Unit #10

MIS5214

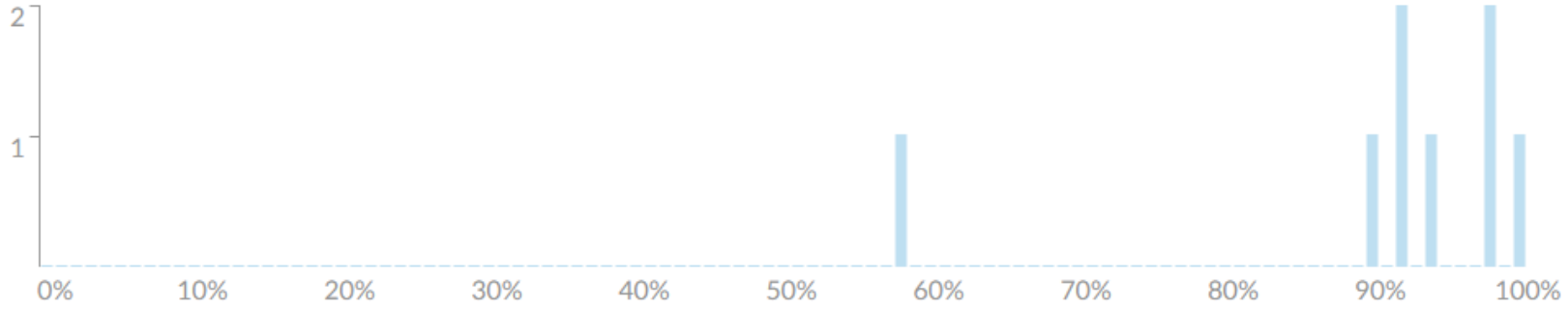
Application Security

Agenda

- Midterm Exam Review
- Team Project Guidance
- Distributed Systems
  - File Server Architecture
  - Client/Server Architecture
  - N-Tier Architecture
  - Cloud Architecture
  - Service Oriented Architecture (SOA)
- Example Cloud-based N-Tier SOA Application Development System
- Control Stages, Objectives, Application Security Testing
- Additional Best Practices

Midterm Exam Summary

Average Score	High Score	Low Score	Standard Deviation	Average Time
90%	100%	58%	12.63	01:13:25



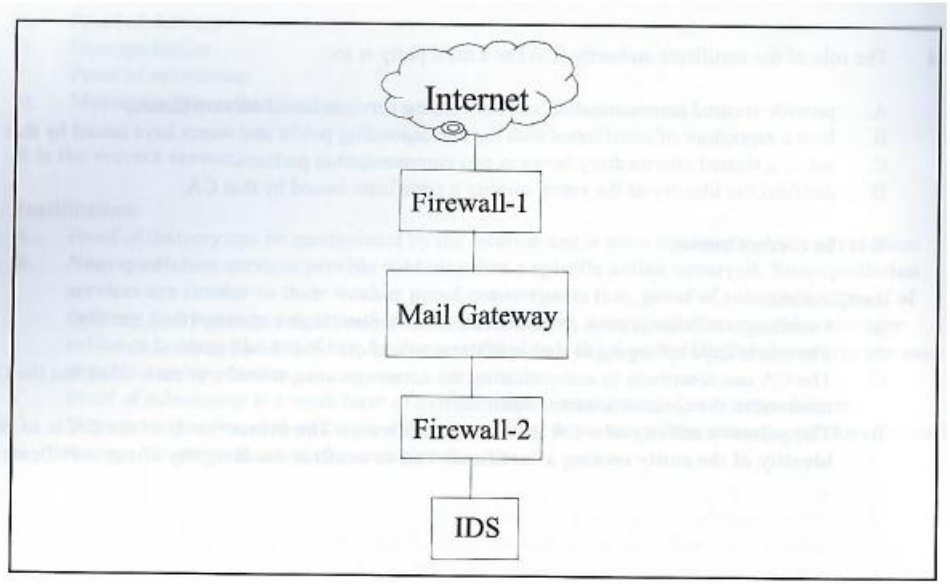
With respect to IT network security domains which of the following is false:

Routers are prohibited from connecting two Local Area Network security domains of different impact categorizations	5 respondents	**63 %**	
Logical and physical resources are available to users, processes and applications	1 respondent	13 %	
Different domains are separated by logical boundaries created by security components that enforce security policy for each domain		0 %	
Resources within each domain are working under the same security policy and managed by the same group	2 respondents	25 %	

Which of the following types of firewalls cannot make access decisions based on protocol commands?

