Software Development Security

Domain 4

Overview

This domain involves processes and activities regarding the planning, programming, and management of software and systems, as well as referring to the controls that aren’t clued within systems and applications software and the steps used in their development.

The key objective of software development security is to ensure the confidentiality, integrity and availability of data.

For an application security program to be effective and applicable security policy must be in place.

Key Areas of Knowledge

- Understand and apply security in the software development life cycle
- Understand the environment and security controls
- Assess the effectiveness of software security

Agenda - 1

- Understand and apply security in the software development life cycle
- Understand the environment and security controls
- Assess the effectiveness of software security

Software Development Methods

Types of development methods include:

- Waterfall
- Prototyping
- Spiral
- Clean Room
- Extreme Programming

Waterfall Method

Waterfall

- Traditional Model
- Completion of one task leads to start of another
- Long term projects

Prototyping Method

Prototyping

- Address time issues with Waterfall
- Produce basic prototype that evolves each round
- Four main phases:
  - Initial concept
  - Design and implement initial prototype
  - Refine prototype until acceptable
  - Complete and release final version

Spiral Model

Spiral

- Combination of Waterfall and Prototyping
- Development of initial version is based on Prototyping
- Development of each version is similar to Waterfall

Clean Room Method

Clean Room

- Engineered development process
- Focus on Defect Prevention instead of Defect Removal
- Increased Design time reduces requirements for testing
- Results in substantial savings over the long term

Extreme Programming

Extreme Programming

- Function specific programming instead of application
- Executed by small teams
- Dynamically changing requirements

Software Development Models

- **Different models are used to manage software development**
- **There are similar basic processes:**
  - Project Initiation
  - Functional Design Analysis and Planning
  - System Design Specifications
  - Software Development
  - Installation / Test / Implementation
  - Operational / Maintenance
  - Disposal

Project Initiation

Project Initiation

- Conceptual definition of project is decided upon
- Identify security requirements
- Perform an initial risk analysis
  - Analyze threats and countermeasures
  - Estimate costs and benefits per countermeasure
- Identify security framework
- Determine service level agreement

Functional Design

Functional Design Analysis and Planning

- Define security requirements
- Propose security checkpoints in plan
- Develop contingency plans
- Preliminary security test plans
- Formal functional baseline includes security requirements

System Design

System Design Specifications

- Define security specifications
- Update test plans involving security
- Security specifications
- System design checklist
- Formal methods developed

Software Development

Software Development

- Write programming code to meet specifications
- Implement security within code
- Perform unit tests

Testing and Installation

Installation / Test

- Test system components
- User acceptance testing, data checking, and resiliency testing
- Install system
- Create manuals
- Perform acceptance test – certification and accreditation
- Accept system
- Garbage collection

Testing Types

- **Unit testing**
  - Individual component is in a controlled environment where programmers validate data structure, logic, and boundary conditions.
- **Integration testing**
  - Verifying that 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

Testing techniques

- Black-box testing
- White-box testing
- Gray-box testing

Operation and Disposal

Operation / Maintenance

- Maintain system through service level agreement
- After changes, re-certify
- Audit and test security components periodically

Disposal

- Properly dispose of system
- Repeat full cycle with a new project initiation
- Data moved to another system or discarded

Verification vs. Validation

Verification

- Determining if the product accurately represents the developer’s description and specifications
- Follows structure and logic of product

Validation

- Determining the degree to which the model represents a real world use
- Does the product actually solve a problem and/or meet a demand?

Change Control

- Without proper change control, a project can take longer than appropriate to complete
- Applications should be centrally managed
- Security controls must always be protected
- Programmer changes must be made through controlled libraries
- Changes should be made to source code and not production code

Change Control Procedure – 1 of 2

1. **Request for change is made formally to CCB**
2. **Analyze request**
   - Develop the implementation strategy
   - Calculate the cost of this implementation
   - Review any security implications
3. **Record change request**
4. **Submit change request for approval**
5. **Develop change**
6. **Re-code segments of the product, modify functionality**

Change Control Procedure – 2 of 2

7. Link these changes in the code to the formal change control request
8. Submit software for testing and quality approval
9. Repeat until quality is adequate
10. Make version changes
11. Report changes to management
12. Establish a baseline setting

Change Control Issues

Key Points:

- Changes must be submitted, approved, tested, and recorded
- Logic of change is separate from the process of change control

Computer Aided Software Engineering

CASE Tools

- General term for a wide variety of tools to help programmers, project managers, and analysts during a project
- Automation of repetitive tasks
- Debuggers, code analyzers, version control tools
- Development is quicker and more accurate with computer aided tools
- Allows for rapid prototyping

Capabilities Maturity Model

- Developed by Software Engineering Institute at Carnegie-Mellon University in Pittsburgh
- Assumes that the quality of the software development process directly impacts the quality of the software product
- Five maturity levels for software development

Capabilities Maturity Model Levels

- **Level 1 – Initiating**
  - Ad-hoc procedures
  - Few processes are defined
- **Level 2 – Repeatable**
  - Software development can be repeated
  - Basic project management processes track cost, schedule, and functionality
- **Level 3 – Defined**
  - Engineering procedures are documented, standardized, and integrated into a standard process for the organization

Capabilities Maturity Model Levels

Level 4 – Managed

- Detailed measures of the software process and product quality are collected
- Both the software process and products are quantitatively understood and controlled

Level 5 – Optimizing

- Continuous process improvement is enabled by quantitative feedback from the process and from piloting innovative ideas and technologies

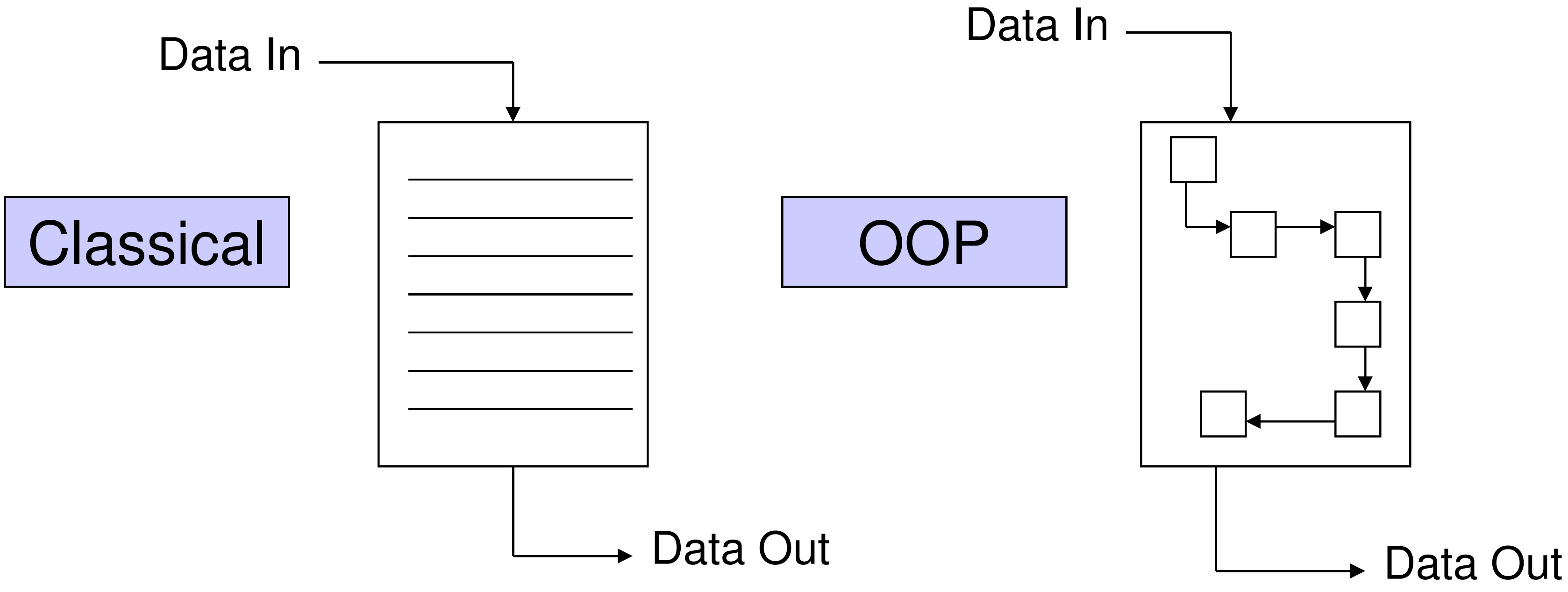
Object Oriented Programming

Benefits

- Increases speed in software development because objects can be re-used
- Saves money and time
- Closely maps to real activities in the business world
- Self contained
- Highly modular

Object Oriented Programming

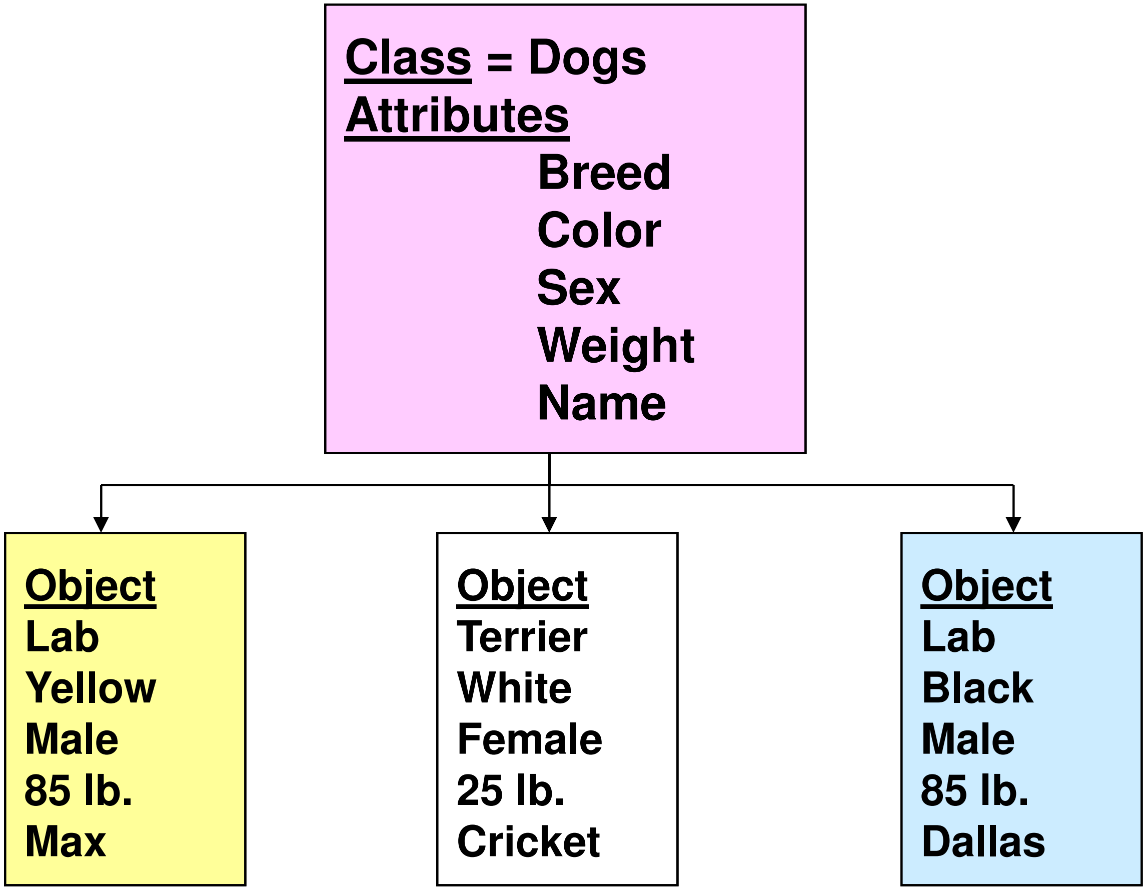
- **Comparison of the two methodologies:**



OOP Terminology

- **Classes – define attributes and characteristics of the possible objects within them**
  - Objects are members of classes
- **Objects – software entities that are grouped into classes**
  - Objects inherit attributes from class type

OOP Example



Class = Dogs

Attributes

Breed
Color
Sex
Weight
Name

Object
Lab
Yellow
Male
85 lb.
Max

Object
Terrier
White
Female
25 lb.
Cricket

Object
Lab
Black
Male
85 lb.
Dallas

OOP Objects

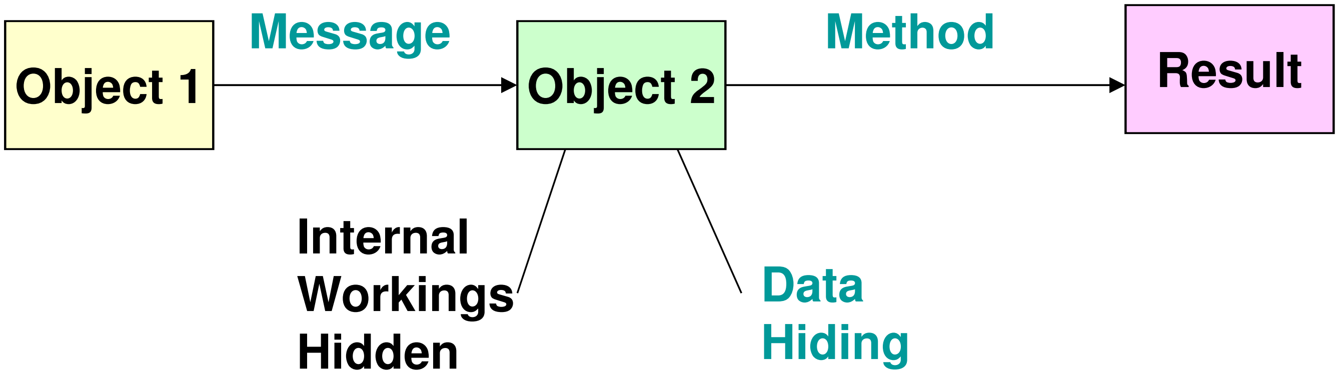
Objects

- Re-usable module of code
- Each reuse of the object creates an instance
- Modularity approach provides more efficiency and structure

OOP Behavior

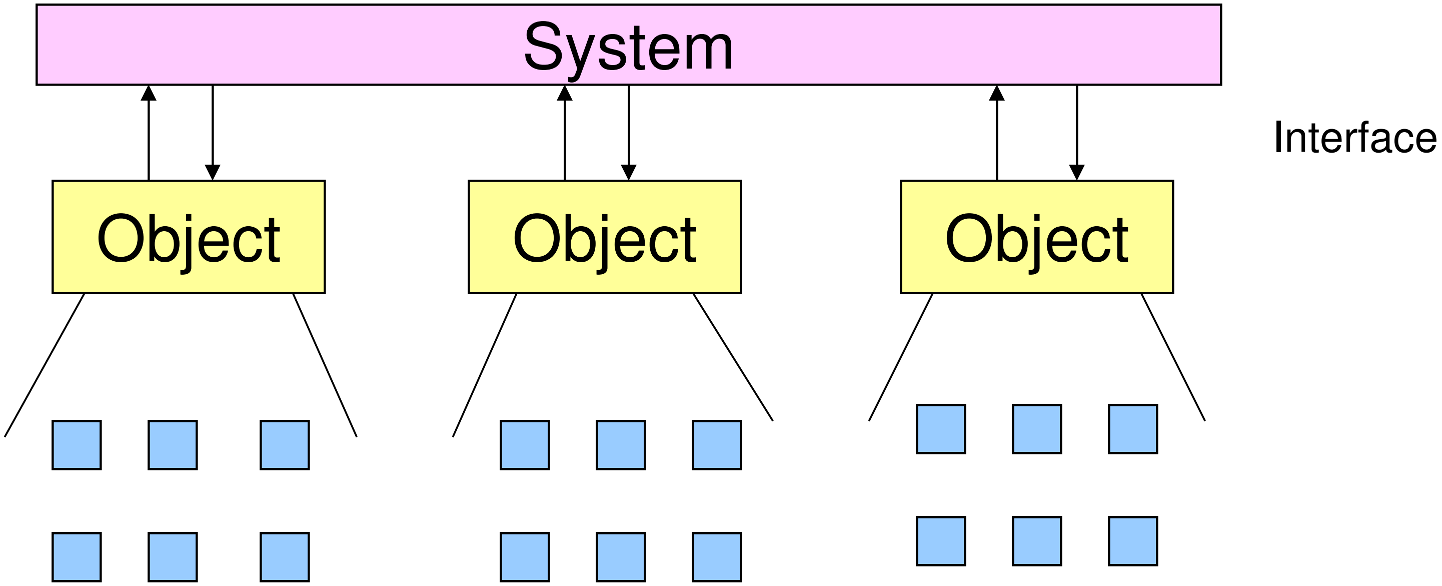
Behavior of Objects

- Commands performed by object are a **method**
- Objects communicate with each other through **messages**



