Securing Distributed Applications Oriented on Very Large Data Sets

Dr. T V Gopal
Professor &
Chairman, CSI Division II [Software]
Department of Computer Science and Engineering, CEG
Anna University, Chennai
E-mail: gopal@annauniv.edu
http://www.csi-india.org/web/csi/division2

OWASP
Saturday, 11 August 2012

The OWASP Foundation
http://www.owasp.org

Securing Distributed Applications Oriented on Very Large Data Sets

- Aspects of Security – Distributed Applications
- Very Large Data Sets – Distributed Applications
- Very Large Datasets Oriented Project – Risks
- Architecture of the Application
- **Threat Control Measures**
- Good Strategies
- Security Services
- **Application Security – Some Existing Solutions**

Aspects of Security – Distributed Applications

- communication between the layers of the distributed application
- security processes viewed as resources consuming
- the account's policy and how access privileges are managed
- the authentication process as an important aspect for the entire security level.

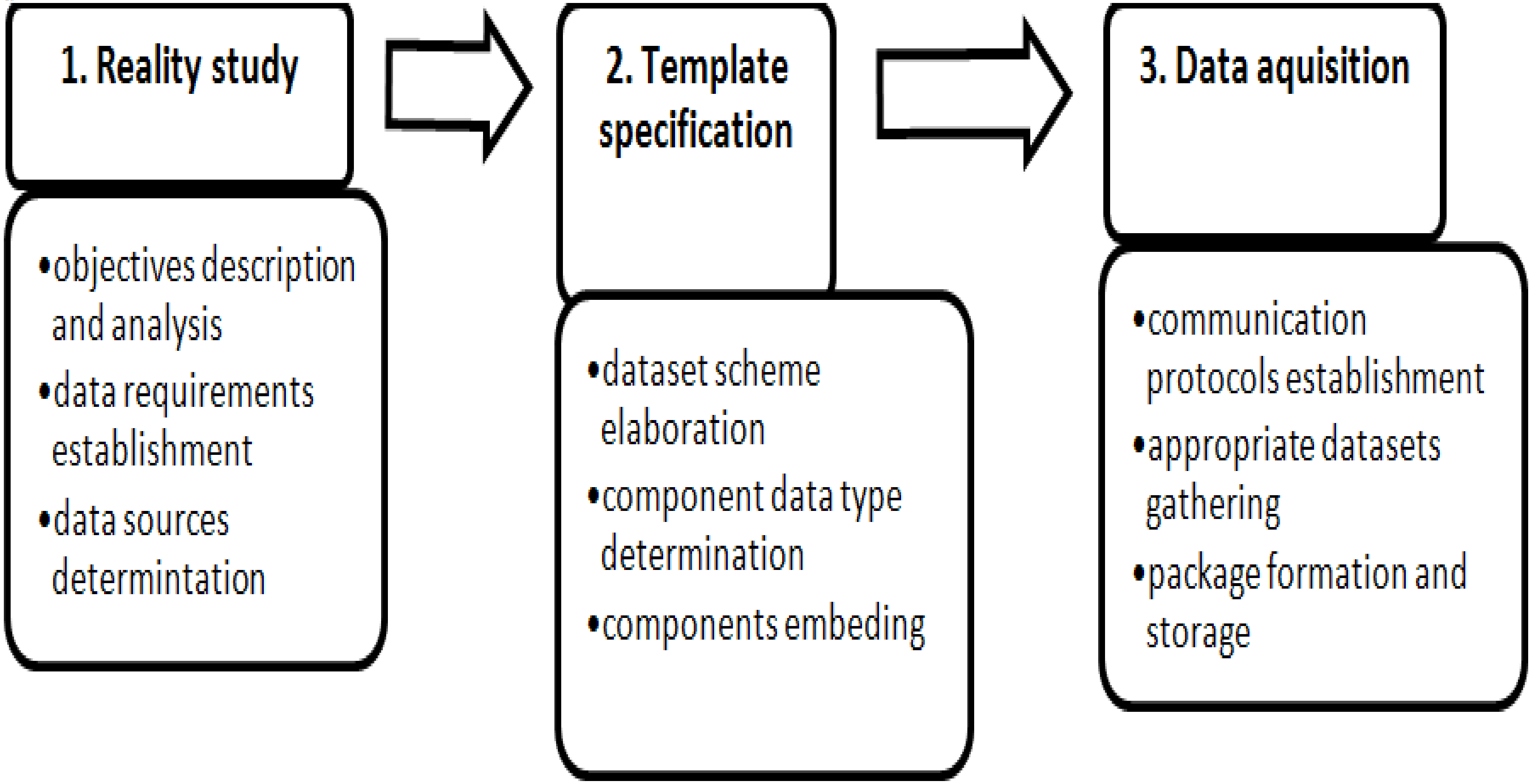
Very Large Data Sets – Distributed Applications

- Telecommunications operators record each call or message within the network for a period of six months
- Internet and e-mail providers record accessed sites for each IP address in its administration, together with the exact date of access and data about each email message
- Government keep track of different payments for millions of people
- National providers of utilities – gas, electricity and so on – process hundreds of millions of annual consumer bills
- Online search engines integrate content management of billions of sites
- The amount of data stored is quickly growing due to the pervasive presence of sensors.

Factors Affecting the Database Size

- Type and complexity of the information contained within a certain field
- Number of record features or fields
- Number of table records
- Number of database tables
- Number and complexity of integrity constraints within a table
- Number and complexity of relationships between tables
- Number and complexity of auxiliary files: indexes, stored procedures and so on

Design of Very Large Data Sets


1. Reality study
•objectives description and analysis
•data requirements establishment
•data sources determintation
2. Template specification
•dataset scheme elaboration
•component data type determination
•components embeding
3. Data aquisition
•communication protocols establishment
•appropriate datasets gathering
•package formation and storage