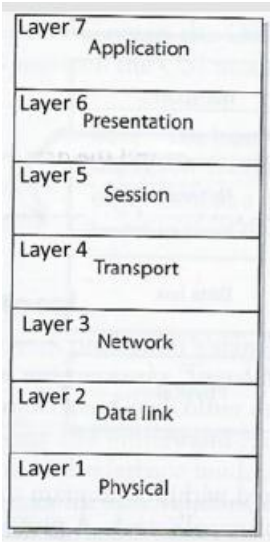
Kernal proxy	1 respondent	13 %	
Application-level	1 respondent	13 %	
Circuit-level proxy	5 respondents	**63 %**	
Next-generation	1 respondent	13 %	

With reference to the figure below, Email traffic from the Internet is routed via Firewall-1 to the mail gateway. Mail is routed from the mail gateway, via Firewall-2, to the mail recipients in the internal network. Other traffic is not allowed. For example, the firewalls do not allow direct traffic from the Internet to the internal network. The intrusion detection system (IDS) detects traffic for the internal network that did not originate from the mail gateway.



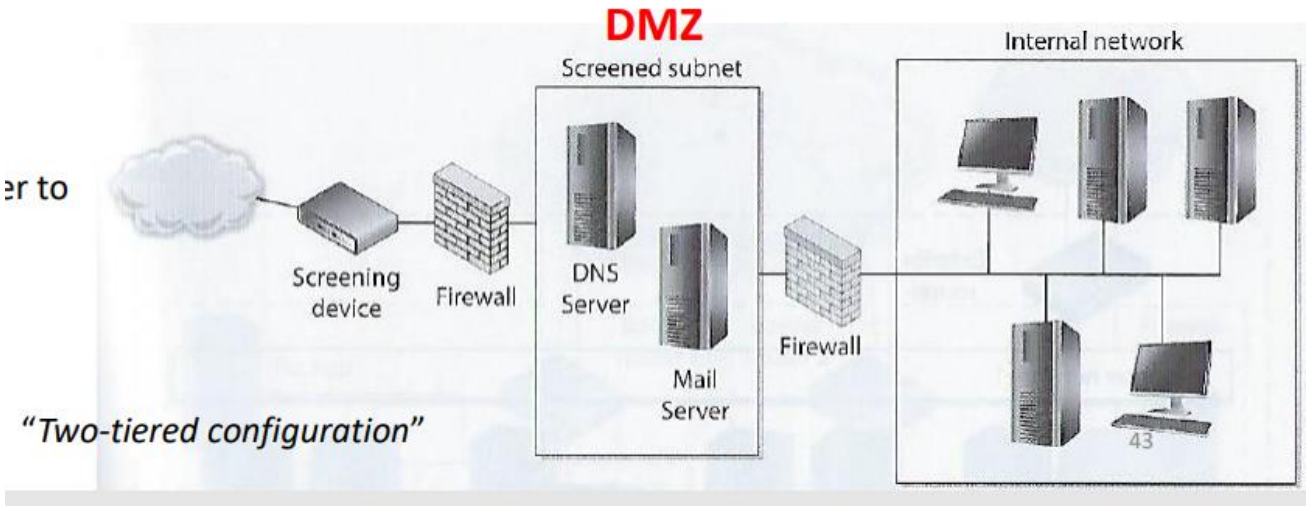
The first action triggered by the IDS should be to:

Close Firewall-2	3 respondents	38 %	
Alert the appropriate staff		0 %	
Close Firewall-1		0 %	
Create an entry in the log	5 respondents	**63 %**	✓



Which best describes the IP protocol?

A connection-oriented protocol that deals with sequencing, error detection, and flow control		0 %
A connectionless protocol that deals with the addressing and routing of packets	6 respondents	**75 %** ✓
A connection-oriented protocol that deals with the addressing and routing of packets	2 respondents	25 %
A connectionless protocol that deals with dialog establishment, maintenance, and destruction		0 %



Which of the following types of firewalls would BEST protect a network for an Internet attack?

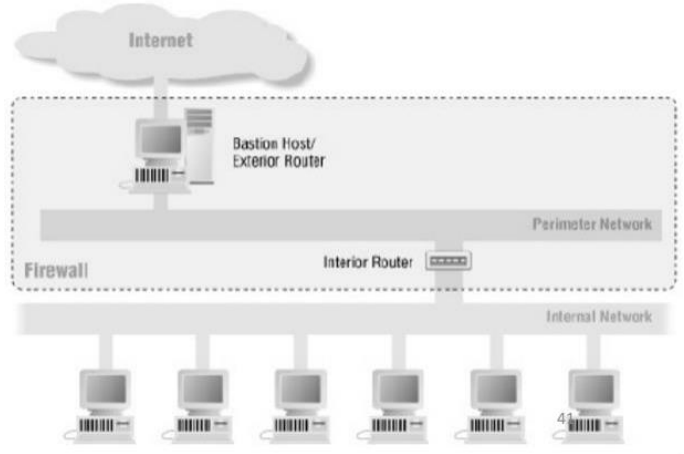
Packet filtering router	1 respondent	13 %	
Screened subnet firewall	6 respondents	**75 %**	✓
Application filtering gateway	1 respondent	13 %	
Circuit-level gateway		0 %	