Abstraction

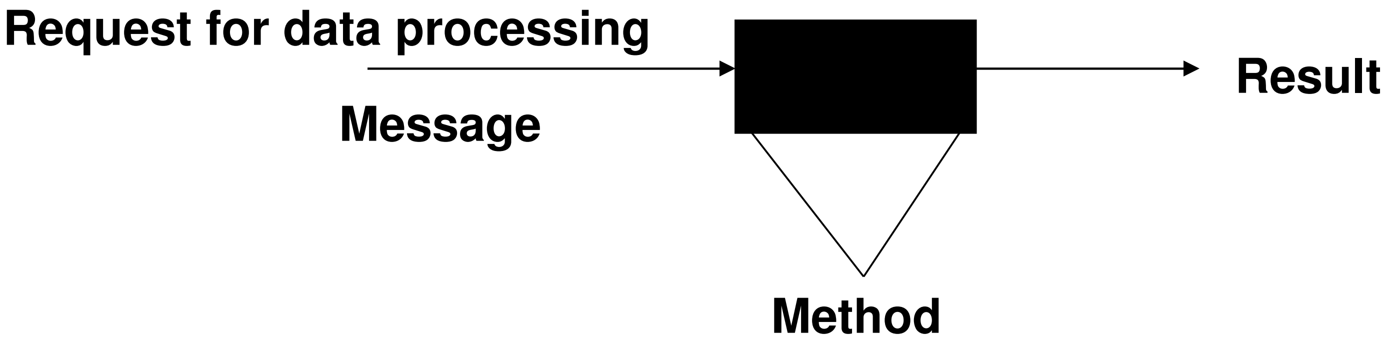
- Looking at the "big picture" versus focusing on the all the small details of the picture
- Suppressing unnecessary detail so that the overall goal of an object can be utilized



Abstraction Example

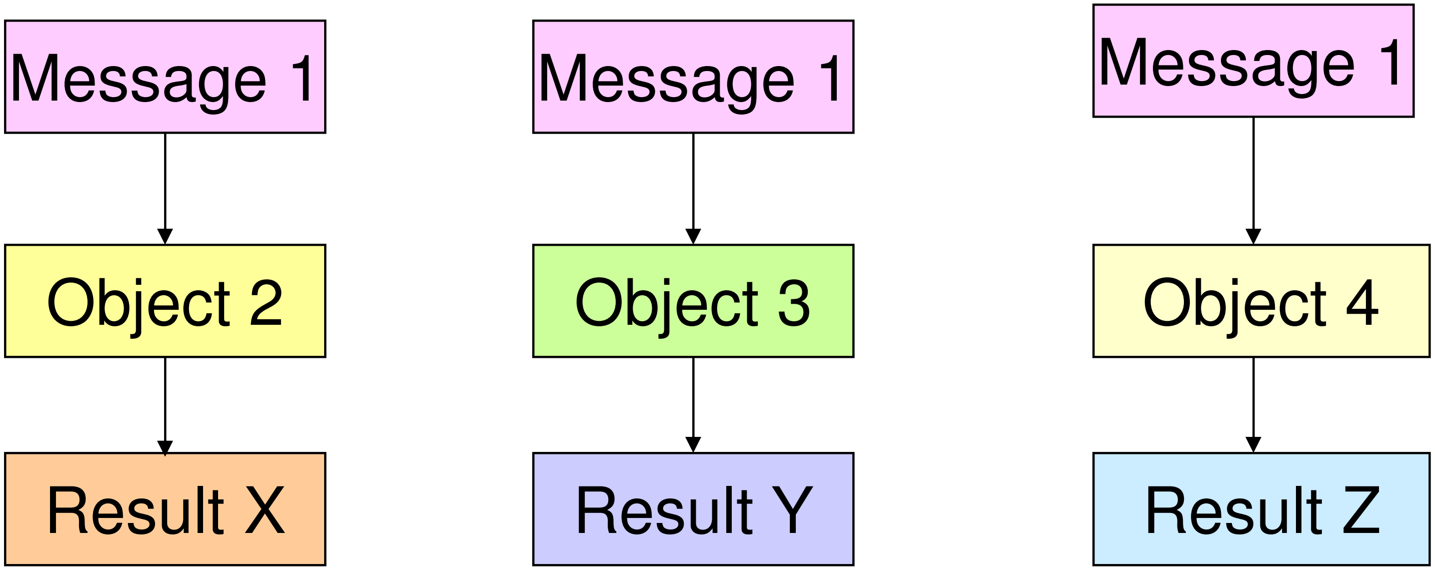
Objects

- Characteristics of objects are encapsulated
  - Only accessible through messaging
- Viewed as a “black box”
  - Internal workings are not apparent to the system

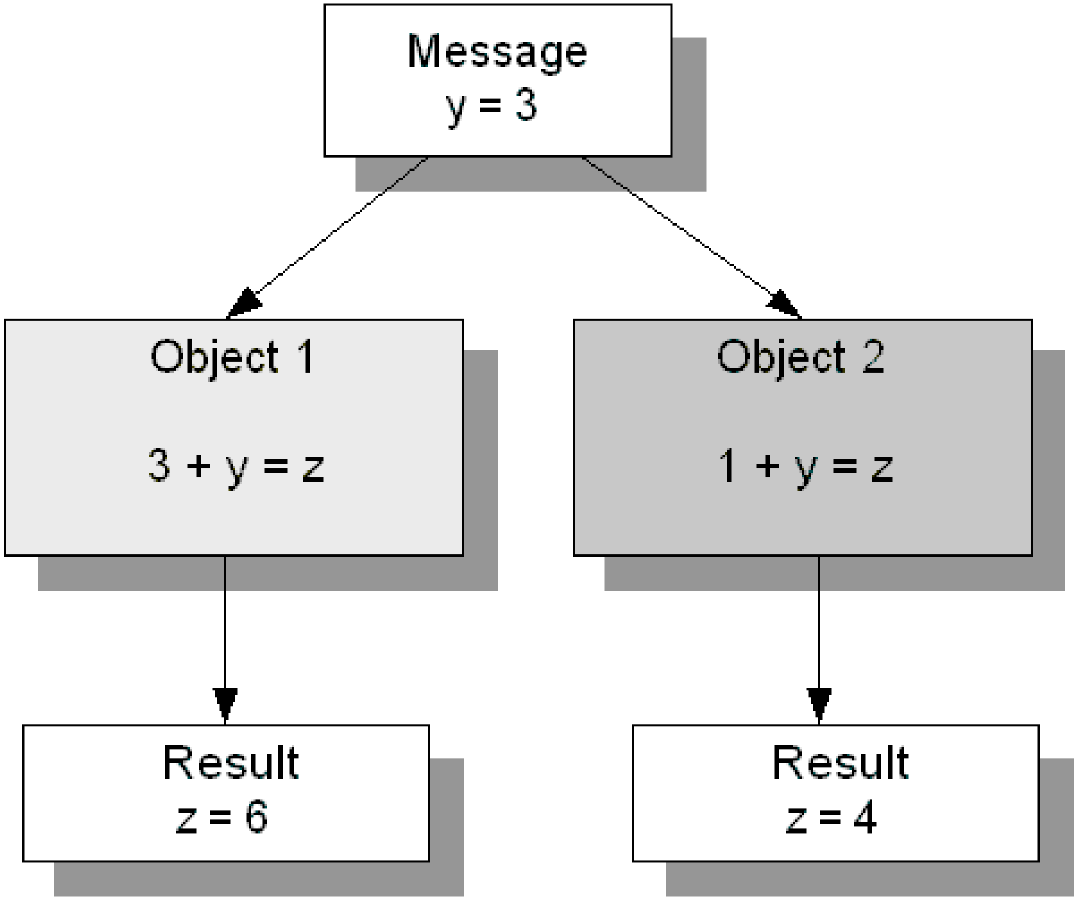


Polymorphism

- **Two objects are sent the same message but react differently**
  - This is possible because objects can belong to different classes, meaning they will exhibit different behaviors



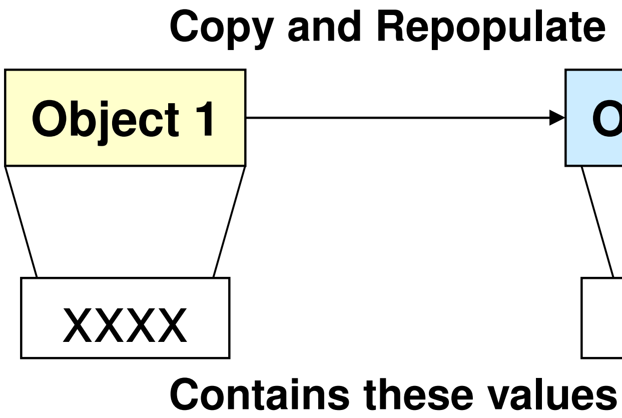
Polymorphism Logic



Polyinstantiation

- **Polyinstantiation is the creation of another version of an object using different values for its variables to ensure that lower level subjects do not access data at a higher classification**

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



Module Interaction – Cohesive

Cohesive

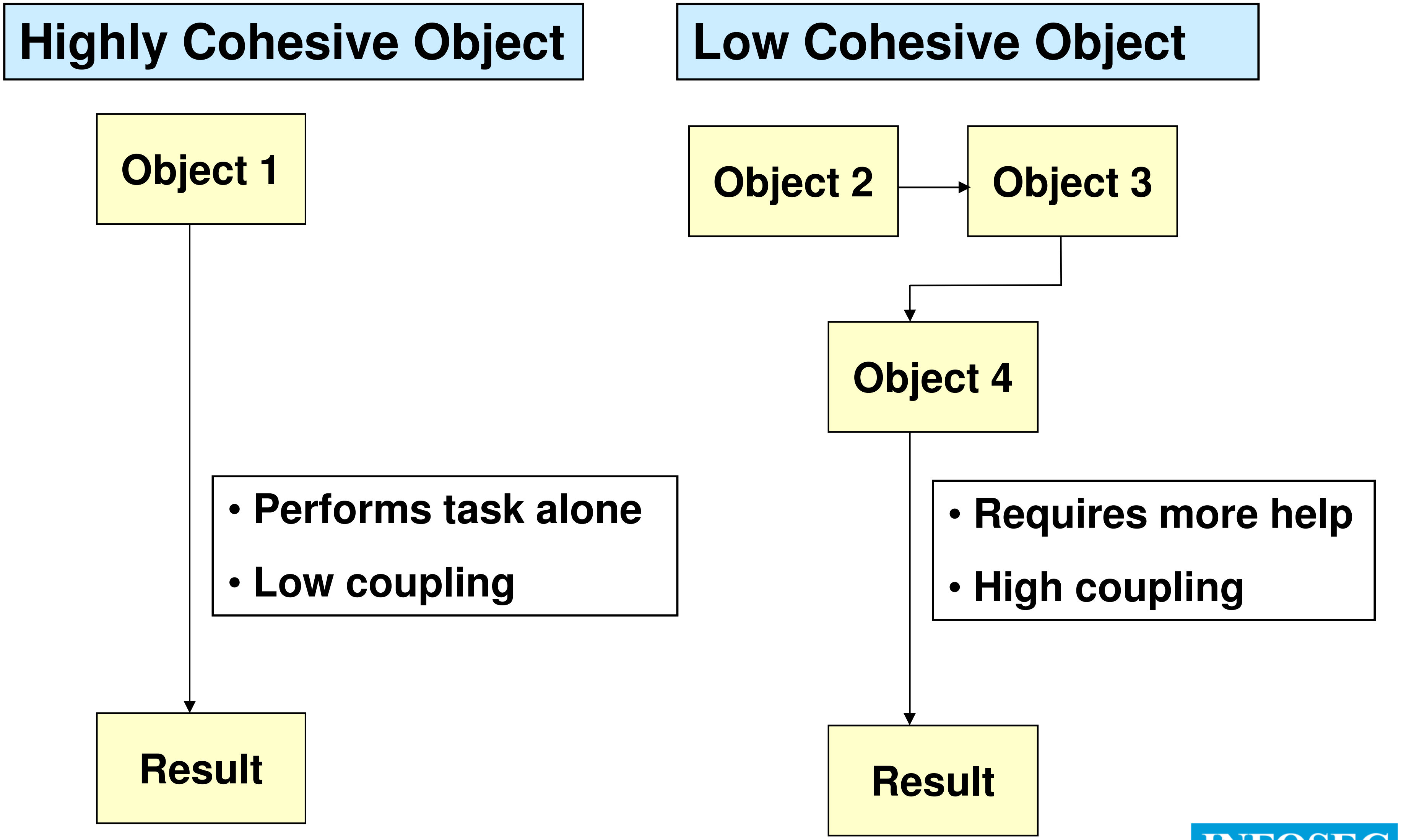
- Object is highly cohesive if it can perform a task with little or no help from others
- Highly cohesive objects are not as dependent upon other objects as low cohesive objects
- Higher cohesive objects are better (within reason)

Module Interaction – Coupling

Coupling

- Measurement of interaction between objects
- Lower coupling means less interaction
- Lower coupling provides better software design because objects are more independent
- Lower coupling is easier to troubleshoot and update

Cohesion and Coupling Example



Highly Cohesive Object

Low Cohesive Object

Object 1

Object 2

Object 3

Object 4

- Performs task alone
- Low coupling

- Requires more help
- High coupling

Result

Result

Databases

Topics:

- DB Management System
- Models
- Terminology
- Security Issues
- Distributed Databases
- Data Mining

Database Model

Model Characteristics

- Describes relationships between data elements
- Tool used to represent the conceptual organization of data
- Formal method of representing information

Database Types

Relational Database

- Presents data in tables with columns and rows (attributes and tuples)

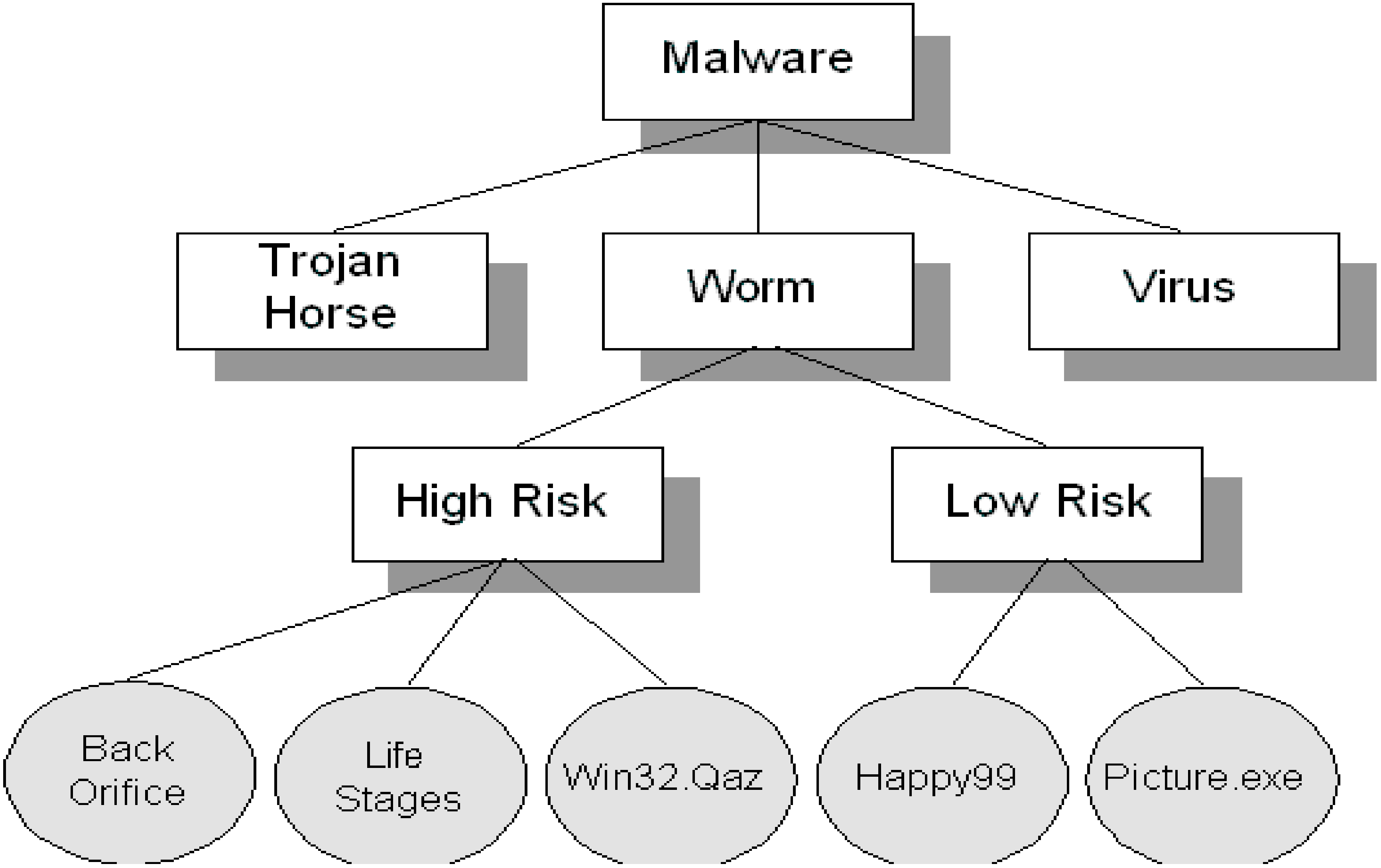
Hierarchical Database

- Logical tree structure – branches and leaves (data fields)
- Highest node is root; parents can have “children”