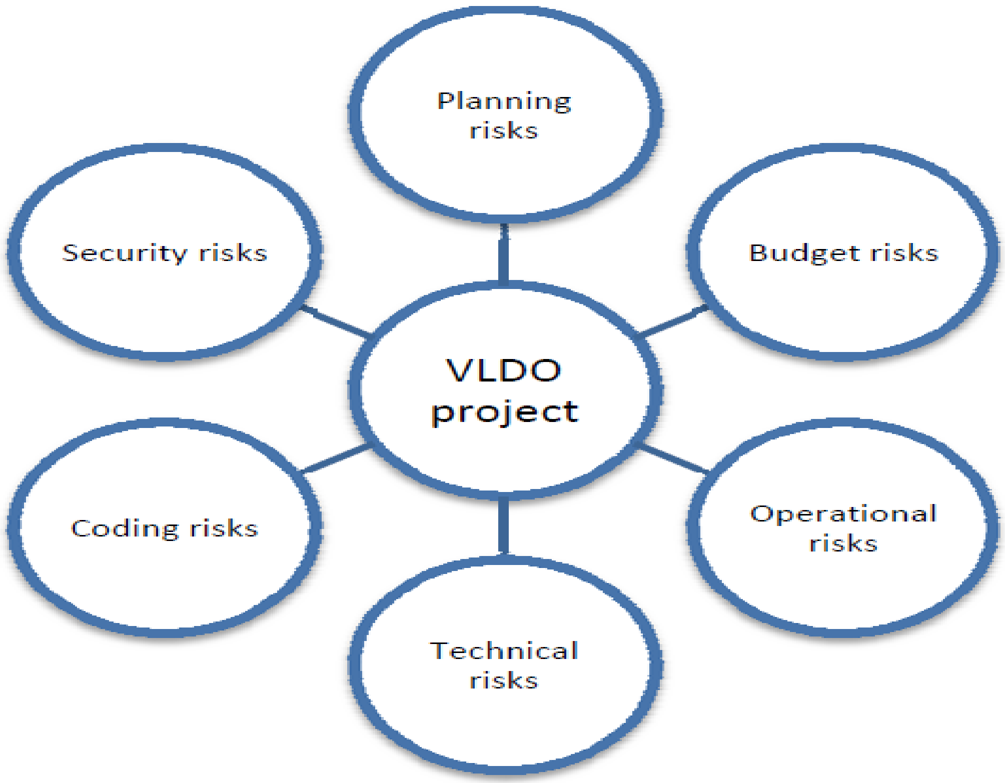
Pragmatics of Very Large Datasets

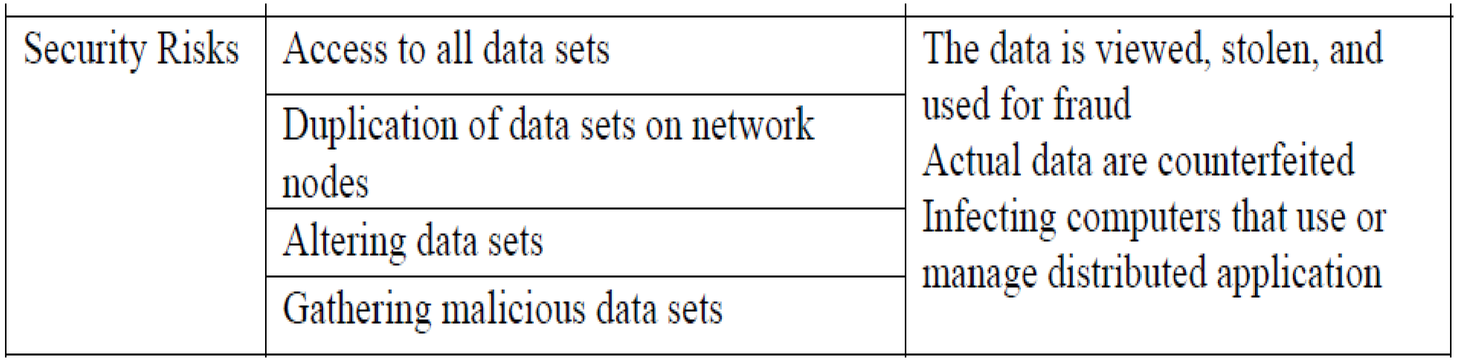
- Very large databases operate with virtual databases obtained by local databases concatenation
- Databases are continuously updated by adding information, with the exact identification of the moment, place, the operation and the person which generated the update process
- The level of homogeneity in databases is extremely low because of the diversity of data: text, images from photography, images from scanning, animated sequences
- Database are designed so that the target group elements access only the feature that defines the personal objective, and generate access procedures in other related databases in order to solve the problem
- Databases require access rights for application management so that the confidentiality terms are met

Characteristics of These Applications

- Number of Users at any given time does not make an impact
- User behavior does not affect the application structure – the degree to which the user understands the application does not matter.
- Quality of Input does not affect processing flow – the processing goes on with warning of possible errors

Very Large Datasets Oriented Project - Risks





Security Risks	Access to all data sets	The data is viewed, stolen, and used for fraud Actual data are counterfeited Infecting computers that use or manage distributed application
	Duplication of data sets on network nodes	
	Altering data sets	
	Gathering malicious data sets	

Probability * Impact Matrix

Probability	Frequently	░	▒	▓	▓	▓
	Probably	░	▒	▓	▓	▓
	Occasionally	░	░	▒	▓	▓
	Rarely	░	░	▒	▓	▓
	Unlikely	░	░	░	▒	▓
		Negligible	Minor	Major	Critical	Catastrophic
		Impact				

░	– low risk;
▒	– moderate risk;
▓	– high risk.

Systems are often developed without security in mind.

Vulnerability 1: Authorization

- Credential / Session Prediction is a method of hijacking or impersonating a user.
- Insufficient Authorization permits access to sensitive content or functionality that should require more access control restrictions.
- Insufficient Session Expiration permits an attacker to reuse old session credentials or session IDs for authorization.
- Session Fixation attacks force a user’s session ID to an explicit value.

