and data level security needs

Summary

- Security in the software development lifecycle
- Development environment security controls
- Software security effectiveness
- Acquired software security impact