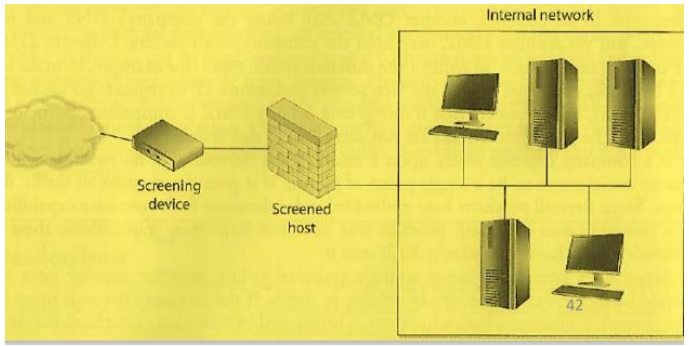
All of the following is true about the Screened Subnet Architecture except:

It has similar defense in depth characteristics as the Dual-Homed Firewall Architecture.	6 respondents	75 %
It is used to create a DMZ.	1 respondent	13 %
It includes the Screened-Host Architecture.	1 respondent	13 %
It is created using a router and two firewalls.		0 %

Characteristics of Firewall Architecture

- **Dual-homed**
  - A single computer with separate NICs connected to internal and external network
  - Used to divide an external untrusted network from an internal trusted network
  - Must harden and disable computer's forwarding and routing functionality so the two networks communicate through the computer's firewall software and are truly segregated
- **Screened host**
  - A router filters and screens traffic applying its ACL to drop 'junk' traffic before it is passed to the firewall
- **Screened subnet**
  - An external router filters/screens traffic before it enters the subnet, sending remaining traffic through two firewalls before making its way to the internal network





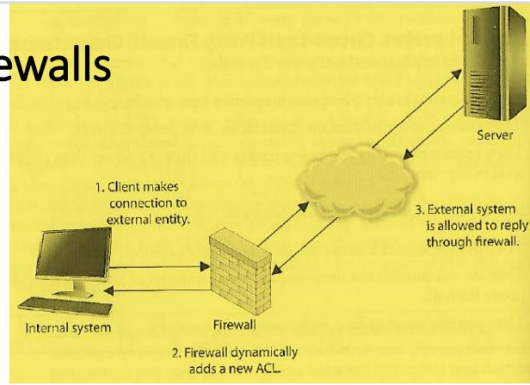
Ports 0 to 1023 are Well-Known Ports
Ports 1024 to 49151 are Registered Ports – Often registered by a software developer to designate a particular port for their application

Ports 49152 to 65535 are Public Ports

Dynamic Packet-Filtering Firewalls

When an internal system needs to communicate with a computer outside its trusted network it needs to choose an identify its source port so the receiving system knows how/where to reply

- Ports up to 1023 are reserved for specific server-side services and are known as "well-known ports"
- Sending system must choose a randomly identified port higher than 1023 to use to setup a connection with another computer



- The dynamic packet-filtering firewall creates an ACL that allows the external entity to communicate with the internal system via this high-numbered port
- The ACLs are dynamic in nature – once the connection is finished the ACL is removed
- The dynamic packet-filtering firewall offers the benefit of allowing any type of traffic outbound and permitting only response traffic inbound

A security manager at a large medical institution oversees a group that develops a proprietary software application that provides distributed computing through a client/server model. She has found that some of the systems that maintain the proprietary software have been experiencing half-open SYN flood denial-of-service attacks. Some of the software is antiquated and still uses basic remote procedure calls, which can allow for buffer overflow and remote attacker executing arbitrary code.

What type of client ports should the security manager make sure the institution's software is using when client-to-server communication needs to take place?

Free		0 %
Registered	2 respondents	25 %
Well known		0 %
Dynamic	6 respondents	**75 %** ✓

There are common cloud computing service models. _____ is the software environment that runs on top of the infrastructure. _____usually requires companies to deploy their own operating systems, application, and software onto the provided infrastructure. In the _____ model the provider commonly gives the customer network-based access to a single copy of an application. Select the answer that fills in the blanks in order:

Platform as a Service, Infrastructure as a Service, Software as a Service	6 respondents	75 %	 ✓
Infrastructure as a Service, Application as a Service, Software as a Service		0 %	
Platform as a Service, Platform as Software, Application as a Service		0 %	
Infrastructure as a Service, Platform as a Service, Software as a Service	2 respondents	25 %	

The MOST important difference between hashing and encryption is that hashing:

Is the same at the sending and receiving end	1 respondent	13 %
Is concerned with integrity and security		0 %
Is not reversible	6 respondents	**75 %** ✓
Output is the same length as the original message	1 respondent	13 %

Team Project SSP Deliverable Sections to Fill out: 1, 2, 7, 8, 9, 10.1, 11, 13 (one technical control family), Attachments 3, 4, 6, and 10

TABLE OF CONTENTS

Team Project Guidance

Instructions for diagrams for sections:

- 9.2 (Information System Components and Boundaries),
- 9.4 (Network Architecture)
- 10.1 (Data Flow)