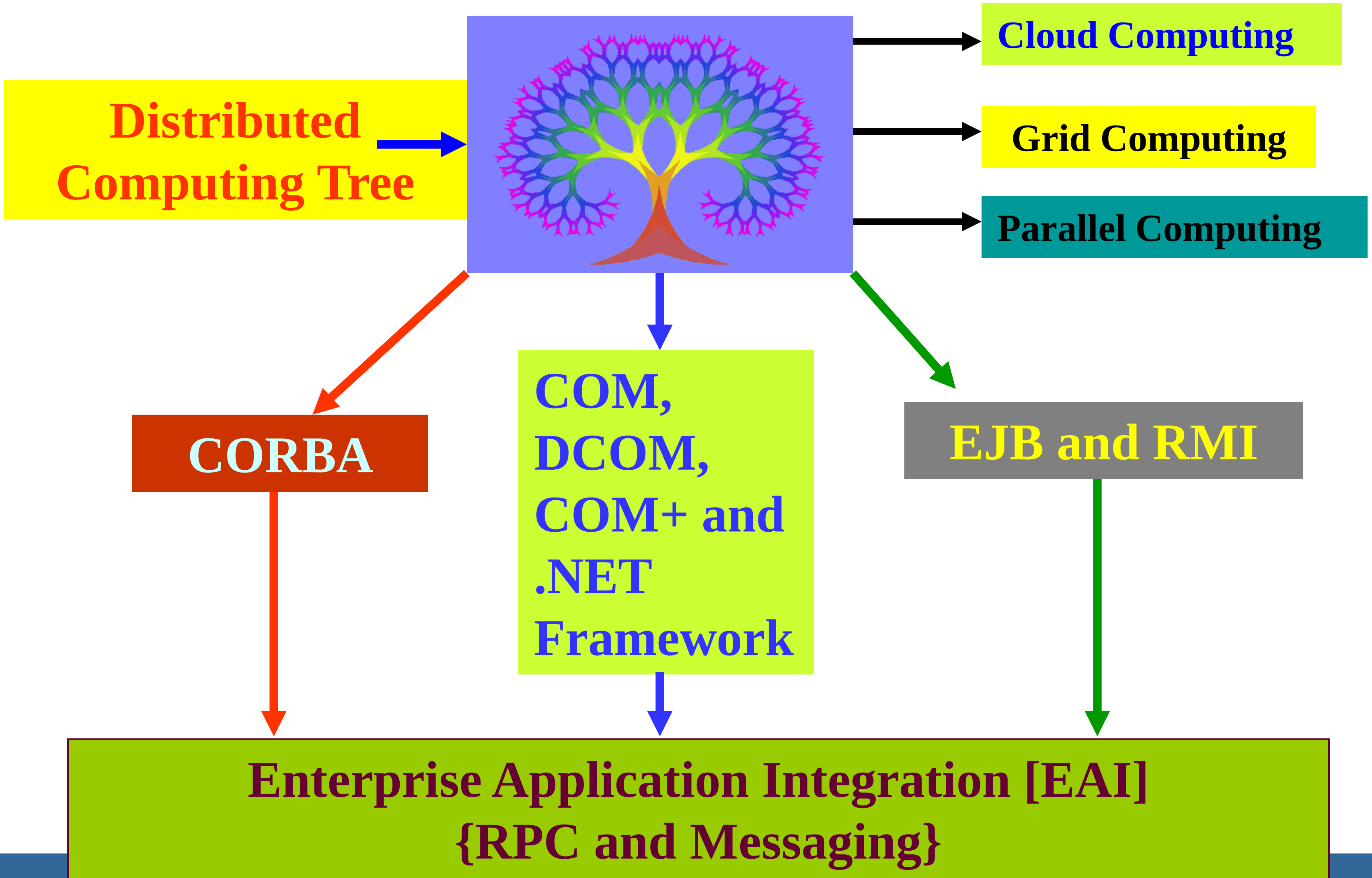
Vulnerability 2: Command Execution

- Buffer Overflow attacks alter the flow of an application by overwriting parts of memory.
- Format String Attack alters the flow of an application by using string formatting library features to access other memory space.
- LDAP Injection attacks exploit web sites by constructing LDAP statements from user-supplied input.
- OS Commanding executes operating system commands on a web site by manipulating application input.
- SQL Injection constructs illegal SQL statements on a web site application from user-supplied input.
- SSI Injection (also called Server-side Include) sends code into a web application, which is later executed locally by the web server.
- XPath Injection constructs XPath queries from user-supplied input.