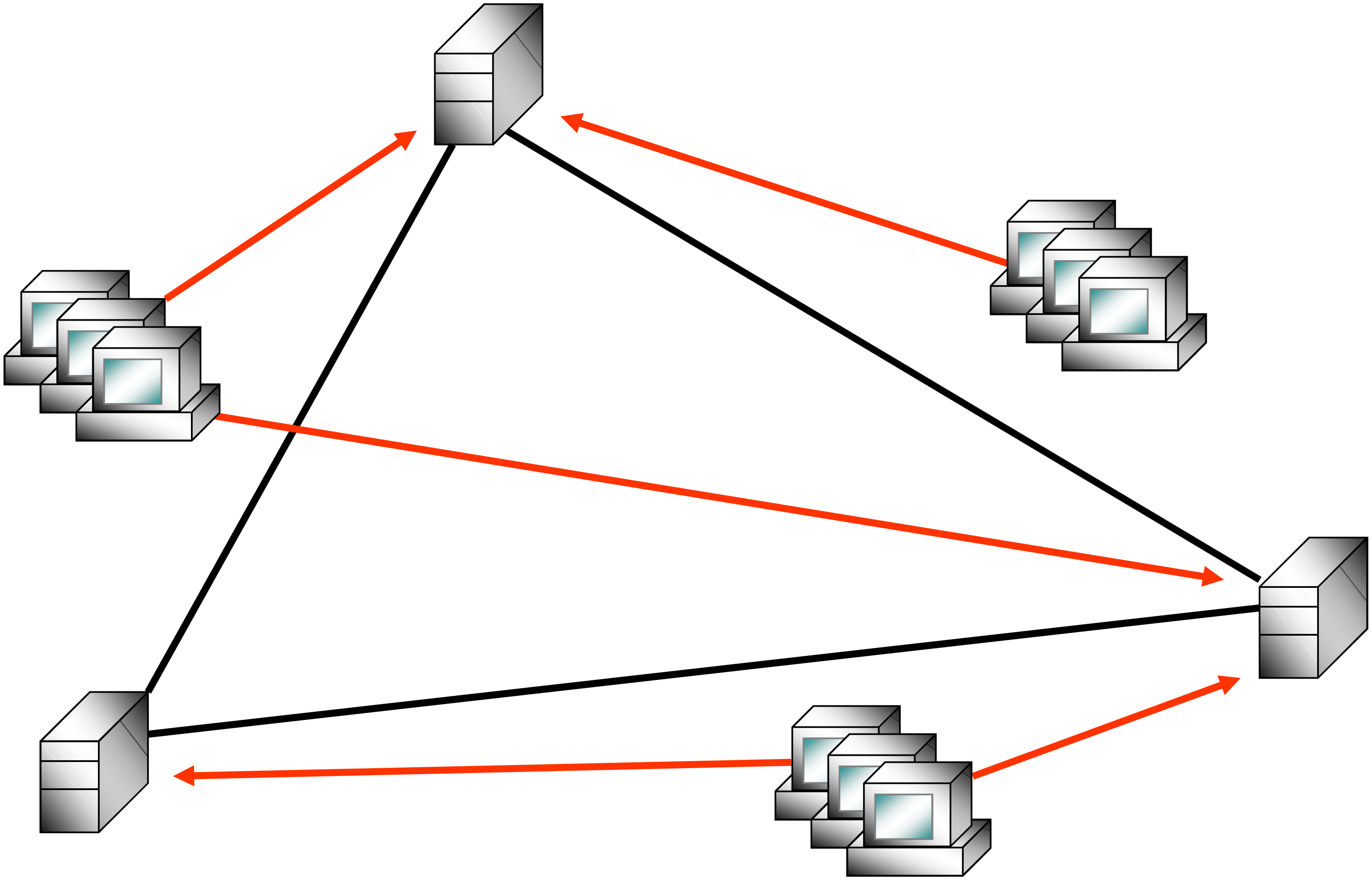
Distributed Database

- Data is stored in databases in different places
- Databases are logically connected

Database Models – Hierarchical

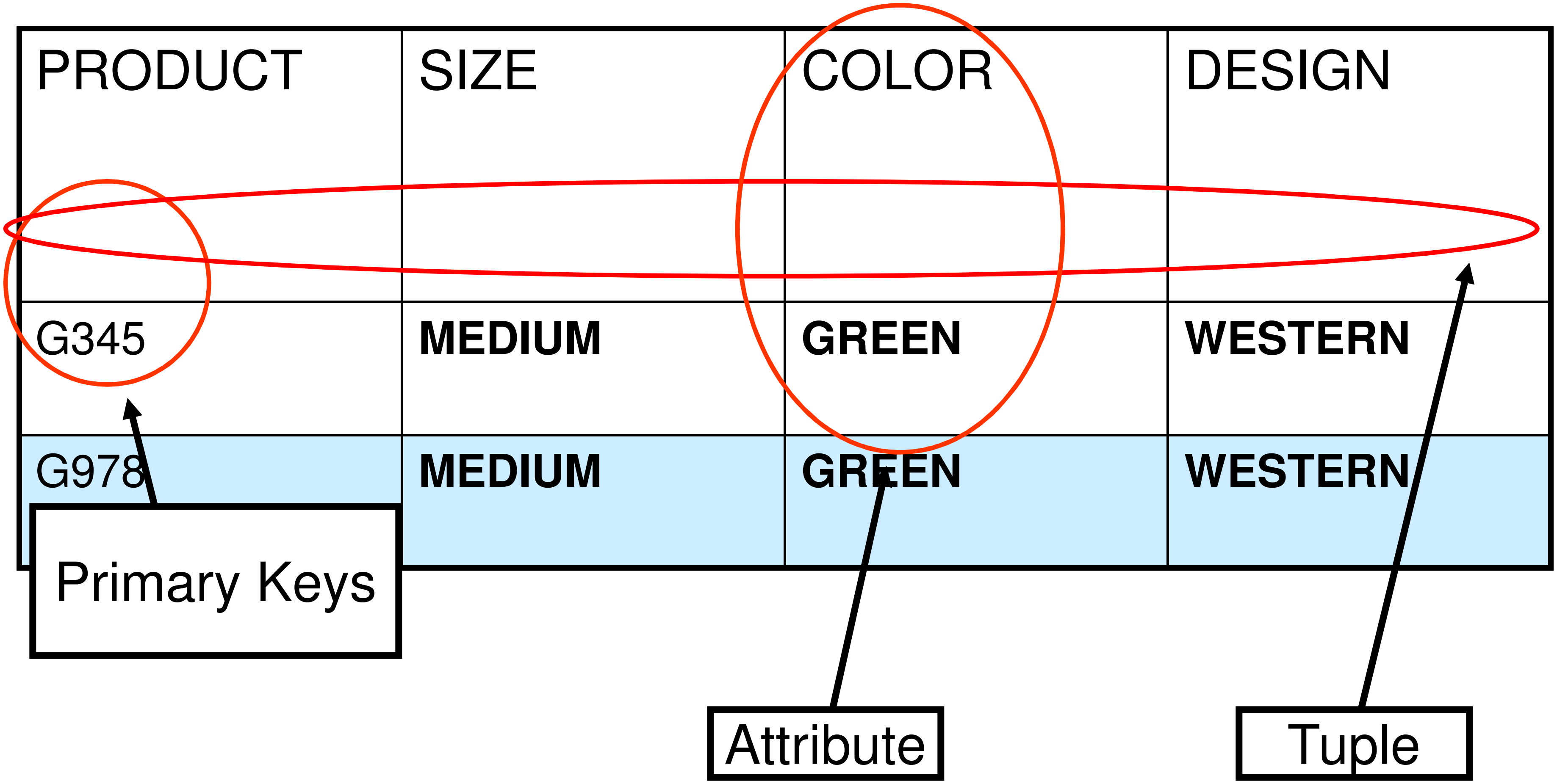


Database Types – Distributed



Location Transparency

Database Models – Relational



PRODUCT	SIZE	COLOR	DESIGN
G345	**MEDIUM**	**GREEN**	**WESTERN**
G978	**MEDIUM**	**GREEN**	**WESTERN**

Primary Keys

Attribute

Tuple

Database Systems

Database

- A collection of interrelated data stored in a meaningful way
- Allows multiple users and applications to access, view, and modify data as needed

Database Management System (DBMS)

- Software that allows user to access the database
- Enforces access control restrictions, provides data integrity and redundancy
- Provides different procedures for data access

Relational Database Terminology

- **Attribute**
  - A column in a relation
- **Tuple**
  - A row representing a relationship among values
- **Cell**
  - the intersection of a row and a column
- **Element**
  - Each data value in a row under a column

Relational Database Terminology

File

- A collection of records of the same type
- Also known as a table

Base Relation

- A table stored in a database

View

- A virtual relation defined by database to control subjects viewing certain data (a.k.a. role-based access control)

Relational Database Terminology

Primary Key

- a field that links all the data within a record to a corresponding value

Foreign Key

- an attribute of one table that is the primary key of another table

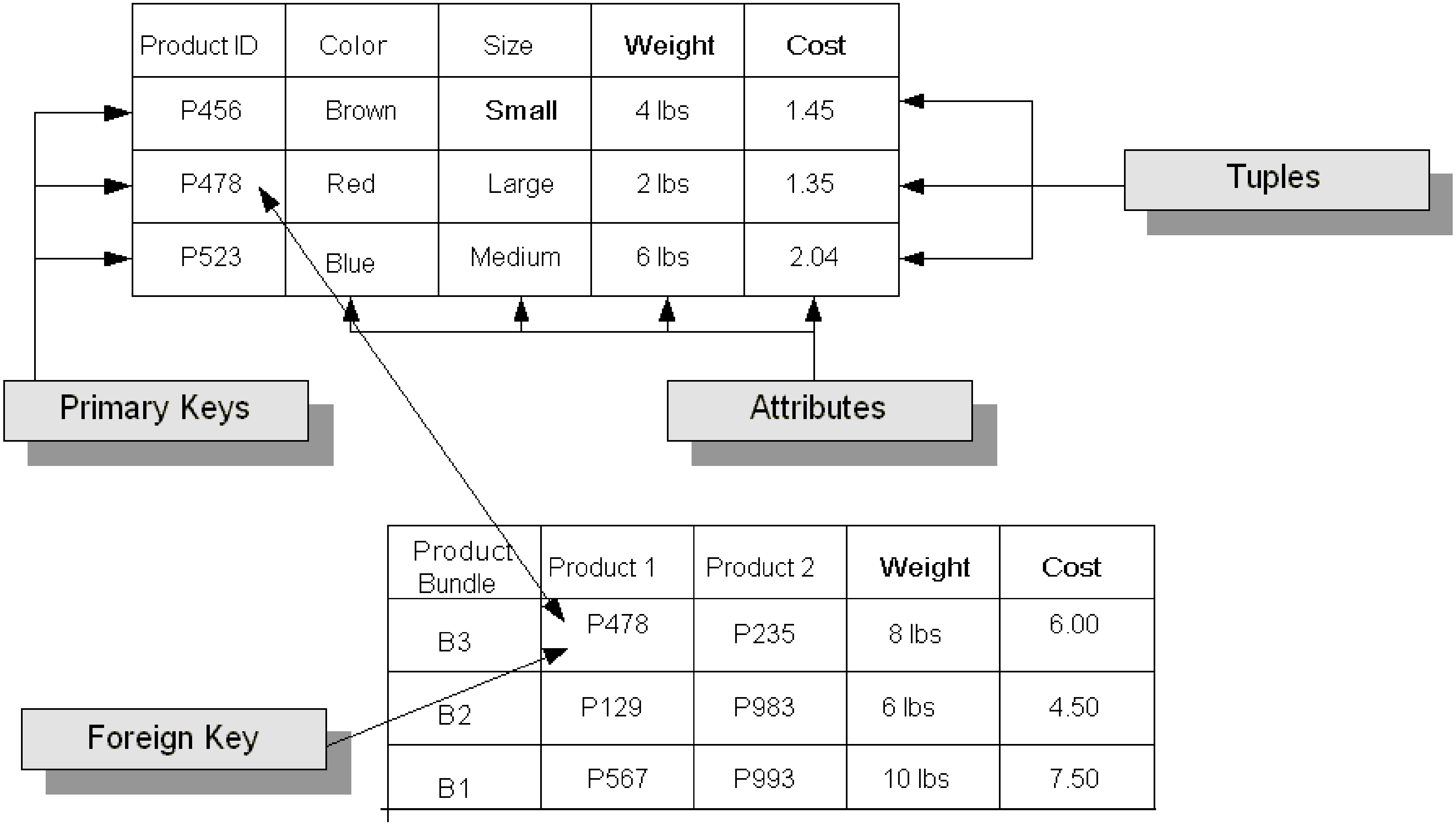
Schema

- Holds data that describes a database
- Each relation in a database is described by the schema

Meta-data

- “Data about data”
- Data used to describe a database and the data within it

Foreign Key Example



Product ID	Color	Size	Weight	Cost
P456	Brown	Small	4 lbs	1.45
P478	Red	Large	2 lbs	1.35
P523	Blue	Medium	6 lbs	2.04

Tuples

Primary Keys

Attributes

Product Bundle	Product 1	Product 2	Weight	Cost
B3	P478	P235	8 lbs	6.00
B2	P129	P983	6 lbs	4.50
B1	P567	P993	10 lbs	7.50

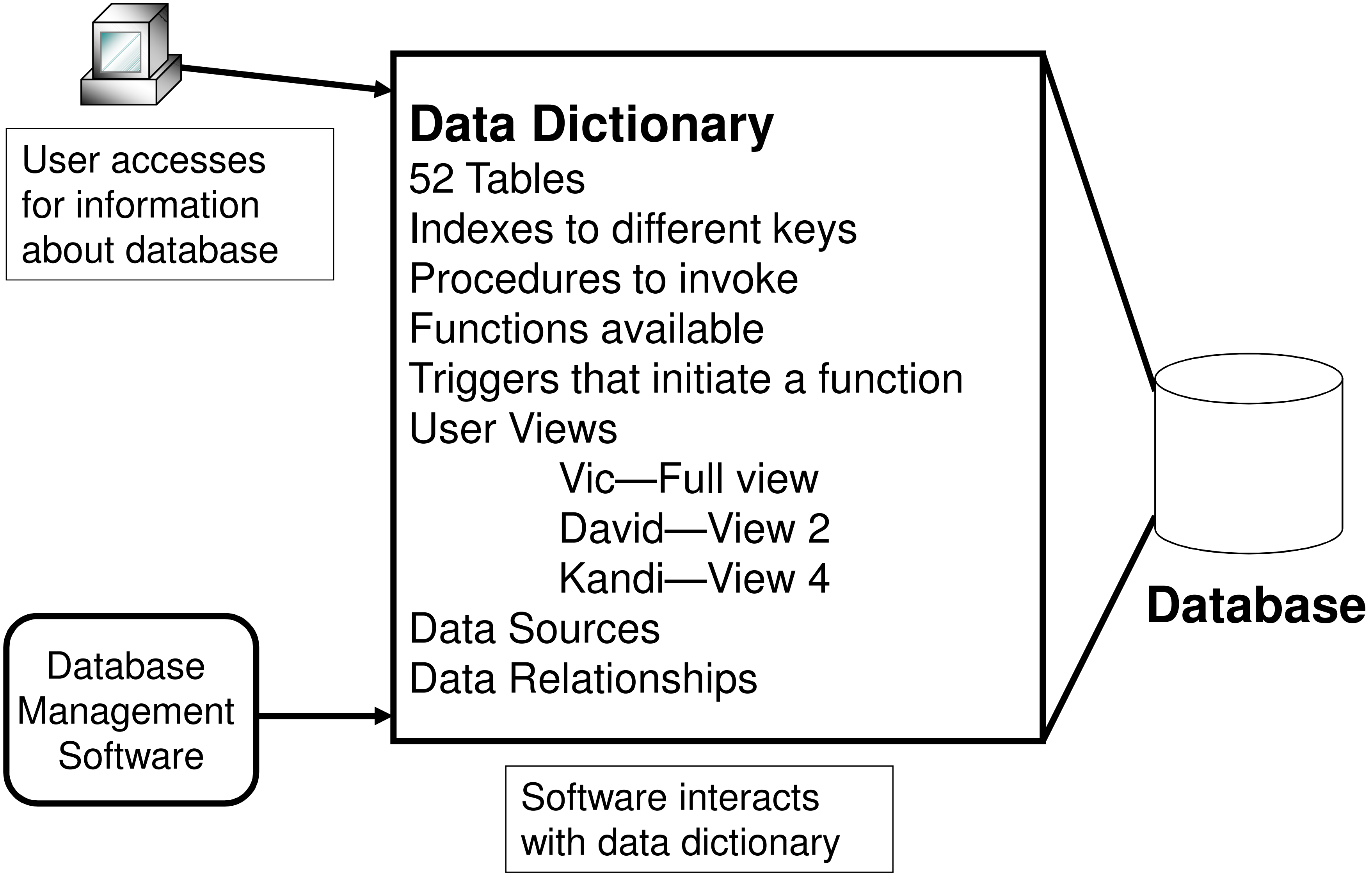
Foreign Key

Database Components

Data Dictionary

- Central repository of meta-data, which is critical information about the whole database
- Cross reference between the groups of data elements and how they interact with each other (relationships)
- A central collection of data element definitions, schema objects, and reference keys
- Allows for central management of database
- Central catalog of data