These can all be based on the high-level logical network diagram you create for section 9.4 (Network Architecture)

Be sure to include the locations of the users in your diagram

In addition to including them in your SSP, you should include and label this diagrams in a separate PDF file document that you deliver along with your SSP

Team Project Guidance

Instructions for Section 11’s Table 11-1, only identify:

- External System (column 2)
- Connection Security (column 4)
- Data Direction (column 5)
- Information Being Transmitted (column 6)

11 SYSTEM INTERCONNECTIONS

Instruction: List all interconnected systems. Provide the IP address and interface identifier (eth0, eth1, eth2) for the CSP system that provides the connection. Name the external organization and the IP address of the external system. Indicate how the connection is being secured. For Connection Security indicate how the connection is being secured. For Data Direction, indicate which direction the packets are flowing. For Information Being Transmitted, describe what type of data is being transmitted. If a dedicated telecom line is used, indicate the circuit number. Add additional rows as needed. This table must be consistent with Table 13-3 CA-3 Authorized Connections.

Delete this and all other instructions from your final version of this document.

The Table 11-1 System Interconnections below is consistent with Table 13-3 CA-3 Authorized Connections.

Table 11-1 System Interconnections

SP* IP Address and Interface	**External Organization Name** and IP Address of System	External Point of Contact and Phone Number	**Connection Security (IPSec VPN, SSL, Certificates, Secure File Transfer, etc.)****	**Data Direction (incoming, outgoing, or both)**	**Information Being Transmitted**	Port or Circuit Numbers
<SP IP Address/Interface>	<External Org/IP>	<External Org POC> <Phone 555-555-5555>	<Enter Connection Security>	Choose an item.	<Information Transmitted>	<Port/Circuit Numbers>
<SP IP Address/Interface>	<External Org/IP>	<External Org POC> <Phone 555-555-5555>	<Enter Connection Security>	Choose an item.	<Information Transmitted>	<Port/Circuit Numbers>
<SP IP Address/Interface>	<External Org/IP>	<External Org POC> <Phone 555-555-5555>	<Enter Connection Security>	Choose an item.	<Information Transmitted>	<Port/Circuit Numbers>
<SP IP Address/Interface>	<External Org/IP>	<External Org POC> <Phone 555-555-5555>	<Enter Connection Security>	Choose an item.	<Information Transmitted>	<Port/Circuit Numbers>
<SP IP Address/Interface>	<External Org/IP>	<External Org POC> <Phone 555-555-5555>	<Enter Connection Security>	Choose an item.	<Information Transmitted>	<Port/Circuit Numbers>
<SP IP Address/Interface>	<External Org/IP>	<External Org POC> <Phone 555-555-5555>	<Enter Connection Security>	Choose an item.	<Information Transmitted>	<Port/Circuit Numbers>

Team Project Guidance

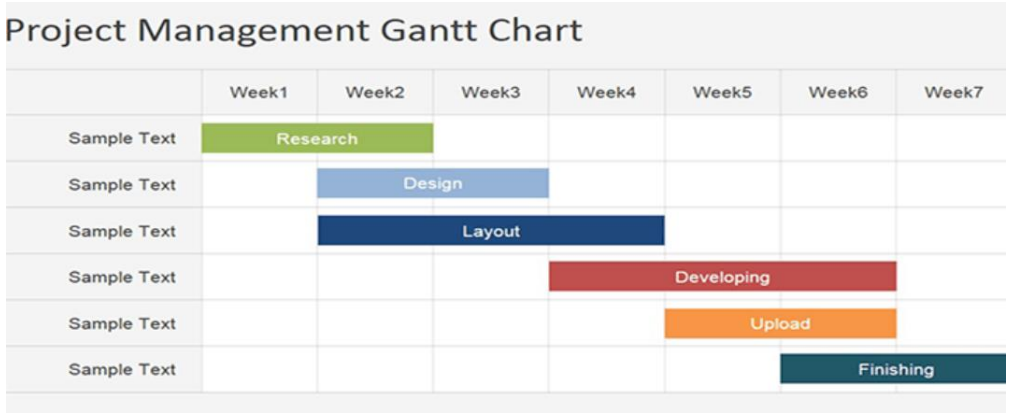
Instructions for Section 13: Only select and complete one **technical** control family

From NIST SP 800-18r1 Guide for Developing Security Plans for Federal Information Systems

CLASS	FAMILY	IDENTIFIER
Management	Risk Assessment	RA
Management	Planning	PL
Management	System and Services Acquisition	SA
Management	Certification, Accreditation, and Security Assessments	CA
Operational	Personnel Security	PS
Operational	Physical and Environmental Protection	PE
Operational	Contingency Planning	CP
Operational	Configuration Management	CM
Operational	Maintenance	MA
Operational	System and Information Integrity	SI
Operational	Media Protection	MP
Operational	Incident Response	IR
Operational	Awareness and Training	AT
Technical	Identification and Authentication	IA
Technical	Access Control	AC
Technical	Audit and Accountability	AU
Technical	System and Communications Protection	SC

Team Project Guidance

- Attachment 6 - Information System Contingency Plan: Only provide a plan (include a schedule tasks with labor estimate in person-hours) for completing Attachment 6 which is an Information System Contingency Plan (ISCP) based on FedRAMP ISCP Template



Questions…?