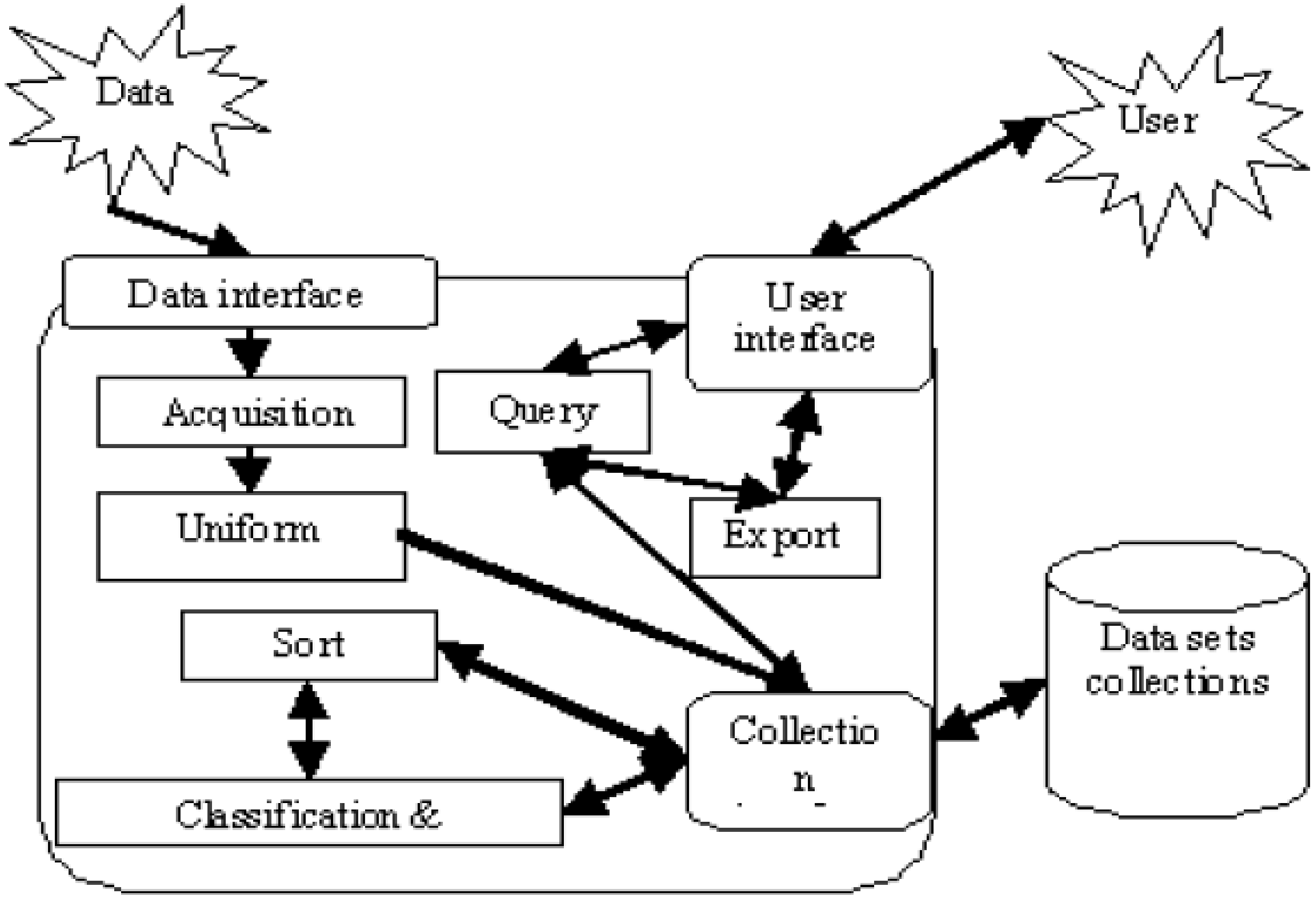
Vulnerability 3: Information Disclosure

- Directory Indexing is an automatic directory listing / indexing web server function that shows all files in a requested directory if the normal base file is not present.
- Information Leakage occurs when a web site reveals sensitive data such as developer comments or error messages, which may aid an attacker in exploiting the system.
- Path Traversal forces access to files, directories and commands that potentially reside outside the web document root directory.
- Predictable Resource Location uncovers hidden web site content and functionality.

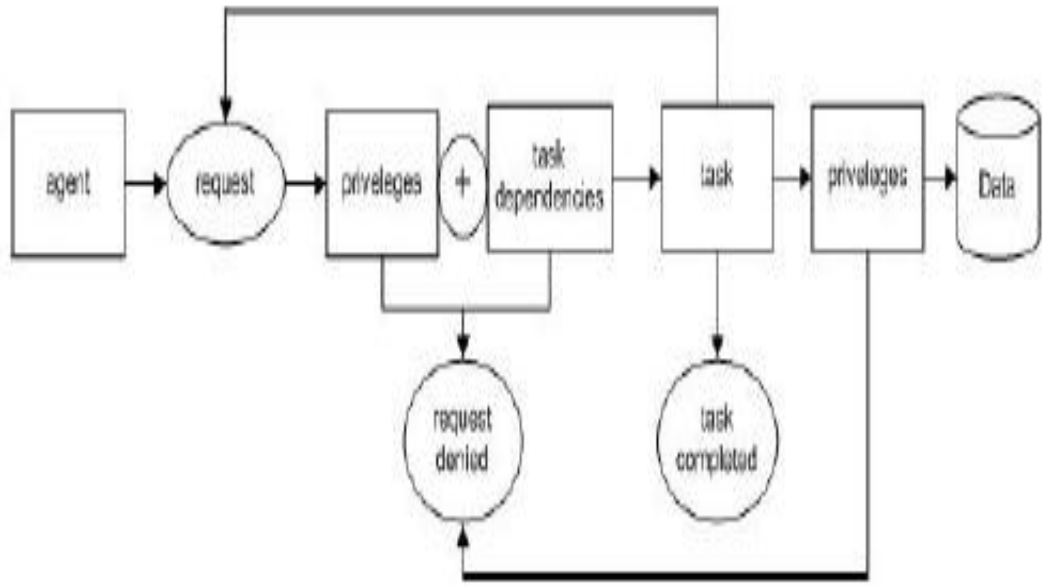
Enterprise Application Integration [EAI]

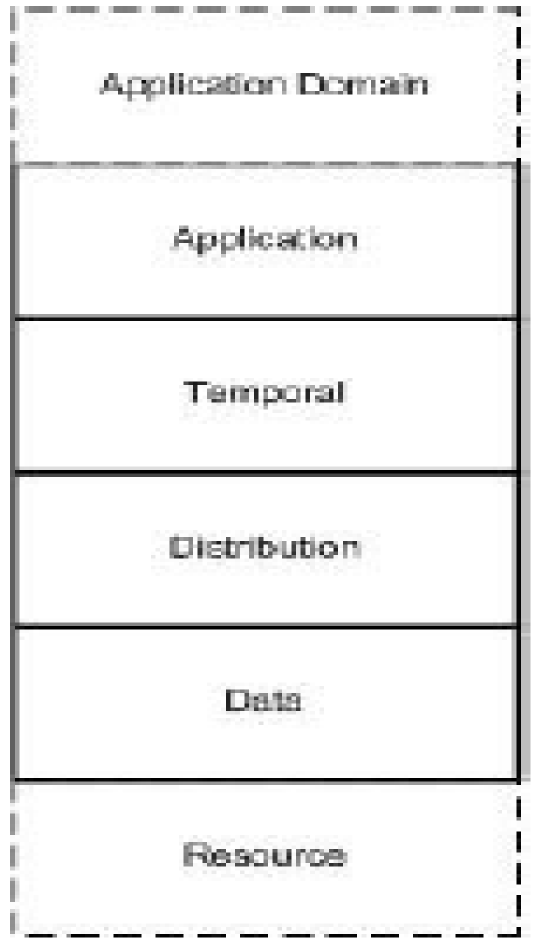

Distributed Computing Tree
Cloud Computing
Grid Computing
Parallel Computing
CORBA
COM, DCOM, COM+ and .NET Framework
EJB and RMI
Enterprise Application Integration [EAI] {RPC and Messaging}