Data Dictionary



User accesses for information about database

Data Dictionary

52 Tables

Indexes to different keys

Procedures to invoke

Functions available

Triggers that initiate a function

User Views

- Vic—Full view
- David—View 2
- Kandi—View 4

Data Sources

Data Relationships

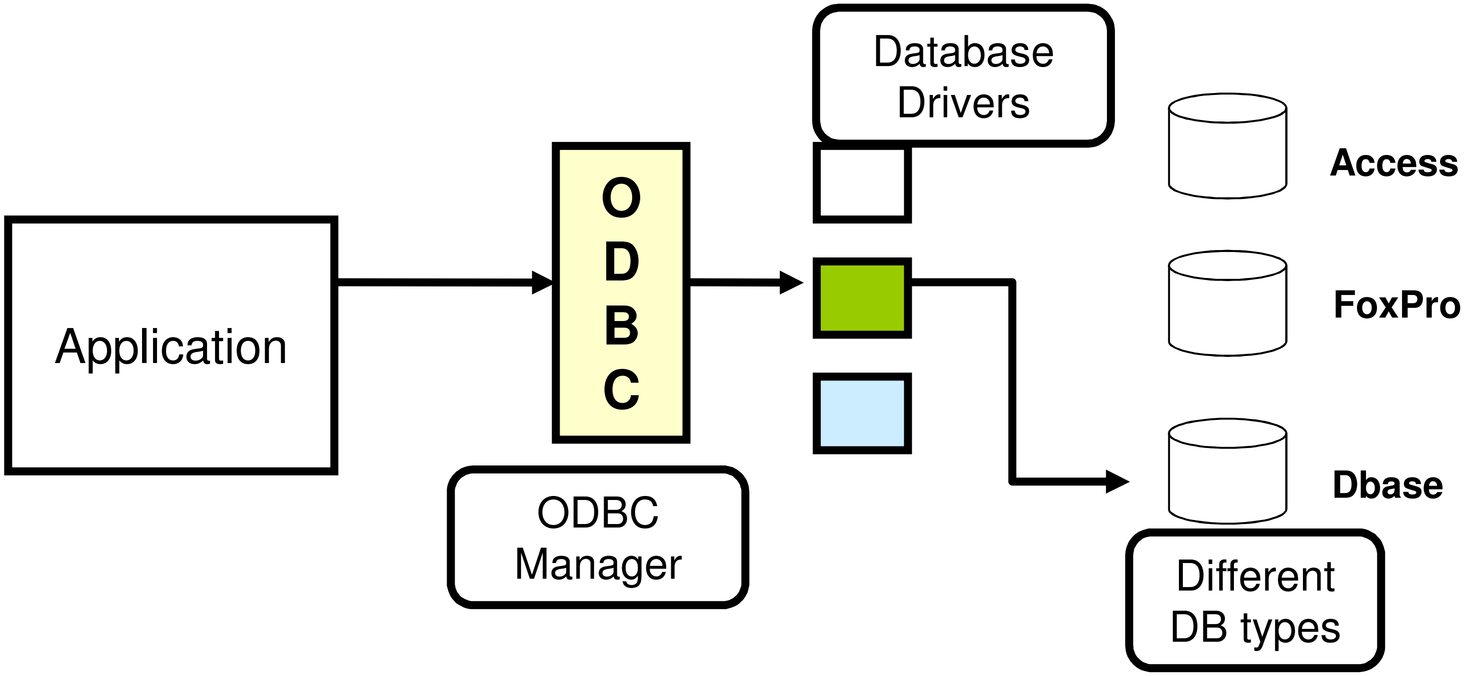
Database

Database Management Software

Software interacts with data dictionary

Open Database Connectivity (ODBC)

- Allows applications to communicate with different types of databases
- Interface between the application and the different database drivers

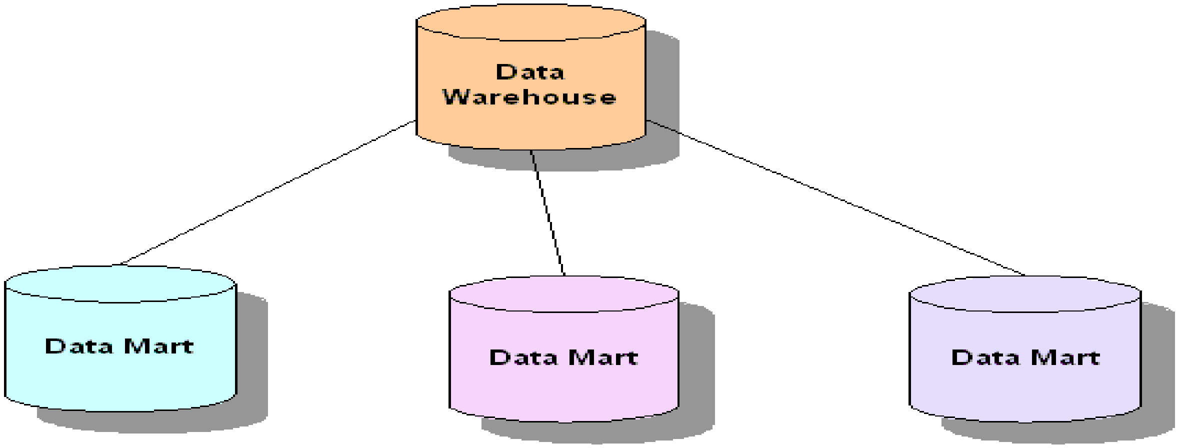


Data Warehousing

- Combines data from multiple databases into a large database with the purpose of a fuller extent of information retrieval and data analysis
- Extracting operational data and making it into informational data
- Not just mirroring data
- Redundancies and inconsistencies are removed
  - normalized

Characteristics of a Data Mart

- Smaller and more focused on a particular subject when compared to data warehouses
- A Data Warehouse combines data across an entire enterprise
- A Data Mart can access data for specific groups



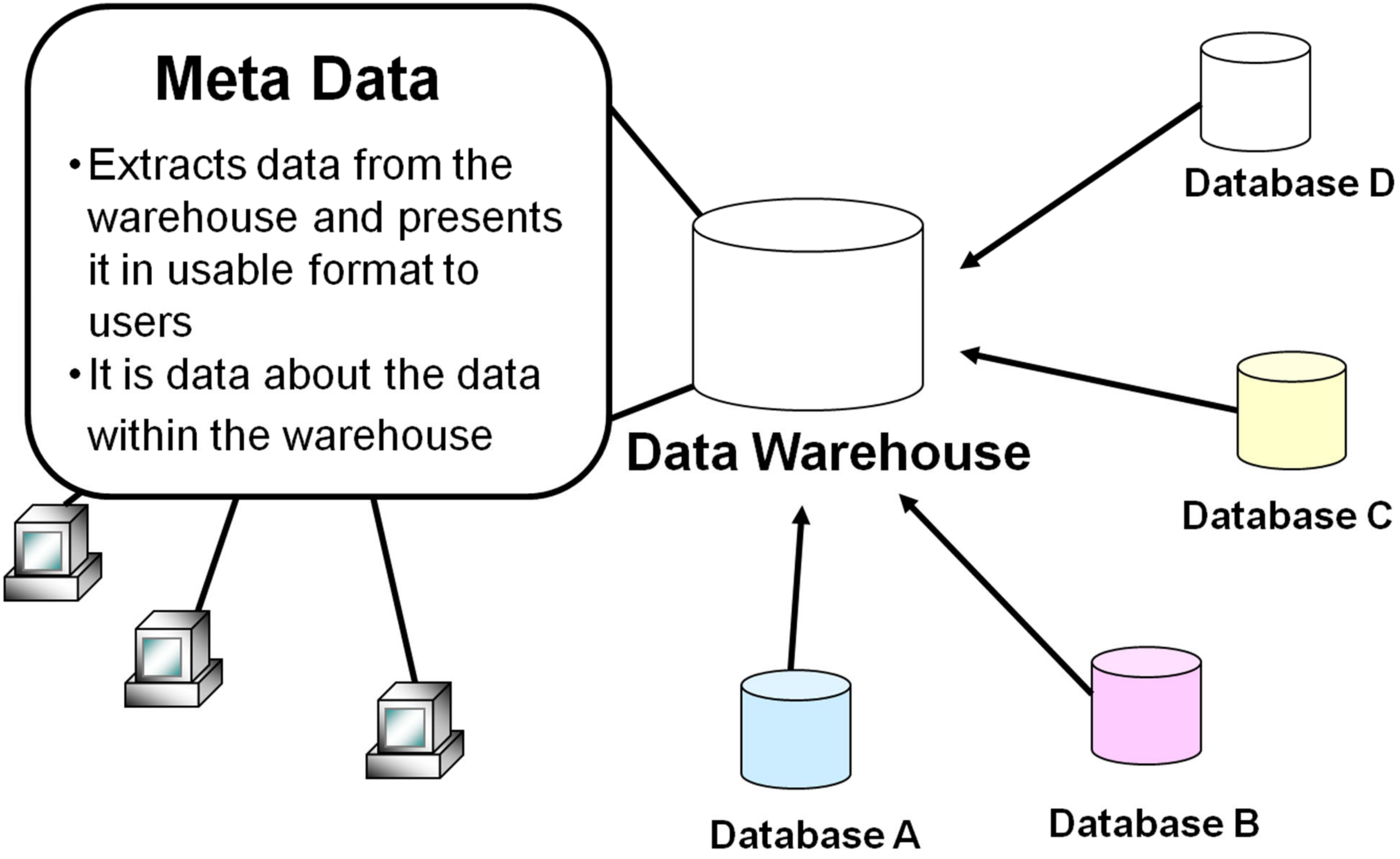
Data Mining

- Data mining is the process of analyzing a database using tools that look for trends or anomalies without having the knowledge of the meaning of the data
- Creates meta-data, which is more useful to the user
- Identifies patterns and relationships between data sets
- Can be used to test for inference vulnerabilities

Data Mining Characteristics

- **Can be applied to databases or warehouses**
- **Set of automated tools that convert data into more useful information**
- **May be used in:**
  - Intrusion detection
  - Fraud detection
  - Auditing to identify abnormal patterns

Meta Data



Meta Data

- Extracts data from the warehouse and presents it in usable format to users
- It is data about the data within the warehouse

Agenda - 2

- Understand and apply security in the software development life cycle

- Understand the environment and security controls

- Assess the effectiveness of software security

Software Security Downfalls

Issues

- Most attacks take advantage of a weakness or vulnerability in some type of software
- Security controls are deployed to prevent attackers from exploiting those weaknesses
- Software is a critical component of all business transactions
- Software is changing faster than software security technology

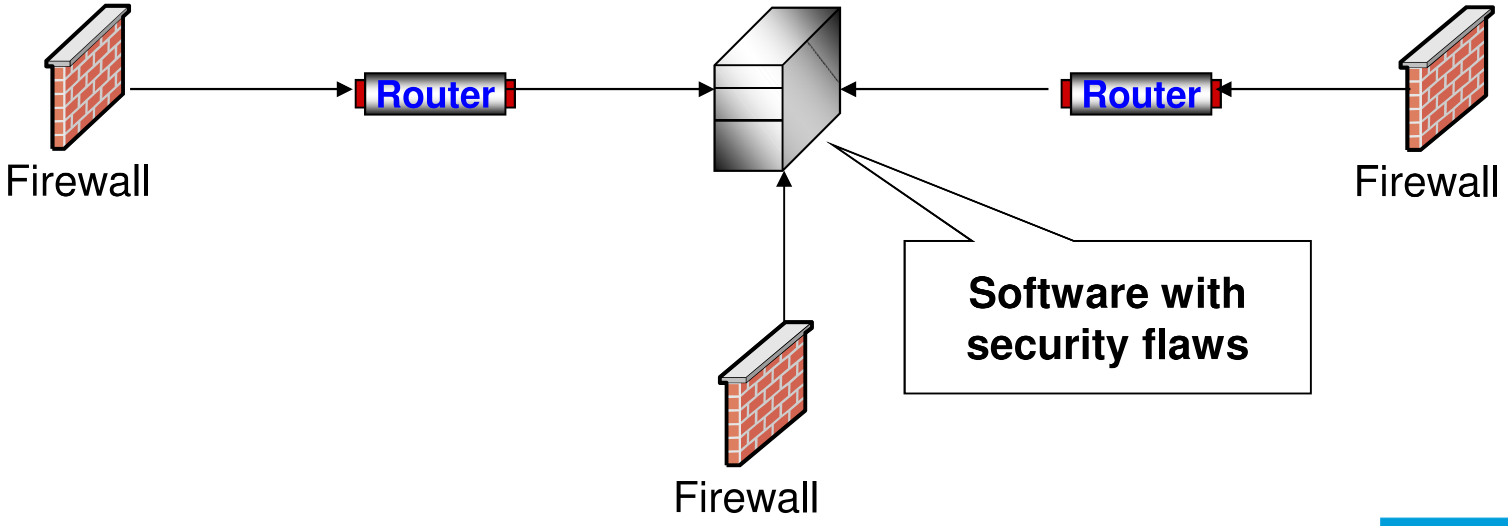
Software Security Challenges

- **Some reasons why implementing security into software has been lacking:**
  - It was not as crucial to implement security during the development stages
  - Many programmers do not practice these procedures
  - A majority of security professionals themselves are not software developers
  - Software vendors rush products to market with their eyes set on functionality, not security

Approaches To Security

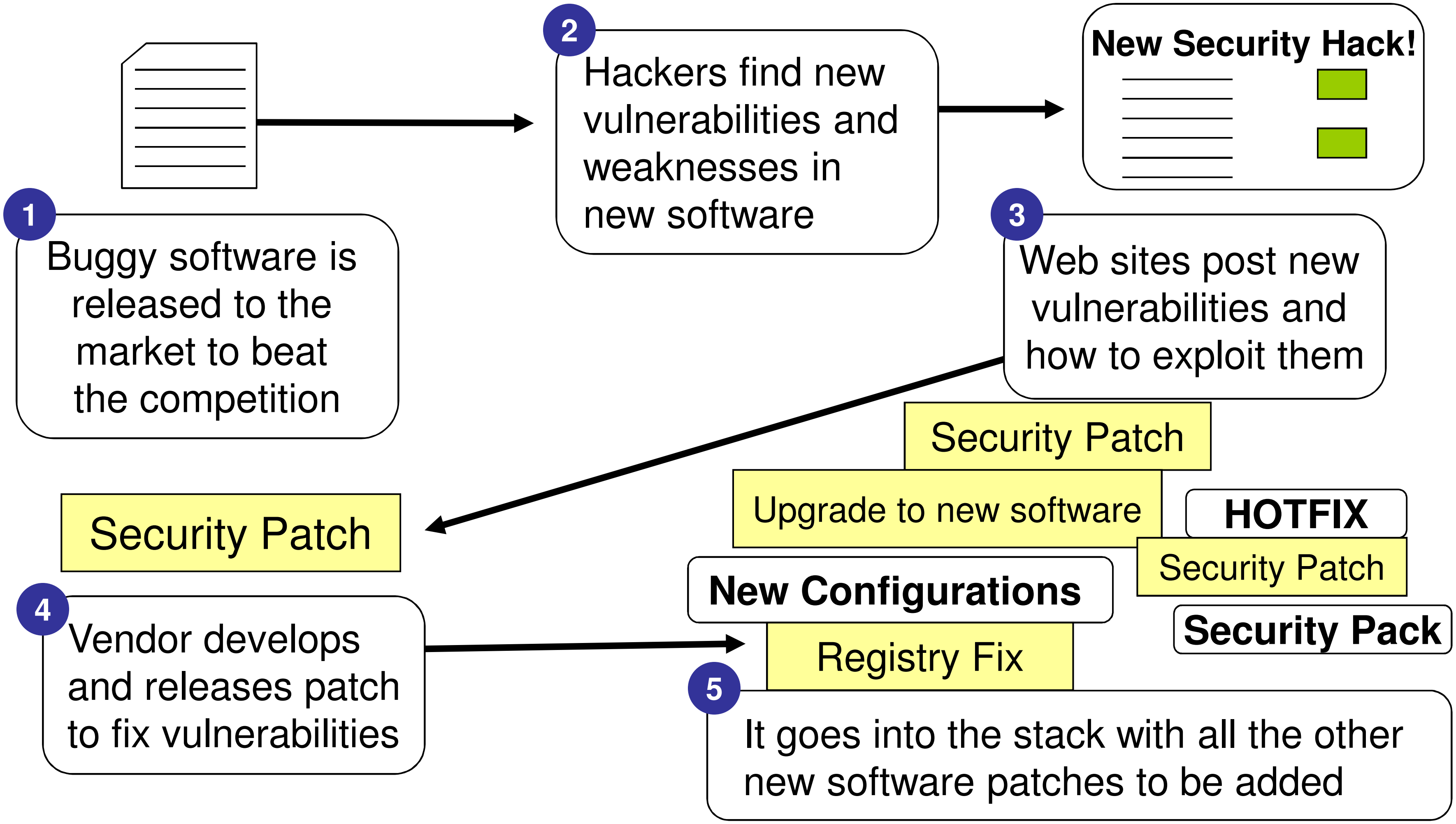
Device vs. Software Protection Mechanisms

- Option 1: Firewalls, Routers, ACLs, IDS, bastion hosts
- Option 2: Design and develop software with security in mind
  - – ISO/IEC 27034: Information Technology Security Techniques & Guidelines For Application Security (Draft)



INFOSEC INSTITUTE

Common Process for Addressing Flaws



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

2 Hackers find new vulnerabilities and weaknesses in new software

New Security Hack!