As far Team Project Deliverables go the System Security Plan document is one of the documentation hand-ins after they make your Team presentation. Each student will be responsible for handing in the Team's documentation set via Canvas i.e. all 4 documents: Powerpoint Slide Deck, diagrams (network, boundary, data flow) as PDF, SSP, and 360 degree review.

With respect to the Powerpoint Presentation they deliver in-class they should be sure to include:

1. Information System Name (SSP section 1)
2. System Function/Purpose (SSP section 9.1)
3. Information System Categorization (SSP Section 2 – all subsections)
4. Information System Operational Status (Section 7)
5. Types of Users (SSP section 9.3)
6. Network Architecture with boundaries (i.e. Logical Network with Boundaries – Boundary Diagram)
7. System Interconnections (i.e. Use the Boundary diagram – logical network diagram with boundaries to identify the system interconnections)
8. Minimum Security Controls (i.e. Indicate which technical control family you focused on and describe interesting controls when presenting 6 & 7)
9. … anything else they wish to cover

Ideally, 1 -5 introduces 6-8. If you have your presentations together nicely, they will quickly cover 1-5 and then present one diagram to quickly discuss 6 – 8.

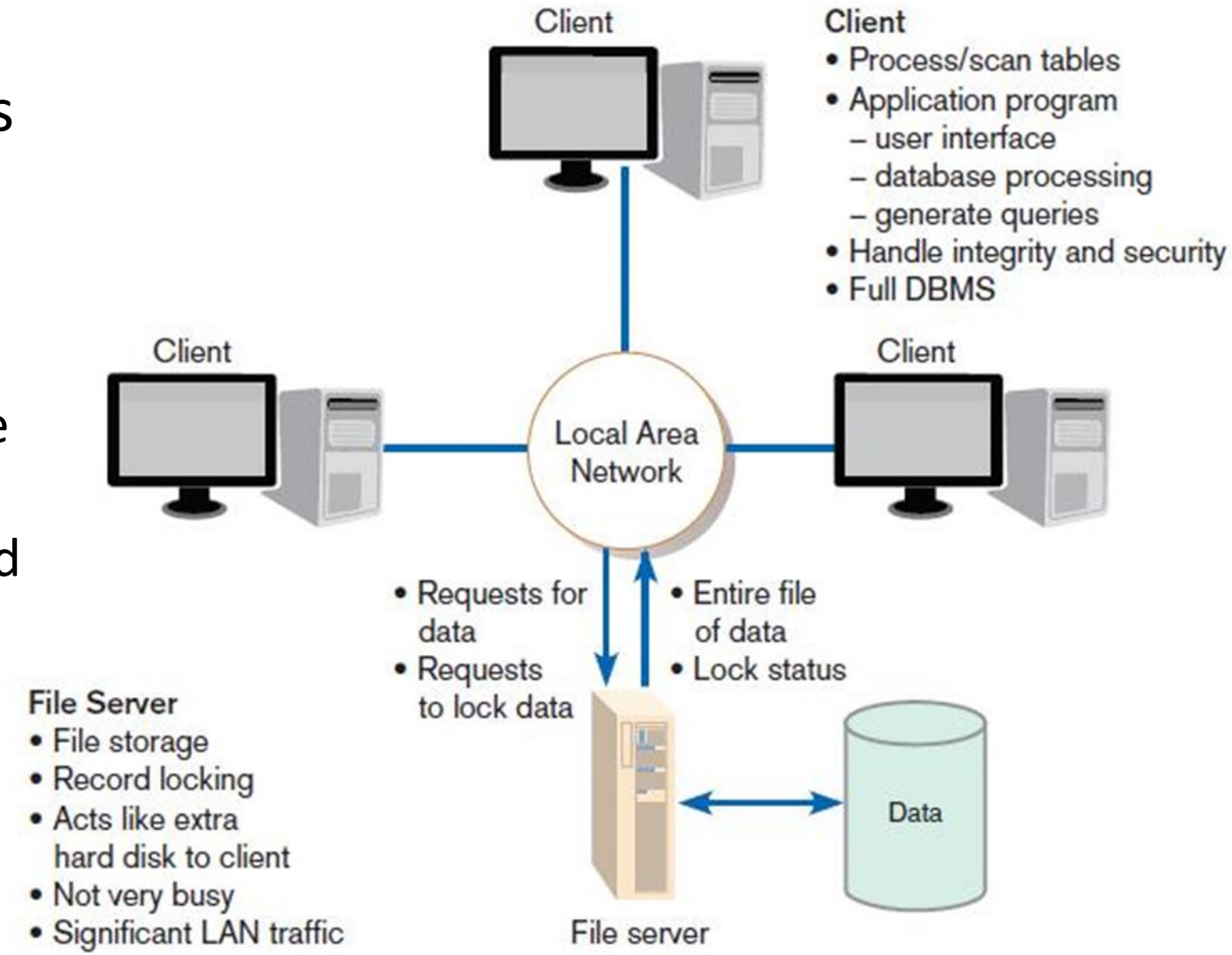
Agenda

- ✓Midterm Exam Review
- ✓Team Project Guidance