Architecture of the Application

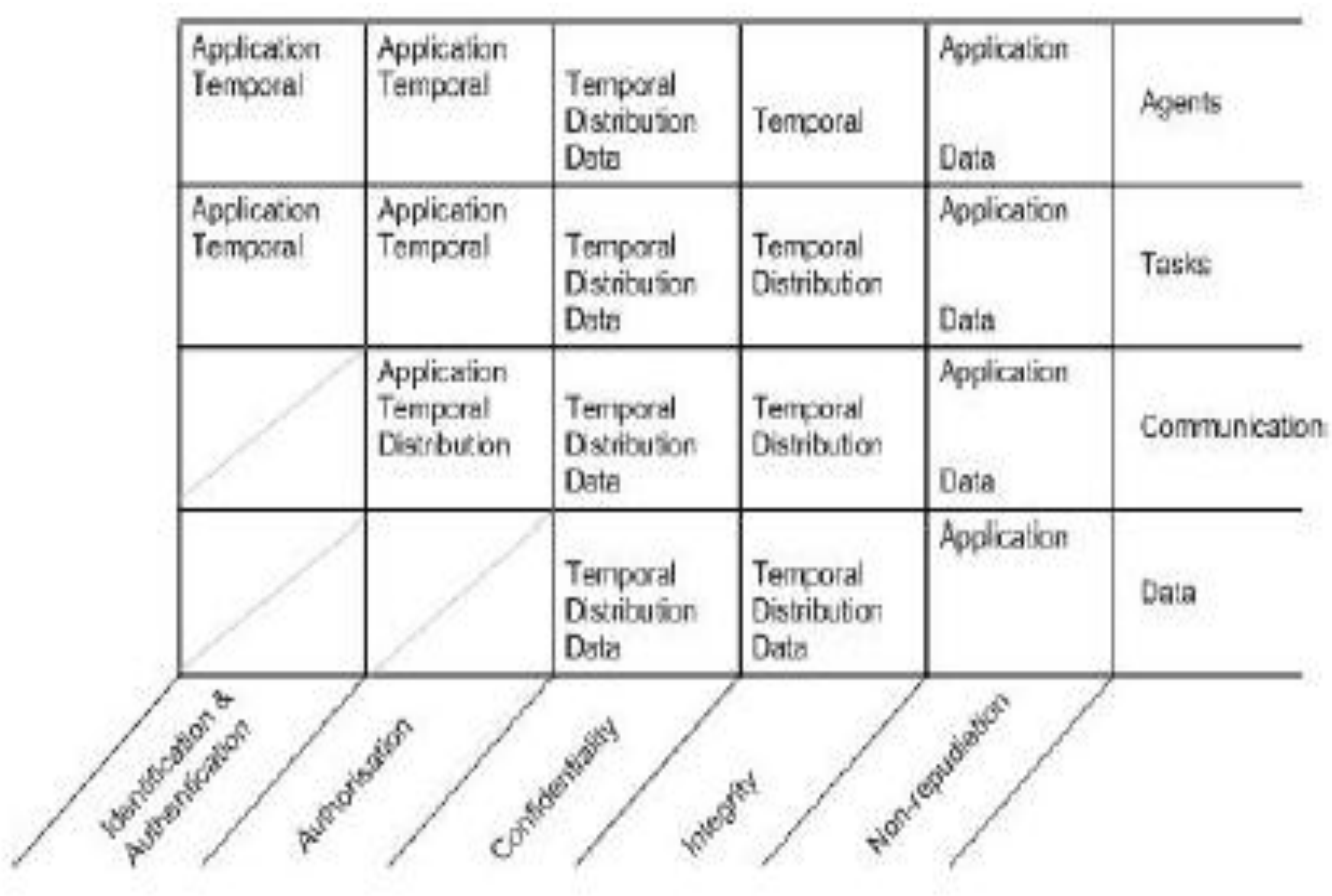


Relationships between System Entities





Security Service Requirements



Identification & Authentication	Authorisation	Confidentiality	Integrity	Non-repudiation	
Application Temporal	Application Temporal	Temporal Distribution Data	Temporal	Application Data	Agents
Application Temporal	Application Temporal	Temporal Distribution Data	Temporal Distribution	Application Data	Tasks
	Application Temporal Distribution	Temporal Distribution Data	Temporal Distribution	Application Data	Communication
		Temporal Distribution Data	Temporal Distribution Data	Application	Data

Threat Control Measures

- **Anticipate / Prevent** - threat types and sources are anticipated a priori, preventive action can be taken to reduce the likelihood of a threat being instantiated and the severity of its consequences
- **Detect imminent known or suspected attacks**, whether or not they are successful
- **Characterize -** attacks are defined so that appropriate short-term responses and long-term recovery actions can be formulated
- **Respond -** short-term responses are implemented to quickly isolate the consequences of threat instantiation
- **Recover -** long-term recovery measures are deployed to eliminate or mitigate the consequences of the same or similar threats in the future

Good Strategies - 1

- Use well known and carefully vetted validation code - Rather than rolling your own code for user input validation, use APIs that have been carefully developed and rigorously tested by others
- Specify variable types - Limit the kind of data that can be entered into some fields.
- Don’t define all possible bad characters - accept only the good ones
- Limit the size of input
- Canonicalize before filtering - If user input is encoded in a fashion that your filters aren’t designed to handle, your application very well might get hacked. Thus, whenever you receive user input, convert it to a standard encoding scheme, such as plain ASCII, before applying your filters. This process is known as canonicalization.
- Filter all input - Filter every form of input to your application, including data that comes in via the network, the GUI, hidden form elements, cookies, files read from the file system and so on.