3 Web sites post new vulnerabilities and how to exploit them

Security Patch

4 Vendor develops and releases patch to fix vulnerabilities

Security Patch

Upgrade to new software

HOTFIX

Security Patch

New Configurations

Security Pack

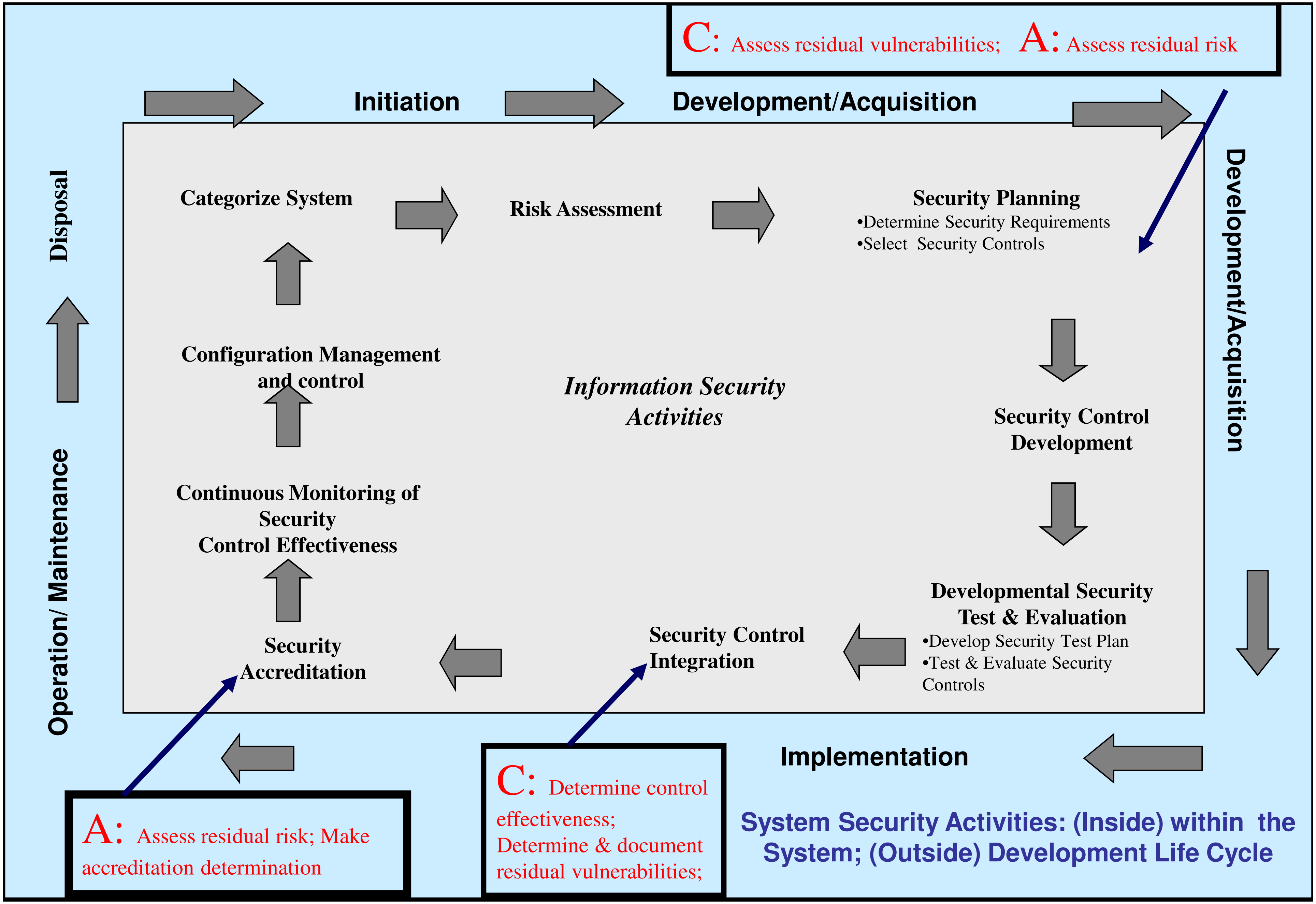
Registry Fix

5 It goes into the stack with all the other new software patches to be added

Project Development

Security Within Software Development

- Security should be planned and managed throughout the life cycle of a system
- Security controls should not be added in as an afterthought, which is more time-consuming
- One administrative control is to ensure that security within projects is managed properly



C: Assess residual vulnerabilities; A: Assess residual risk

Initiation

Development/Acquisition

Disposal

Categorize System

Risk Assessment

Security Planning
- Determine Security Requirements
- Select Security Controls

Configuration Management and control

Information Security Activities

Security Control Development

Continuous Monitoring of Security Control Effectiveness

Developmental Security Test & Evaluation
- Develop Security Test Plan
- Test & Evaluate Security Controls

Security Accreditation

Security Control Integration

Development/Acquisition

Operation/ Maintenance

Implementation

A: Assess residual risk; Make accreditation determination

C: Determine control effectiveness; Determine & document residual vulnerabilities;

System Security Activities: (Inside) within the System; (Outside) Development Life Cycle

Assurance – Operational vs. Life Cycle

Operational Assurance

- Focus on features and architecture of a system
  - System integrity, trusted recovery, covert channels
- Software development and functionality issues

Life Cycle Assurance

- Ensures that the TCB is designed, developed, and maintained with formally controlled standards that enforce protection at each stage in the system's life cycle
- Requires security testing and trusted distribution
- Configuration management

Separation of Duties

- **Programmers should not be the only ones testing their work**
  - Operations should not have access to source or object code in production
  - Programmers should not be interacting with software in production
- **Once software is completed, it should go to the library**
  - Software released to production should come from a library, and not directly from programmers
  - Proper testing is management’s responsibility

Database ACID Test

- **Atomicity**
  - Either all changes take effect or none do
- **Consistency**
  - A transaction is allowed only if it follows owner- or system-defined integrity constraints
- **Isolation**
  - The results of the transaction are not visible until the transaction is complete
- **Durability**
  - The results of a completed transaction are permanent

Database Issues

Security Mechanisms and Issues

- Concurrency problems
- Checkpoints
- Trusted front end
- Aggregation
- Inference
- Views

Effects of Concurrency

- Database performs many transactions concurrently
- “Double Update” occurs when two programs access the same element simultaneously
- “Dead lock” condition can occur:
  - Two processes waiting for each other to release resources
  - Concurrent transactions are not risks; however, the results can be a threat if the transactions are not resolved properly:
    - Denial of service
    - Loss of data integrity

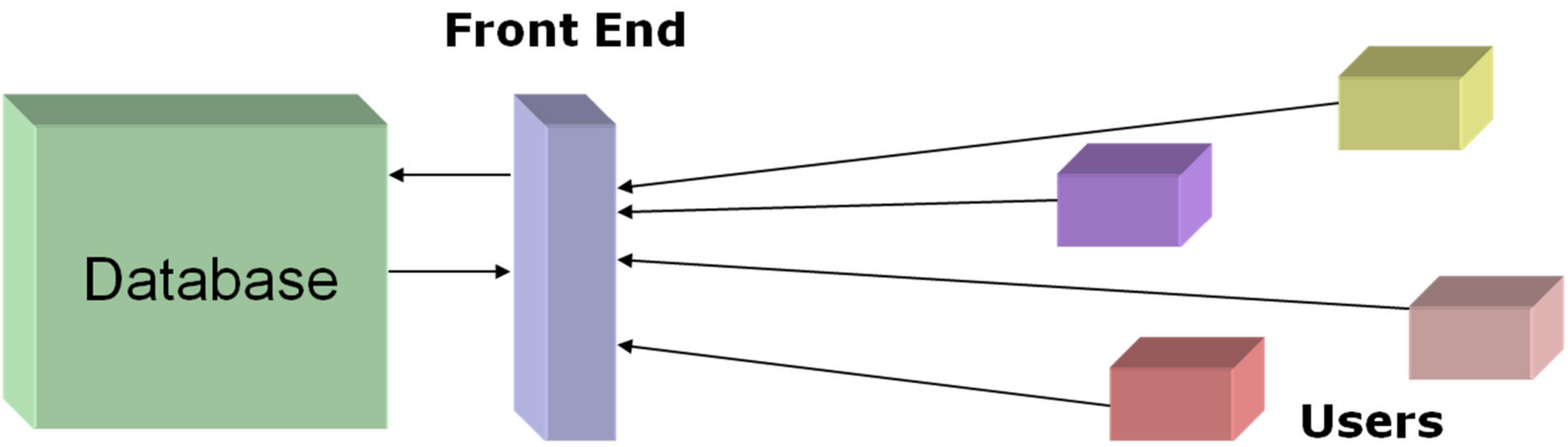
Database Security Mechanisms

- Database locks the data until one program is done
- Commit – completes a transaction and executes all changes that were recently entered
- Rollback – changes are cancelled, and the database returns to its state prior to this transaction
- Checkpoint (or “Savepoint”) – System periodically saves data; if an error is detected, the system can return to a known reliable state

Add-on Security Example

Trusted front-end

- Adds multilevel security to a database
- Another control between subjects and objects
- Used to retrofit security to a database



Database Security Attacks

Aggregation

- Act of combining information from separate sources
- This combination forms new information, to which the subject does not have authorized access
- The combined information has a sensitivity that is greater than the individual parts

Inference

- Inference is the ability to derive additional information from learned facts about a particular system
- Countermeasures: cell suppression, partitioning, insertion of noise data, polyinstantiation

Controlling DB Access

Content-dependent

- Controlling access based on contents of the object
- Query results depend upon the values within tables

Context-dependent

- Access decision based on the context of the request
- Mechanism required to keep track of who accessed and in what sequence

Database Integrity

Entity Integrity

- No primary key attribute can have a null value
- Primary key value must be unique
  - Unique identifier for a set of values

Referential Integrity

- When there is a relationship between two entities, those two entities need to actually exist
- No record can contain a reference to a key of a on-existing record

Distributed Computing

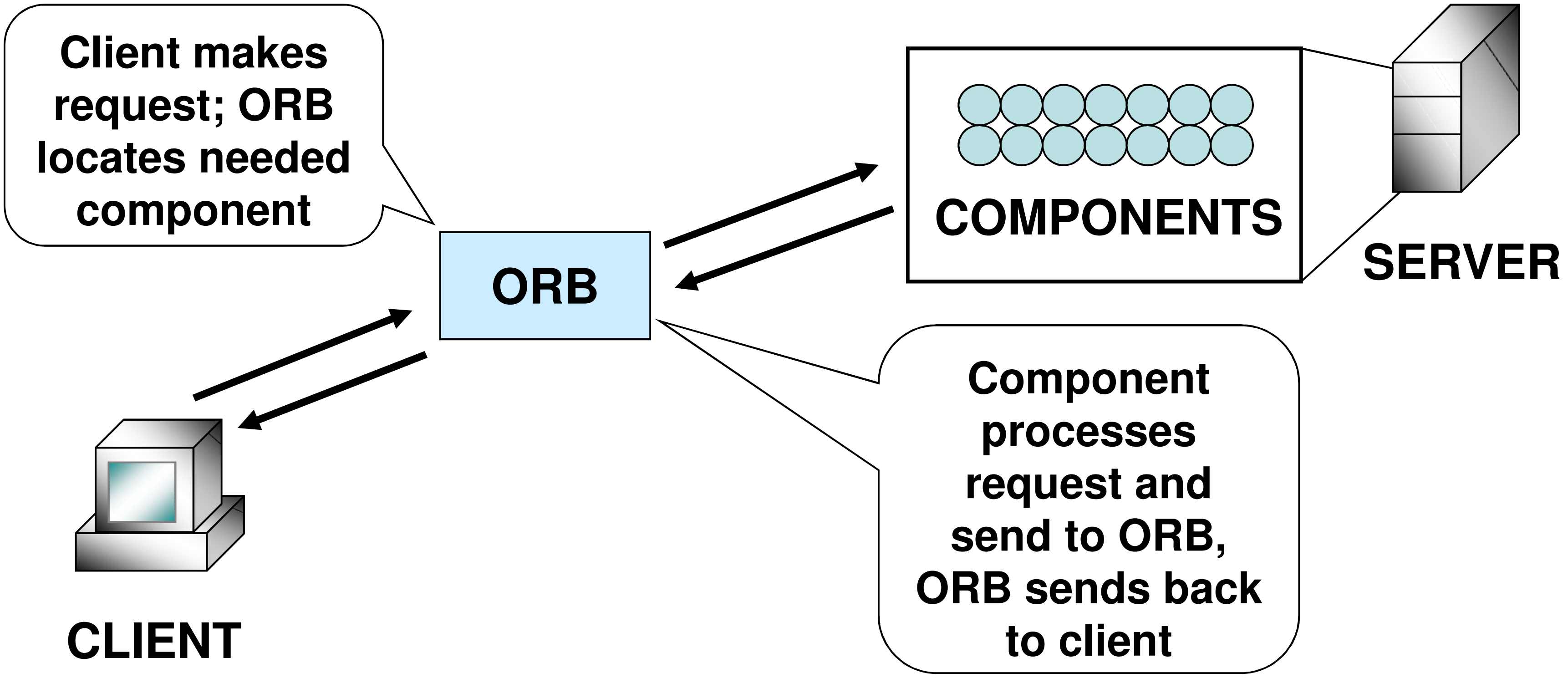
- **Data processing taking place on different systems**
  - Common Object Request Brokers
    - CORBA, ORB
  - Distributed Communication Standard
    - Component Object Model (COM), Distributed COM (DCOM)
    - Enterprise Java Bean (EJB)
    - Simple Object Access Protocol (SOAP)

Object Request Brokers (ORB)

- Manages all communication between components and allows them to interact in a heterogeneous and distributed environment
- ORBs act as locators and distributors of objects across networks
- Middleware for client/server relationships