- Distributed Systems
  - File Server Architecture
  - Client/Server Architecture
  - N-Tier Architecture
  - Cloud Architecture
  - Service Oriented Architecture (SOA)
- Example Cloud-based N-Tier SOA Application Development System
- Control Stages, Objectives, Application Security Testing
- Additional Best Practices

File Server architecture

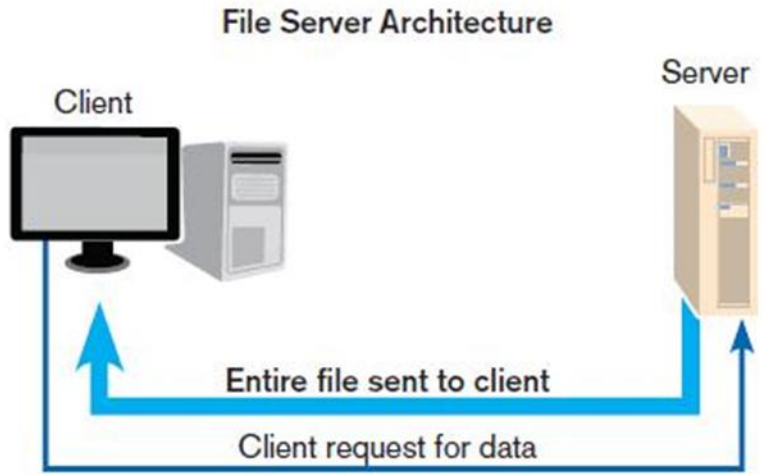
File server: a device that manages file operations and is shared by each client PC attached to a LAN

- The simplest configuration
  - Applications and data control take place on the client computers.
  - The file server simply holds shared data

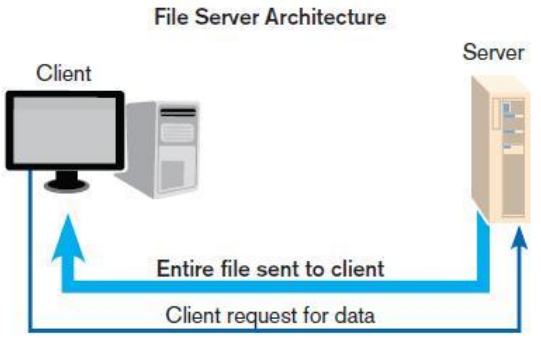


Limitations of File Server Architecture

- Excessive data movement
  - Entire dataset must be transferred, instead of individual data records
- Need for powerful client workstations
  - Each client workstation must devote memory and computational resources to run a complete standalone application
- Decentralized data control
  - Data file concurrency control, recovery, and security are complicated

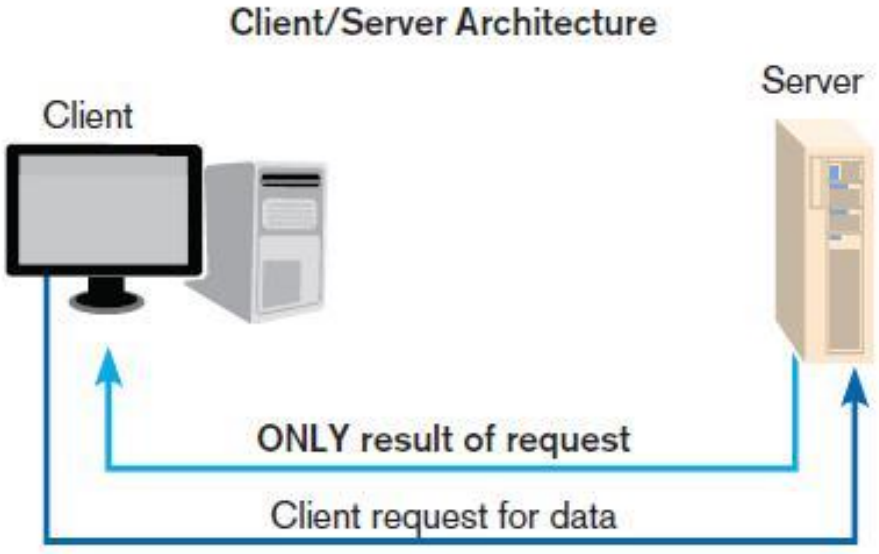


Client-Server Architecture



LAN-based computing environment in which

- A central database server or engine performs all database commands sent to it from client workstations
- Application programs on each client concentrate on user interface functions