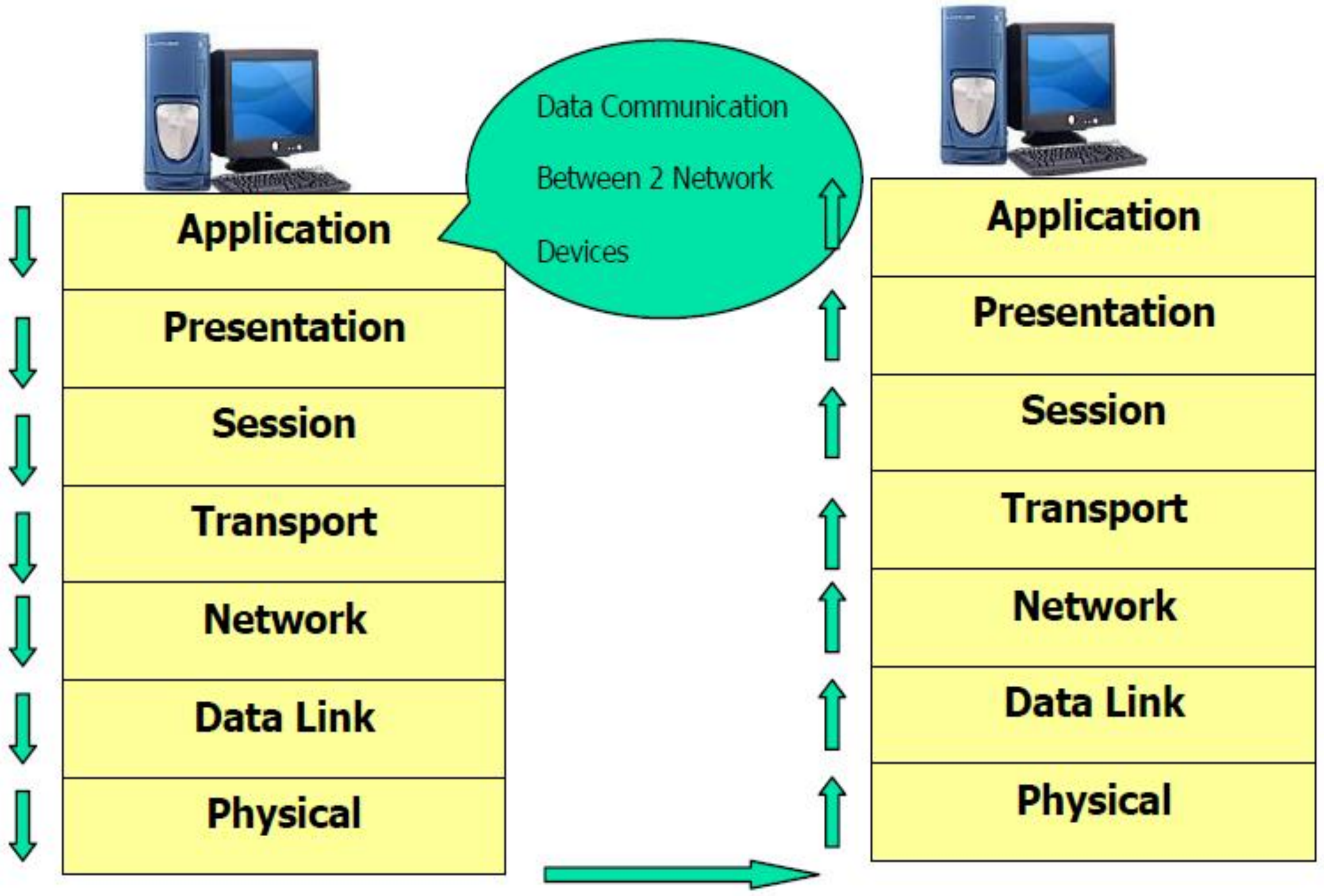
Good Strategies - 2

- Filter on the server side - In many applications, attackers might be able to control clients, such as browsers or thickclient GUIs, tweaking their functionality to bypass filtering done at the client. Filter on the server side to protect all back-end functionality.
- Don’t worry about multiple layers of validation - Sometimes, what appears to be a single application will include multiple subsystems. Multiple layers of filtering help provide defense in depth, making the whole system more secure.
- Utilize output encoding - Output encoding takes place when data is displayed back to the user or utilized by another component of the application.
- Choose the appropriate output encoding - HTML output encoding is only one example of how data may need to be encoded.
- Conduct a secure code review
- Perform a penetration test

SQL Injection Attacks

- SQL injection attacks are typically a way to steal credit card numbers, other valuable data, or as a pivot point from the internet to the internal network.
- We are now beginning to see SQL injection as a way to distribute malware making vulnerable web applications a platform for hackers to launch attacks to the client---side.
- The goal of the hackers is to infect as many computers as possible, adding them to the millions of infected bots already under their control.

ISO – OSI Reference Model



Security Services

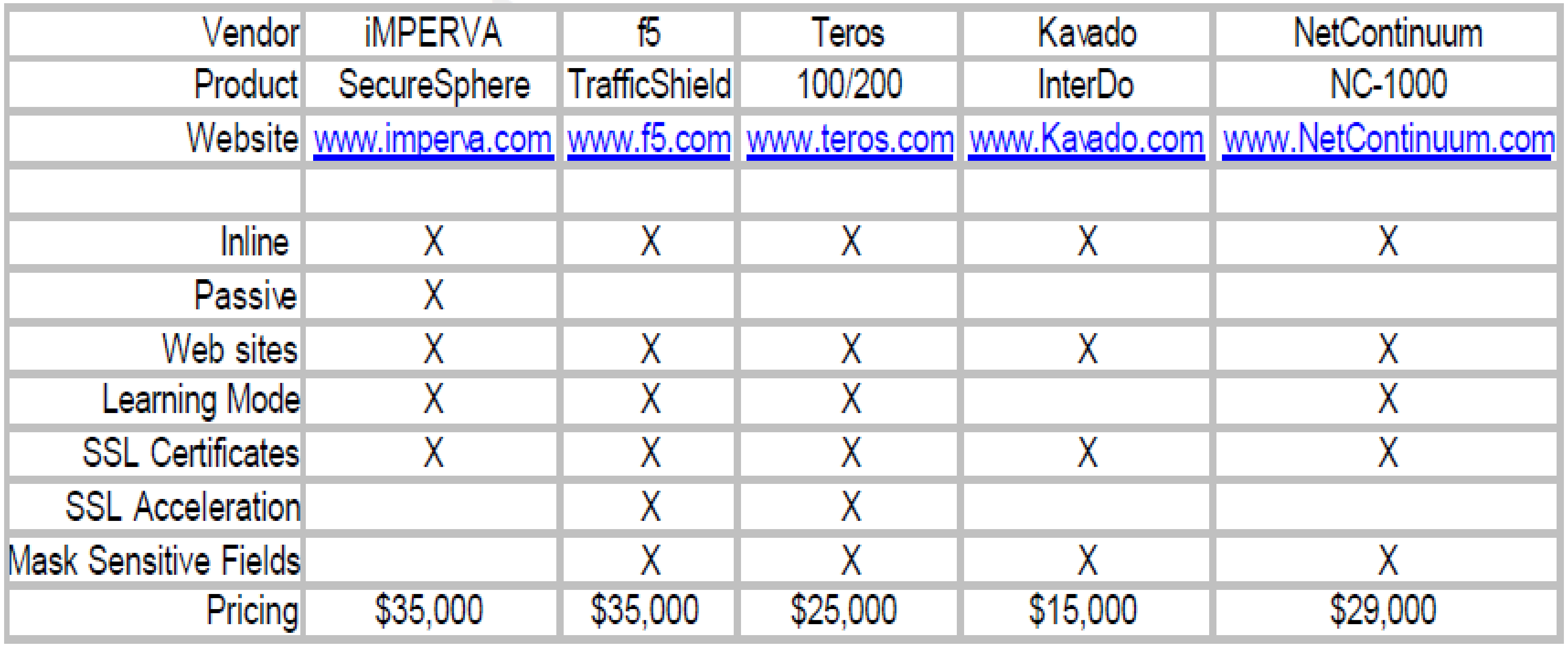
- **Confidentiality:** It is important to ensure that the information exchanged over an insecure channel must be protected against unauthorized disclosure. In other words, data transferred should only be seen by the intended recipient and not by an attacker observing the channel.
- **Data Integrity:** The data transferred should be protected against any modifications by an attacker. If there are any modifications to the data, the recipient should be informed about the tampering during the transfer.
- **Mutual Authentication:** All parties should authenticate themselves, that is prove their identities, before establishing a secure communication channel for data transfer.
- **Authorization:** The data from the restricted resource should only be shared with parties carrying required credentials that prove their rights to access the data.
- **Non repudiation:** The information exchange between two parties should ensure that none of the parties can deny their actions after the exchange. In other words, sufficient proof of information exchange or data transaction should be made available to prevent either parties in communication from repudiating.