Object Request Brokers

- ORBs



Client makes request; ORB locates needed component

ORB

COMPONENTS

SERVER

CLIENT

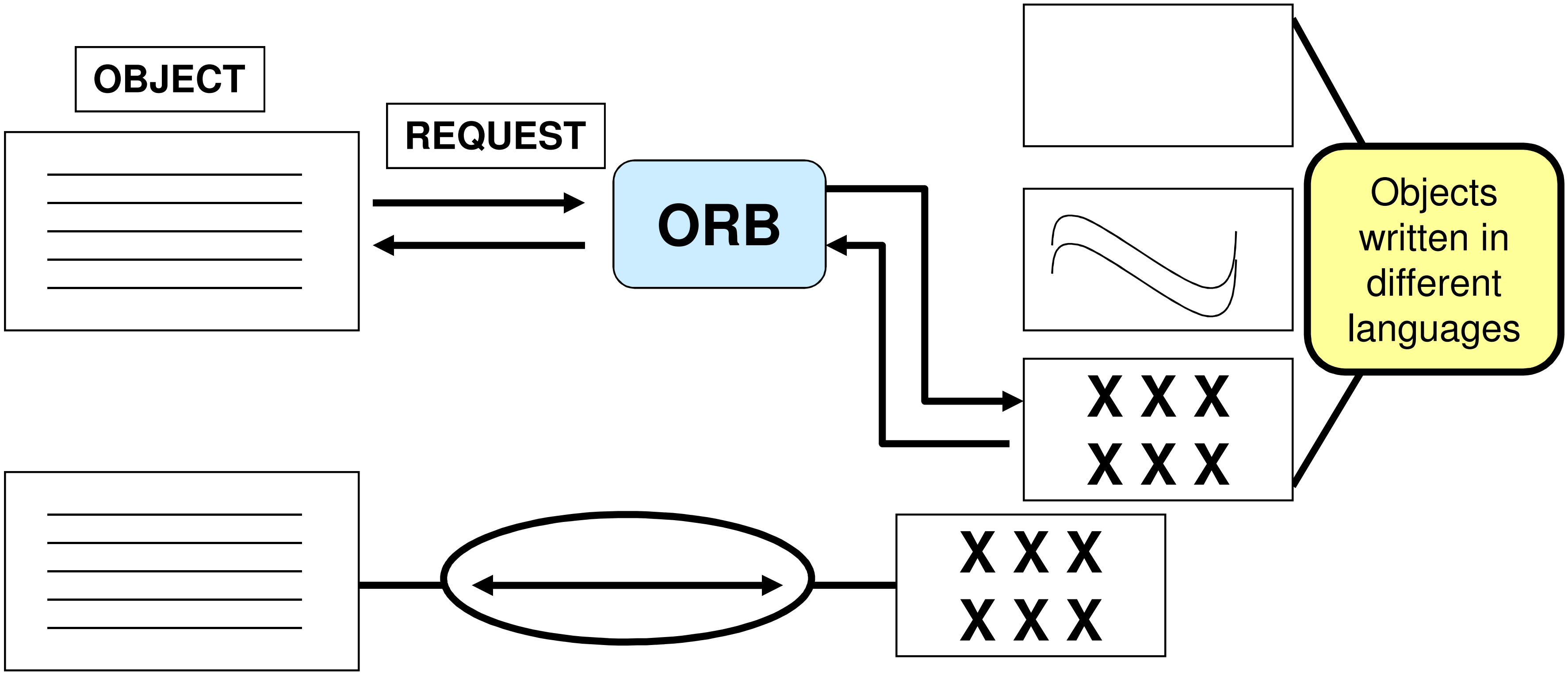
Component processes request and send to ORB, ORB sends back to client

CORBA

- Common Object Request Broker Architecture
- Standard developed by Object Management Group (OMG) which allows different applications written in different languages to communicate
- Standard set of interfaces and APIs for systems to use to communicate with different ORBs

Distributed Communication Architecture

CORBA



This works because of standardized interfaces

Component Object Model

- COM
  - Architecture to allow for simple inter-process communication between objects
  - Allows objects on one system to communicate
- DCOM (Distributed Component Object Model)
  - Allows for COM inter-process communication in a distributed form
  - Allows objects on different systems to interact
  - Works as middleware for distributed processing

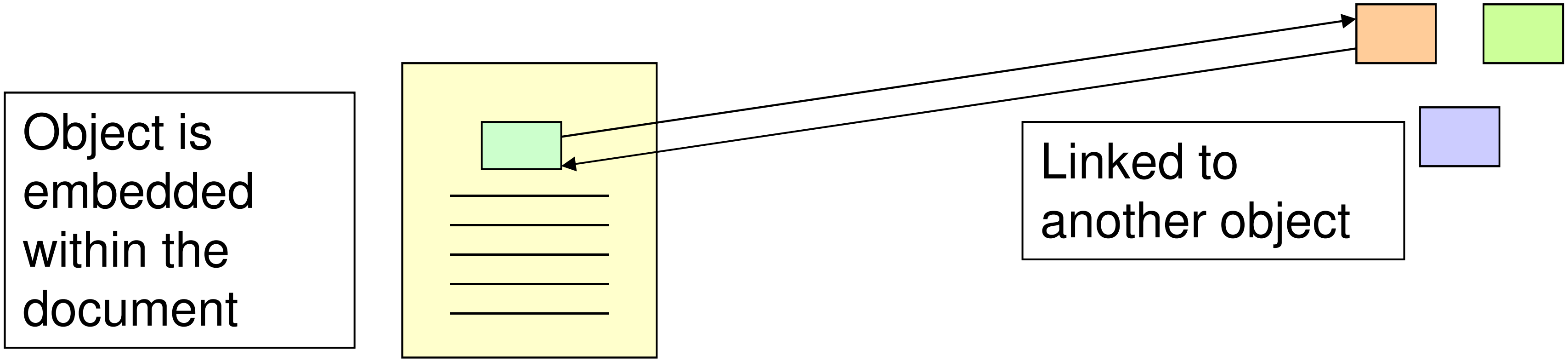
Linking Through COM

Object Linking and Embedding (OLE)

- Provides a way for objects to be shared on a local workstation and uses Component Object Module (COM) as its foundation base
- OLE allows objects to be embedded into documents (i.e., graphics, pictures, and spreadsheets)
- Linking is the capability for one program to call another
- The capability to put a piece of data inside a foreign program or document is embedding

Linking Through COM

- Object Linking and Embedding (OLE)



Object is embedded within the document

Linked to another object

Mobile Code

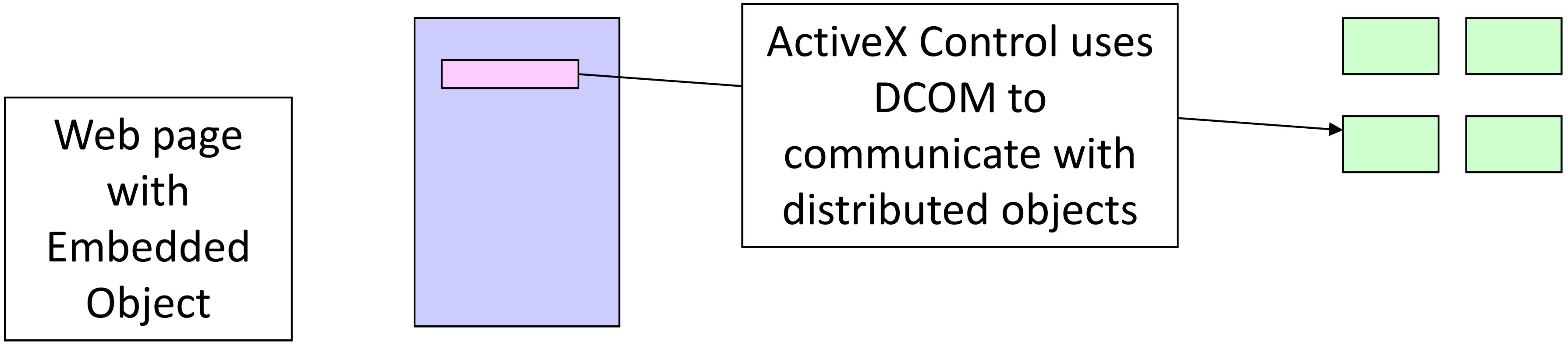
Active Content

- Documents or scripts that can carry out actions without user intervention
- Increase user functionality through a browser
- Java applets, JavaScript, ActiveX controls, macros, and executable e-mail attachments
- Extends capabilities and functionality, but can introduce more risks
- Trojan horses, backdoors, viruses, malicious code, and worms all take advantage of Active Content

World Wide Web OLE

ActiveX

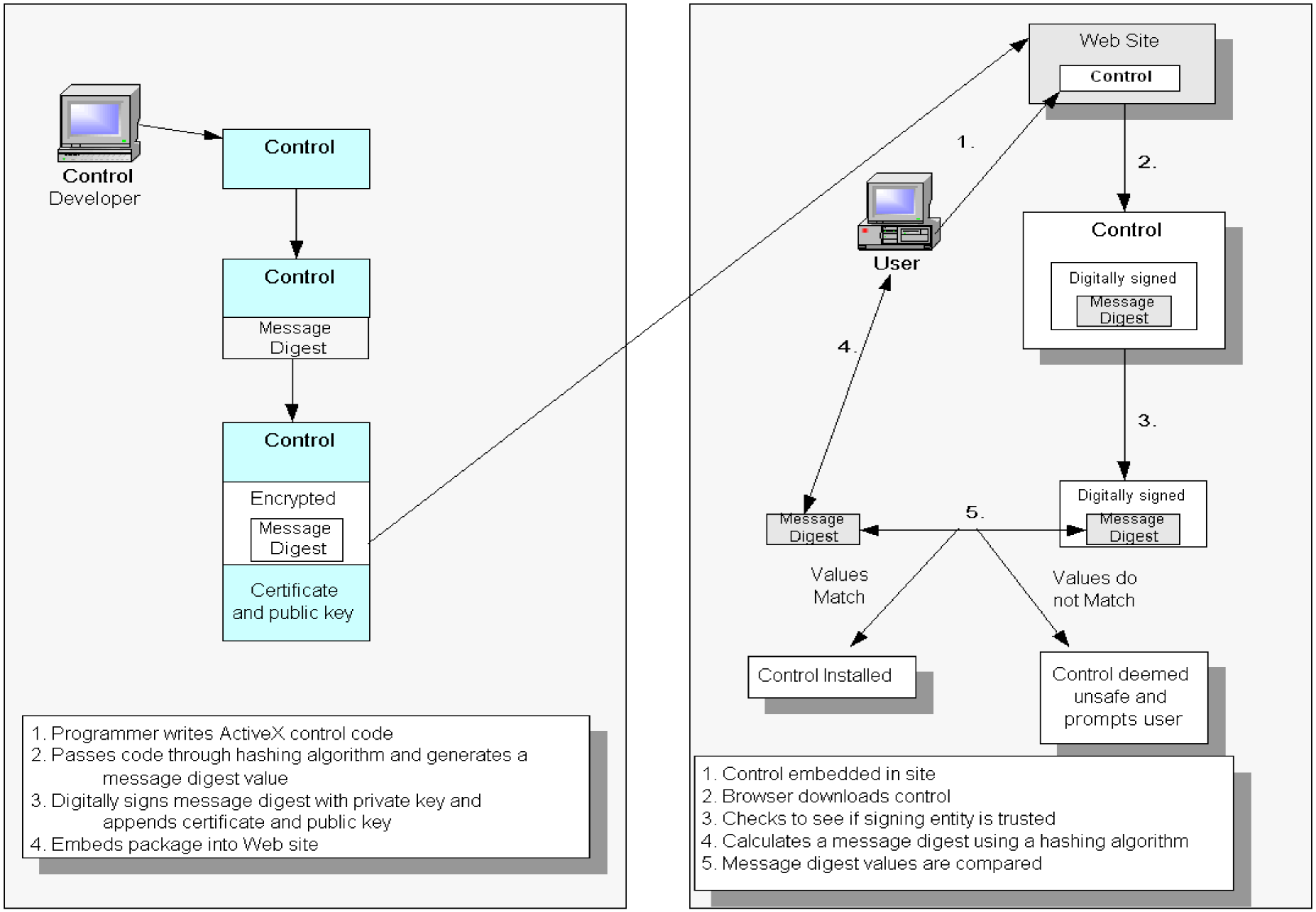
- Platform specific and language independent
- ActiveX Controls communicate to objects using COM services
- Outgrowth of OLE and COM technologies for the World Wide Web


Web page with Embedded Object
ActiveX Control uses DCOM to communicate with distributed objects

ActiveX Controls

- ActiveX is Microsoft technology that is used to write controls that Internet users can download to increase their functionality and Internet experience
- Extension of Object Linking and Embedding
- Security scheme: inform the user of the origin of the component who will decide if it should be trusted
- Security relies on digital certificates and trusting certificate authorities

Digital Signing of ActiveX Controls



1. Programmer writes ActiveX control code
2. Passes code through hashing algorithm and generates a message digest value
3. Digitally signs message digest with private key and appends certificate and public key
4. Embeds package into Web site

1. Control embedded in site
2. Browser downloads control
3. Checks to see if signing entity is trusted
4. Calculates a message digest using a hashing algorithm
5. Message digest values are compared

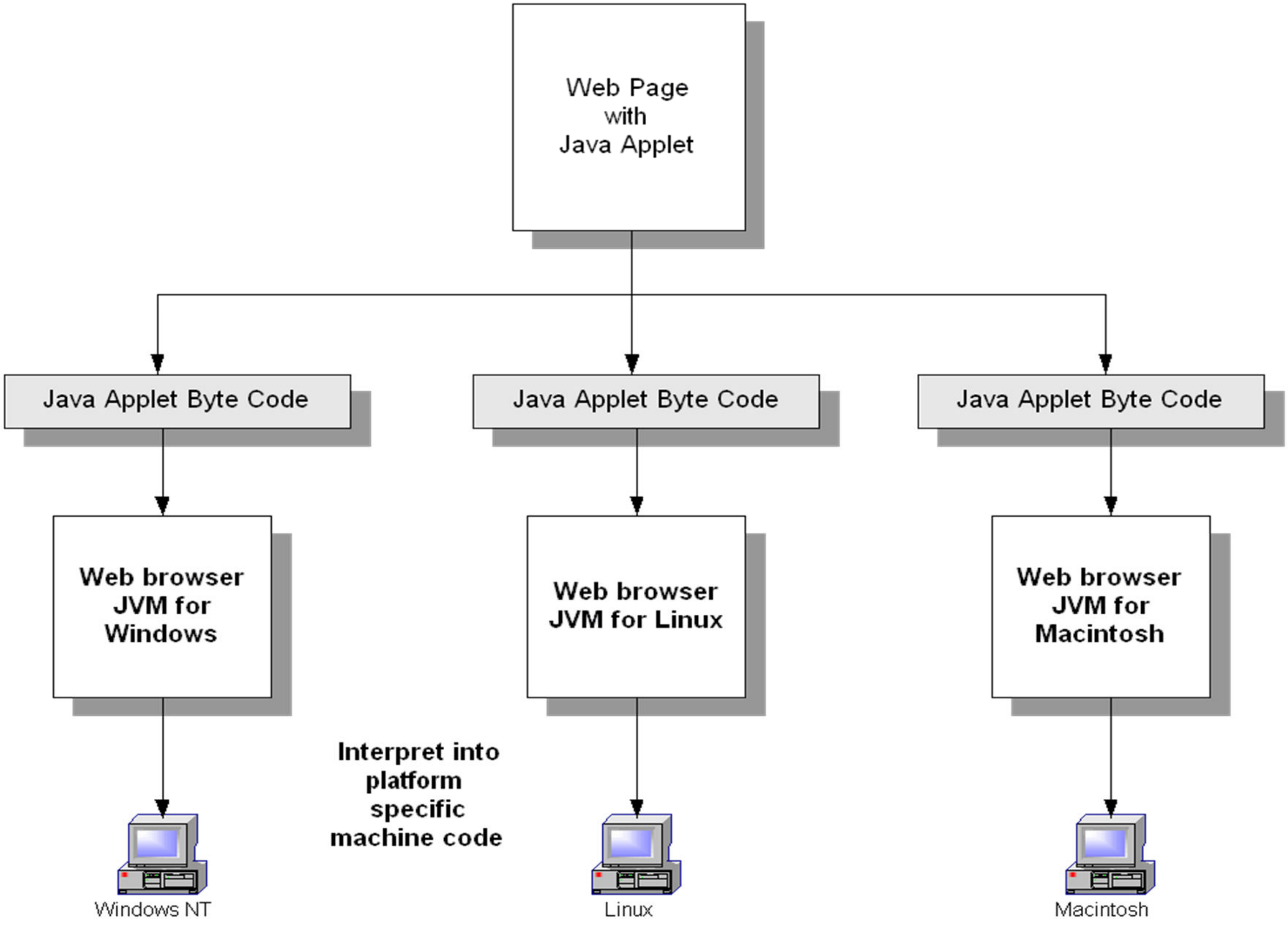
Java

- Language developed by Sun
- Object-oriented, platform-independent programming language
- Runs on top of the Java Virtual Machine, which allows applets to run on several different platforms
- Uses byte-code instead of being compiled directly into machine code