Application processing is divided between client and server

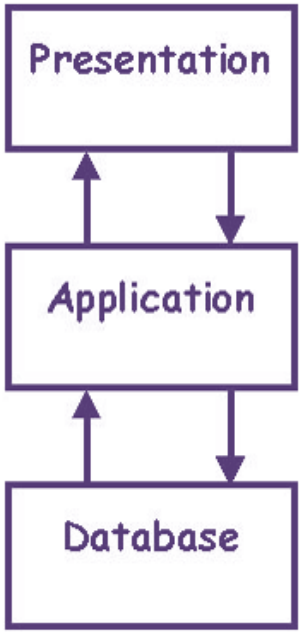
Client manages the user interface

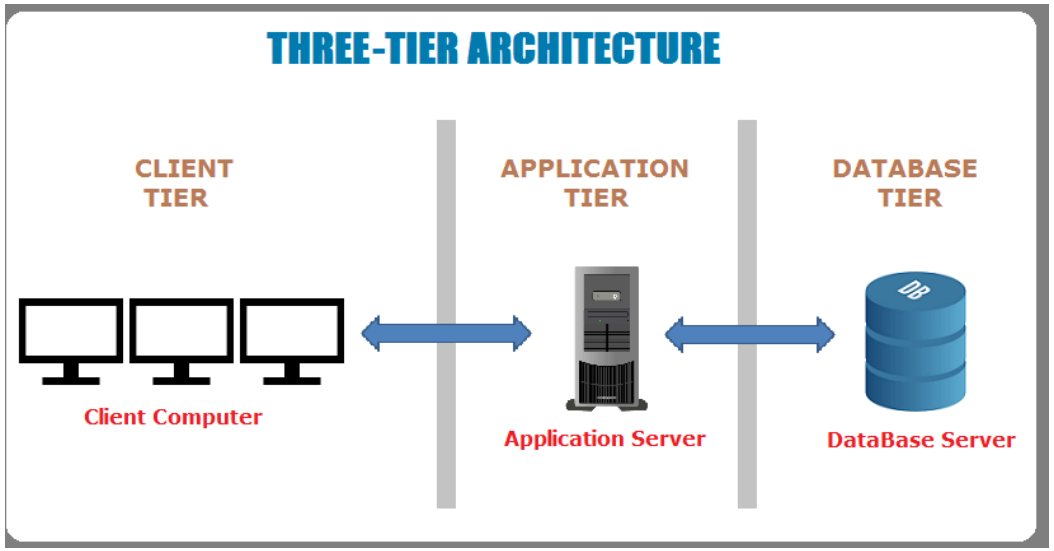
Database server is responsible for data storage and query processing

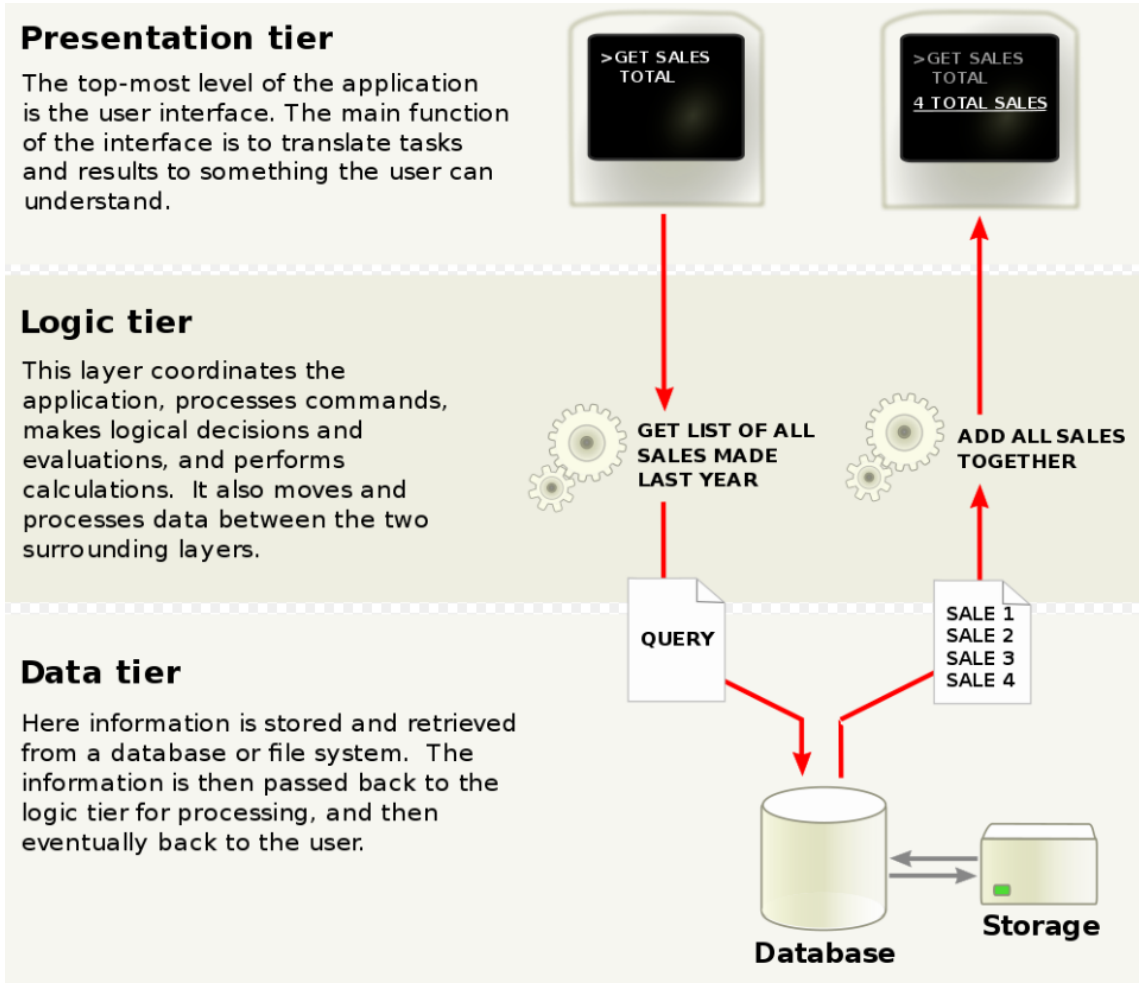
Increased efficiency and control over File server