Application Security – Some Existing Solutions

- Network Intrusion Detection and Prevention Systems (NIDS/NIPS) – Cannot block UDP.
- Host-Based Intrusion Detection Systems (HIDS) and Host-Based Intrusion Prevention Systems (HIPS) can also be used to protect servers.
- **Authentication, Identification and Authorization**
- Typically, network monitoring occurs below the Application layer.
- There are two fundamental security postures. They are the secure "default deny" and the reactive "default permit" stances.
- Typically network engineers see the need for the default deny posture while end users and developers prefer permitting anything "not dangerous".

Application Firewall – Don't Forget Layer 7

- Application – Protect Thyself
- Application Firewalls are a relatively inexpensive means



Vendor	iMPERVA	f5	Teros	Kavado	NetContinuum
Product	SecureSphere	TrafficShield	100/200	InterDo	NC-1000
Website	www.imperva.com	www.f5.com	www.teros.com	www.Kavado.com	www.NetContinuum.com
Inline	X	X	X	X	X
Passive	X				
Web sites	X	X	X	X	X
Learning Mode	X	X	X		X
SSL Certificates	X	X	X	X	X
SSL Acceleration		X	X		
Mask Sensitive Fields		X	X	X	X
Pricing	$35,000	$35,000	$25,000	$15,000	$29,000

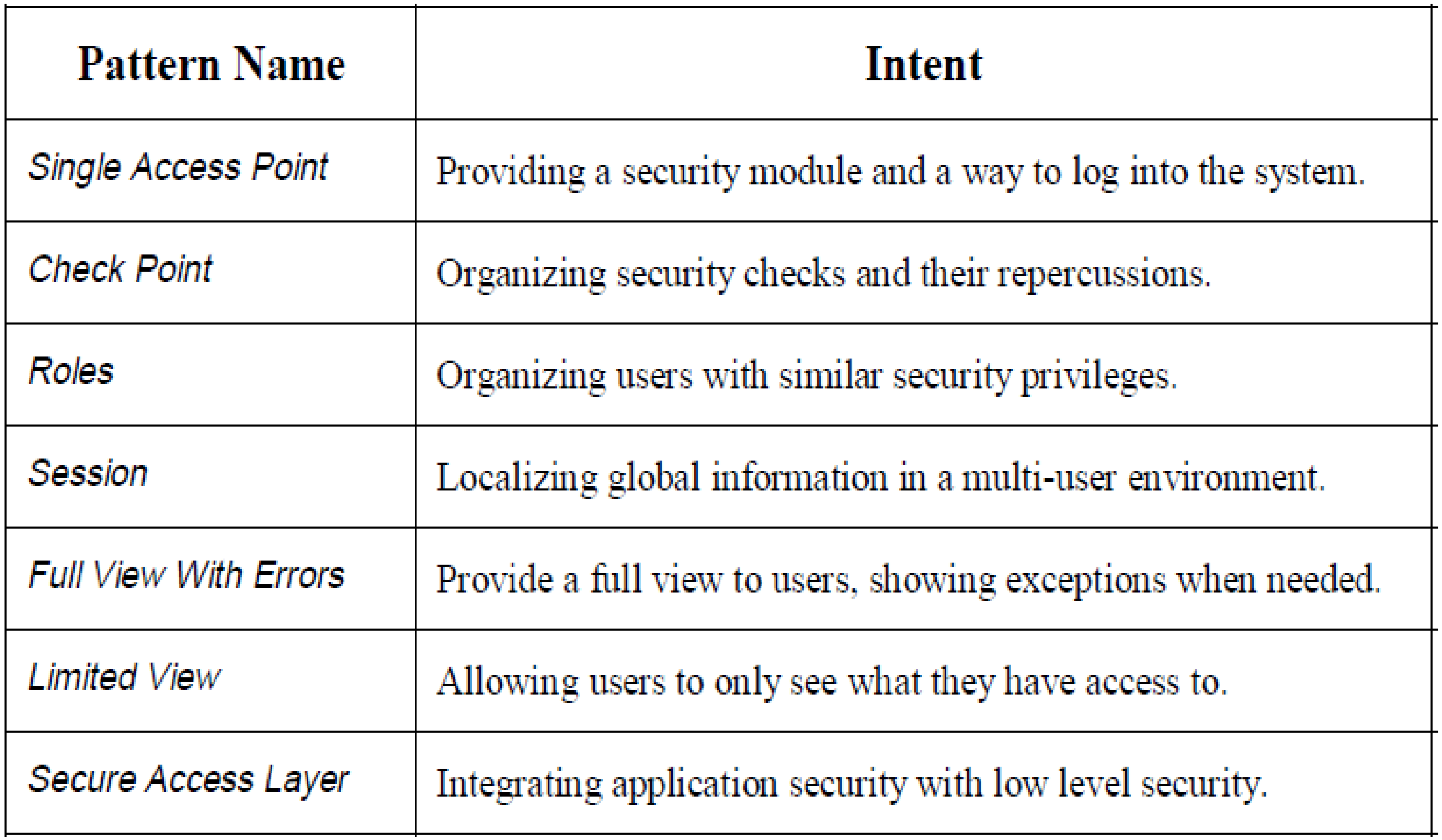
Application Layer Vulnerabilities

- Open design issues allow free use of application resources by unintended parties
- Backdoors and application design flaws bypass standard security controls
- Inadequate security controls force “all-or-nothing” approach, resulting in either excessive or insufficient access.
- Overly complex application security controls tend to be bypassed or poorly understood and implemented.
- Program logic flaws may be accidentally or purposely used to crash programs or cause undesired behavior

Application Level Access Controls

- Application level access controls to define and enforce access to application resources.
- Controls must be detailed and flexible, but also straightforward to prevent complexity issues from masking policy and implementation weakness Standards, testing, and review of application code and functionality.
- A baseline is used to measure application implementation and recommend improvements
- IDS systems to monitor application inquiries and activity
- Some host-based firewall systems can regulate traffic by application, preventing unauthorized or covert use of the network.