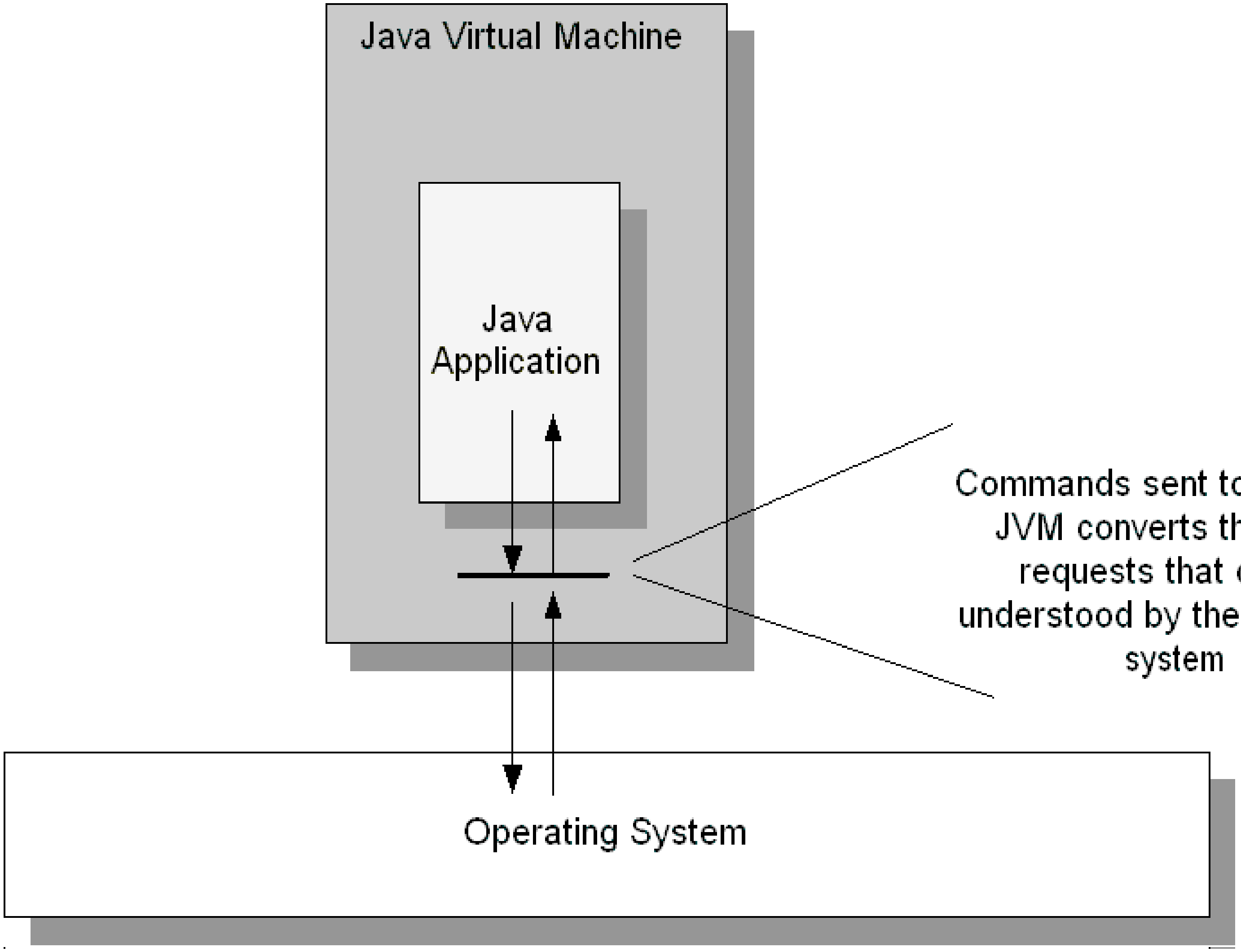
Java Applets

- Small Java programs that run within a user's browser
- Mobile code that is restricted from accessing local disks and network connections by being contained in a "sandbox"
- Several bugs within Java allow malicious code to escape the sandbox and perform malicious acts

Java and Bytecode

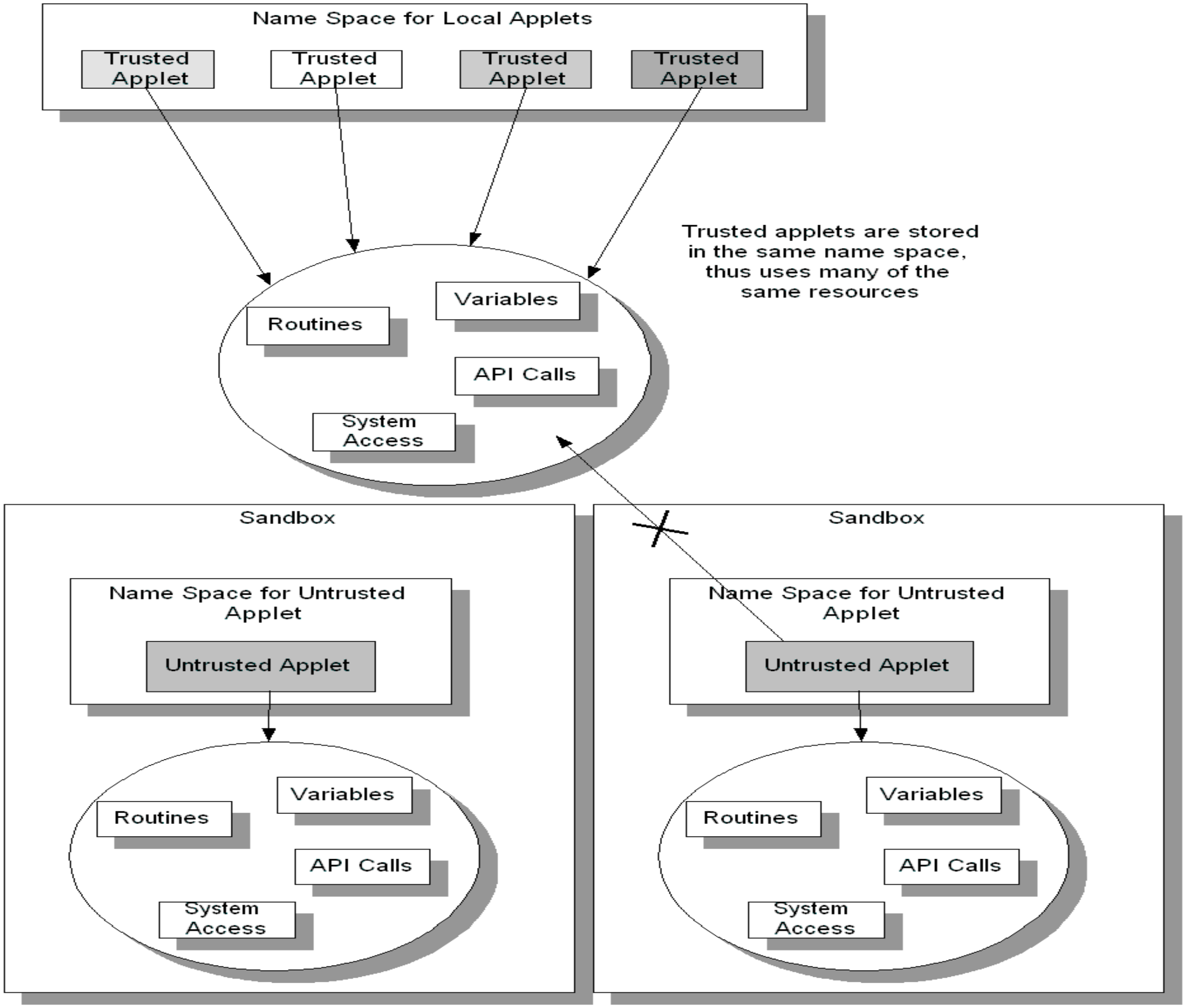


Java Virtual Machine



Commands sent to JVM and JVM converts them into requests that can be understood by the operating system

Java Security

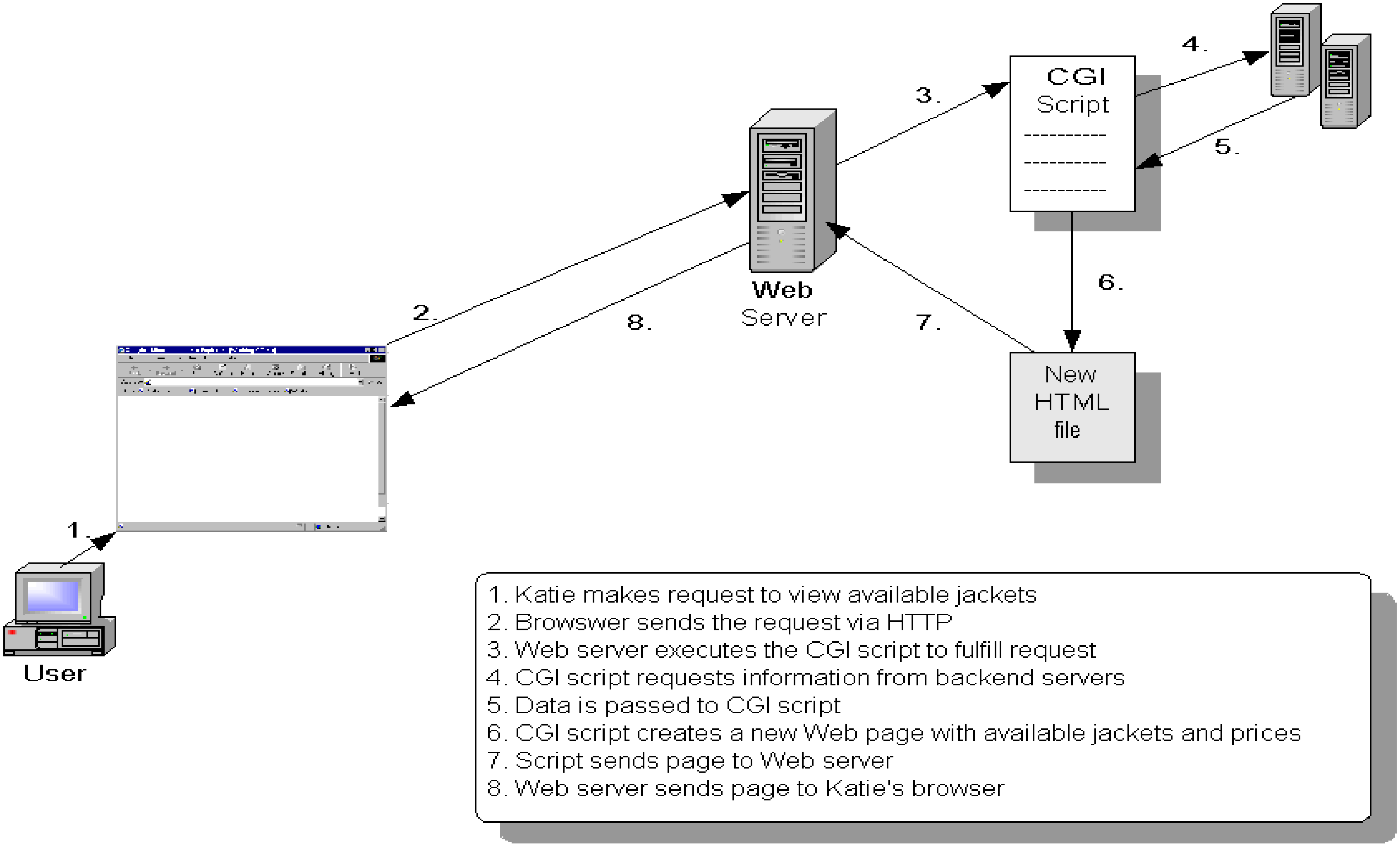


Trusted applets are stored in the same name space, thus uses many of the same resources

Common Gateway Interface

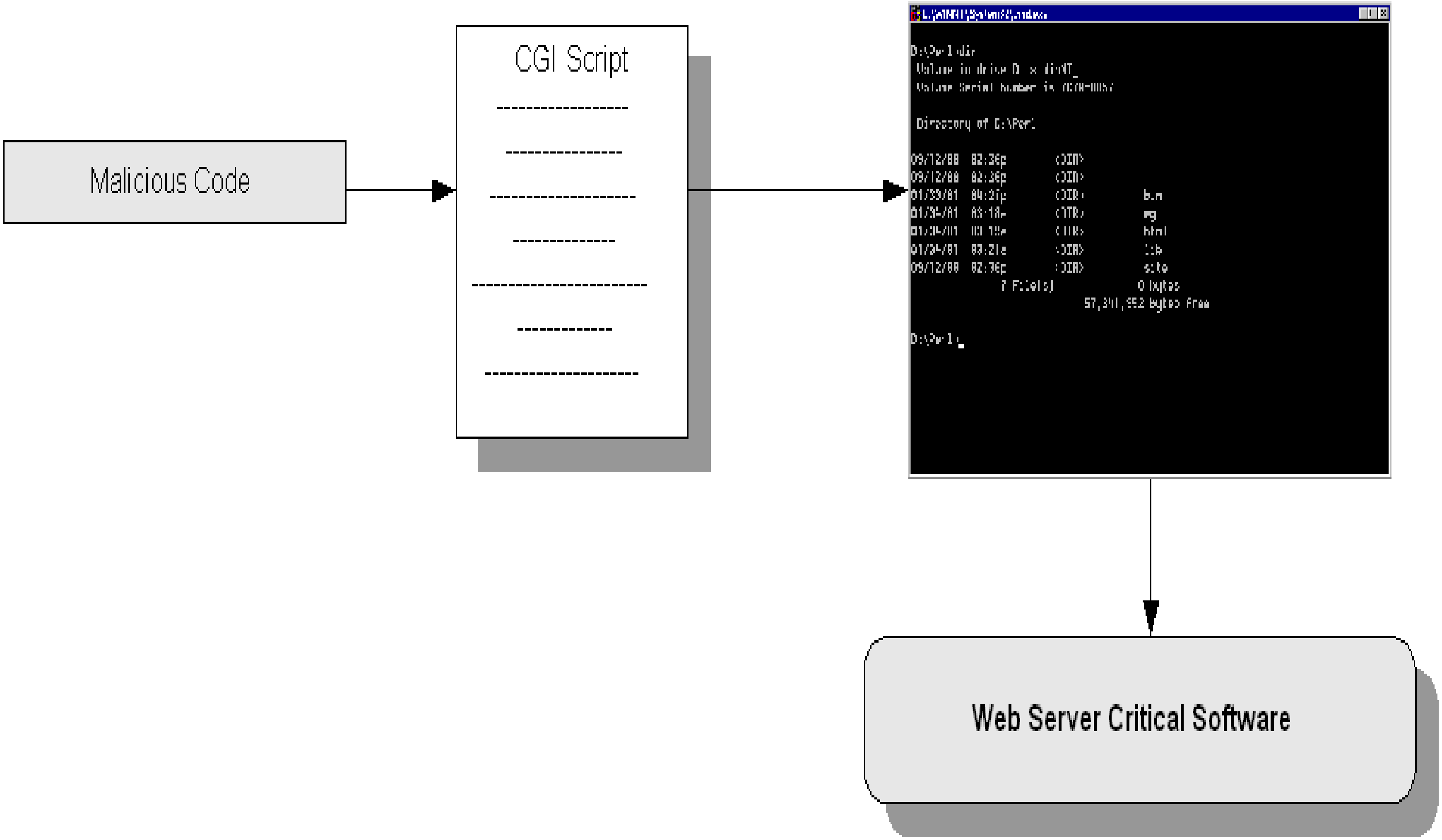
- CGI is a method of manipulating data passed into a Website
- CGI script resides on web server, not the browser
- Allows for interactive Websites that process user input
- Security risks are that they use an array of low level system commands that can be exploited
- Scripts are interpreted and not compiled
  - there is more risk of it being modified
- The CGI scripts should check for illegal commands before processing

How CGI Scripts Work



1. Katie makes request to view available jackets
2. Browswer sends the request via HTTP
3. Web server executes the CGI script to fulfill request
4. CGI script requests information from backend servers
5. Data is passed to CGI script
6. CGI script creates a new Web page with available jackets and prices
7. Script sends page to Web server
8. Web server sends page to Katie's browser

Busting Out of a Script to a Shell



```
D:\Perl>dir
 Volume in drive D is WinNT_
 Volume Serial Number is 0C78-0057

 Directory of D:\Perl

09/12/00  02:36p        <DIR>
09/12/00  02:36p        <DIR>
01/30/01  04:27p        <DIR>          bin
01/30/01  03:18p        <DIR>          eg
01/30/01  03:19p        <DIR>          html
01/30/01  03:21p        <DIR>          lib
09/12/00  02:36p        <DIR>          site
               7 File(s)              0 bytes
                             57,341,952 bytes free

D:\Perl>
```

Web Applications

Input Validation

- Web forms can be a source of attack if special characters or code is allowed to be entered
- Developers must check input parameters and perform “input cleansing” before the form is submitted to the server

SQL Injection

- When SQL statements are entered into a form
- Allows attacker to manipulate the database tier of a web application

XSS -Cross Site Scripting

- JavaScript is entered through a form or url parameter
- Comes back around to the victims browser and executes
- Commonly found in the URLs within phishing emails

Cookies

- **HTTP cannot track Web users and their actions**
- **Cookies can allow sites to "remember" users' visits and choices**
  - Tokens within HTTP requests and responses
- **Can be per session**
  - Cookie expires when browser is closed
- **Can be persistent**
  - Reside as a text file on user's hard drive
- **Privacy issues:**
  - Collect personal information about users
  - Can contain sensitive information that an attacker may try to access

Malware

- Virus
- Worm
- Logic Bomb
- Trojan Horse
- Other Attack Types

Malicious Code Detection

Ways to detect malicious code:

- File size increase
- Many unexpected disk accesses
- Change in update or modified timestamp
- Sudden decrease of hard drive space
- Calculating checksums on system files
- Unexpected and strange activity by applications

Malware Types

Virus

- A virus is a program that searches out other programs and infects them by embedding a copy of itself
- When the infected program with the embedded virus is executed, the infection is propagated

Types of Viruses

- Macro Virus: easy to create because of the simplicity of the macro languages
- Boot Sector Virus: malicious code inserted into the disk boot sector
- Compression Virus: when decompressed, it initializes and attacks the host system
- Stealth Virus: hides its footprints and changes that it has made
- Polymorphic Virus: makes copies and changes the copies in some way – uses a mutation engine
- Multi-Partite Virus: infects both boot sector and hard drive (the file system)

More Malware

Worms

- Can reproduce on their own – different than virus
- Self contained programs

Logic Bomb

- An event triggers the execution of specific code

Trojan Horse

- Program disguised as another program
- Useful program that contains hidden code exploiting the authorization of process to violate security

Attack Types

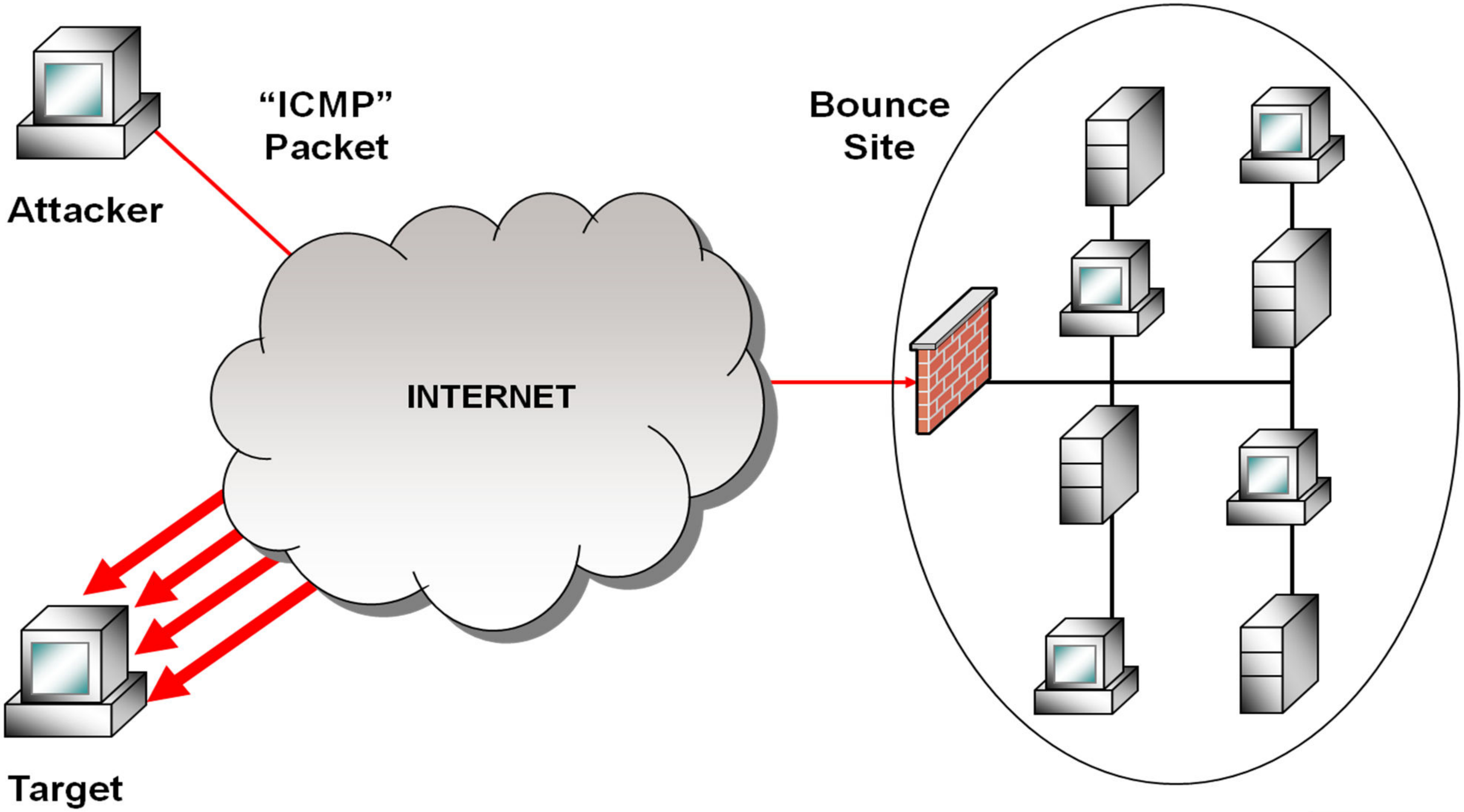
Smurf

- ICMP ECHO broadcast is sent to a network with a spoofed address
- All systems on network segment send ICMP ECHO REPLY packets to victim
- DoS attack

Fraggle

- Same as Smurf
- Uses UDP instead of ICMP protocol

Smurf Attack

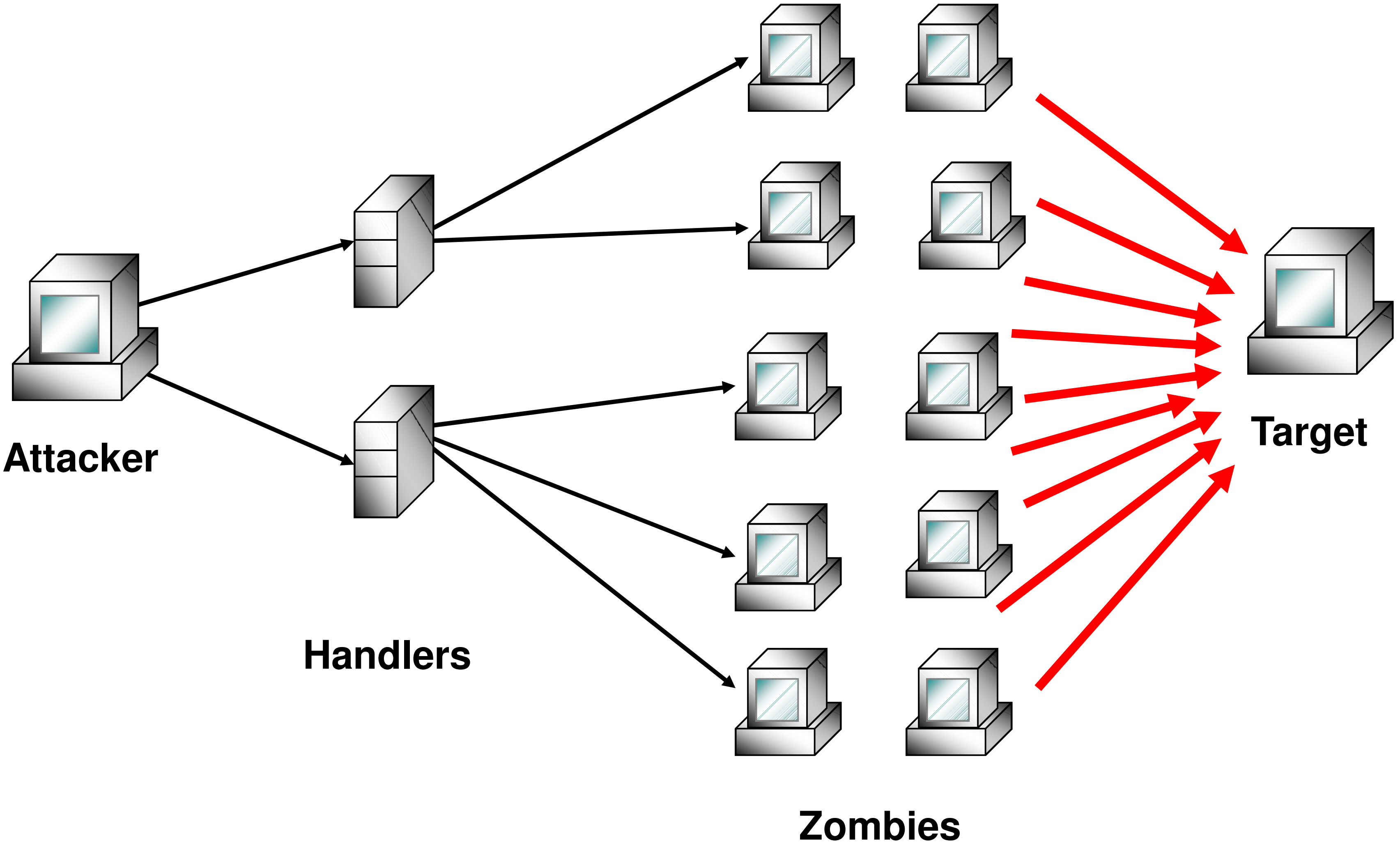


DDoS Attack

Denial of Service

- Committing resources on a computer so that it cannot respond to valid requests
- Can be distributed and amplified by using other systems to commit the attack – Distributed Denial of Service (DDoS)
  - Masters and zombies
  - Handler systems

DDoS – Handlers and Zombies



DDoS Countermeasures

- Drop all ICMP packets originating from the Internet
- Drop any requests to broadcast addresses
- Ingress filtering: do not allow packets in with internal source addresses
- Egress filtering: do not allow packets to leave with external source addresses

Timing Attacks

- **Between the Lines Entry**
  - Tap into an inactive communication line of an active user
- **NAK/ACK Attack**
  - Accessing a system when it is in the negotiation stages of setting up communication
  - Some systems do not respond to a negative ACK properly
- **Line Disconnect Attack**
  - Picking up a user’s session before the system terminates it

Other Attack Types

Salami

- Taking insignificant pieces of data, say pennies, from a user’s bank account and moving them to the attacker’s bank account.

Session Hijacking and Man in the Middle (MITM)

- Capturing a session just prior to the user signing off an attacker could use the existing credentials to continue accessing and altering data

Agenda - 3

- Understand and apply security in the software development life cycle
- Understand the environment and security controls
- **Assess the effectiveness of software security**

Artificial Intelligence

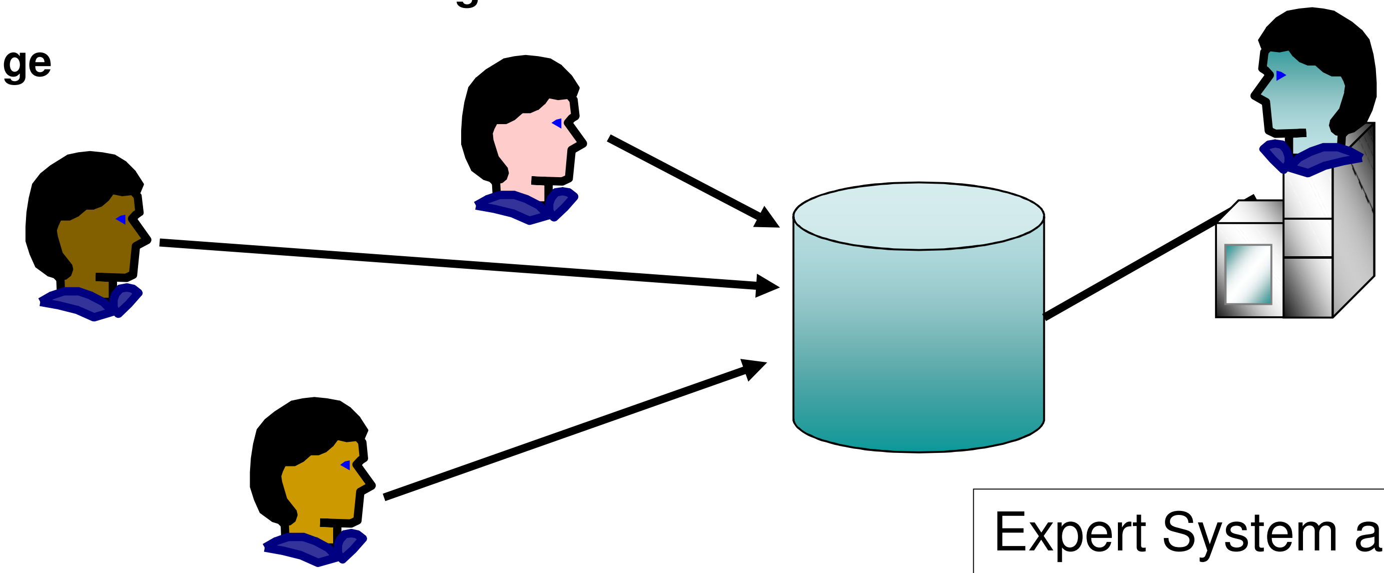
- Expert Systems
- Artificial Neural Networks

Artificial Intelligence – Expert Systems

- A computer program containing a knowledge base and set of algorithms and rules
- Used to infer new facts from existing knowledge and incoming data
- Uses artificial intelligence to mimic human logic
- Data is collected from human experts and stored in a database
  - Collected data is used for problem resolving
  - Best when working with a limited domain of data

Artificial Intelligence – Expert Systems

Example:



Knowledge System Builder

Knowledge Engineer

Subject Matter Expert

Expert System acts as the users' consultant

Artificial Intelligence – Inference Engine

Rule-based Programming

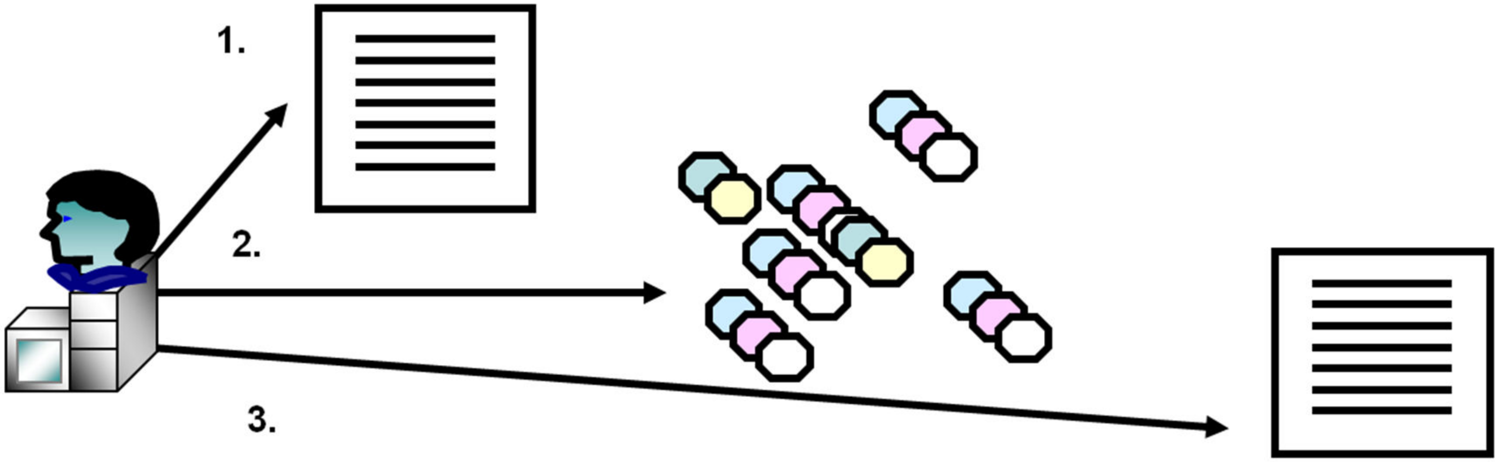
- Rules are based on “if – then” logic units
- This specifies a set of actions to be performed for a given situation
- Uses human logic

Inference Engine performs pattern matching

- The engine matches facts against patterns and determines which rules are applicable
- The engine can find patterns not obvious to humans, and then it can apply human logic

The Process

- **The Expert System will:**
  1. Use rule-based programming to identify any recognizable patterns
  2. Identify a pattern
  3. Apply "human logic" to the pattern



Artificial Neural Networks

- Electronic model based on the neural structure of the brain
- ANN systems have:
  - The ability to remember and learn from new experiences
  - The capacity to generalize
- Decisions by neural networks are only as good as the experiences they are given

Bringing Things Together

- Devices, software products, and procedures must control access to these vulnerabilities
- Proper coding and testing habits would decrease vulnerabilities
- Insert security into each step of a development project
- Object-oriented programming has many benefits in efficiency and security
- Databases have different models, security issues, and countermeasures
- There are several types of malware and attacks
- All processing takes place in some type of software