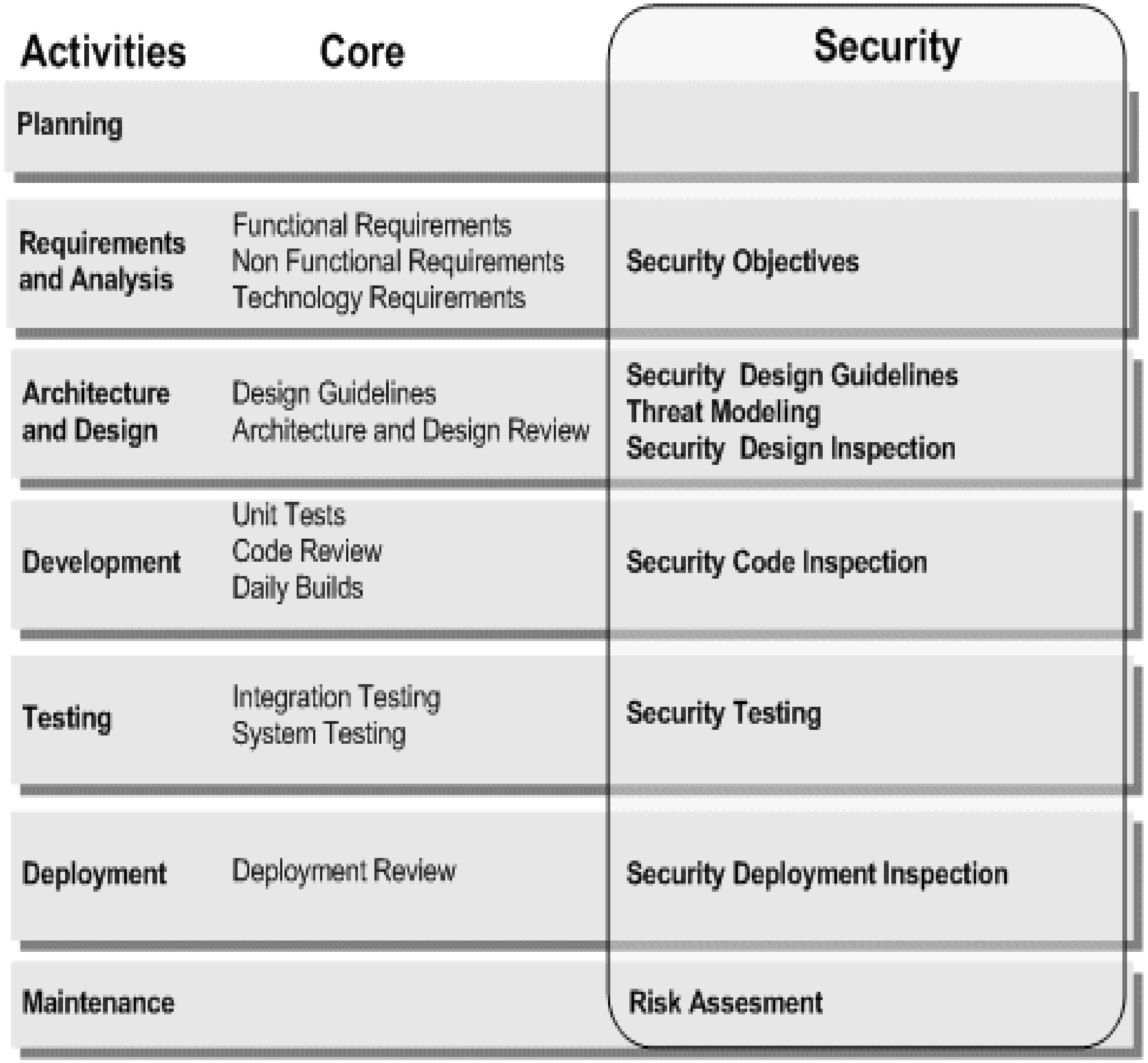
Application Layer Security - Patterns



Pattern Name	Intent
Single Access Point	Providing a security module and a way to log into the system.
Check Point	Organizing security checks and their repercussions.
Roles	Organizing users with similar security privileges.
Session	Localizing global information in a multi-user environment.
Full View With Errors	Provide a full view to users, showing exceptions when needed.
Limited View	Allowing users to only see what they have access to.
Secure Access Layer	Integrating application security with low level security.

OSI + HCI : 10 Layers

HCI **[QoE – Quality of Experience]**	10.	Human Needs (communication, education, acquisition, security, entertainment...)
	9.	Human Performance (perception, cognition, memory, motor control, social...)
	8.	Display (keyboard, GUI/CLI, vocal, bpp, ppi, ppm...)
OSI **[QoS – Quality of Service]**	7.	Application (http, ftp, nfs, pop...)
	6.	Presentation (ps, lz, iso-pp...)
	5.	Session (dns, rpc, pap...)
	4.	Transport (tcp, udp, rtp...)
	3.	Network (ip, dhcp, icmp, aep...)
	2.	Data Link (arp, ppp...)
	1.	Physical (10bt, xDSL, V.42...)



Activities	Core	Security
Planning		
Requirements and Analysis	Functional Requirements Non Functional Requirements Technology Requirements	Security Objectives
Architecture and Design	Design Guidelines Architecture and Design Review	Security Design Guidelines Threat Modeling Security Design Inspection
Development	Unit Tests Code Review Daily Builds	Security Code Inspection
Testing	Integration Testing System Testing	Security Testing
Deployment	Deployment Review	Security Deployment Inspection
Maintenance		Risk Assesment

Thank You