- Server only sends specific data, not entire files, which saves on network bandwidth
- Computing load is carried out by the server
  - Increasing security
  - Decreasing computing demand on the clients

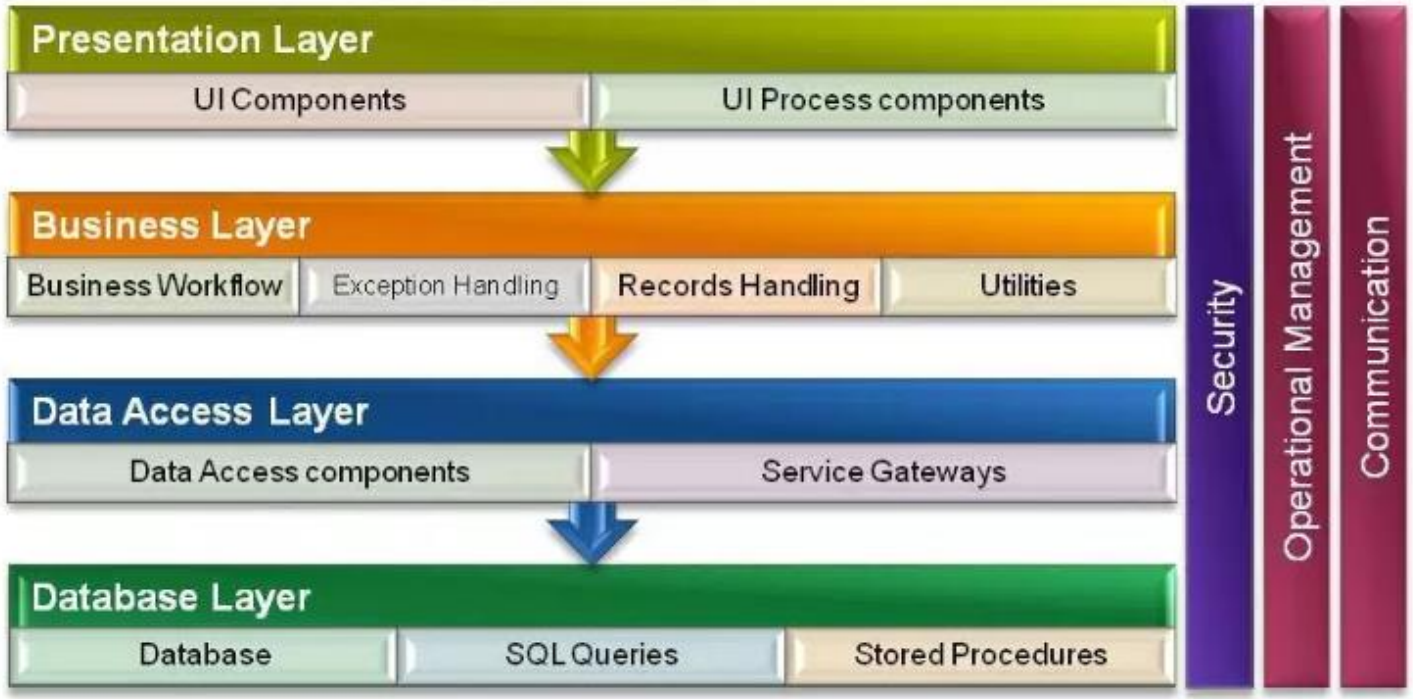
N-Tier Architecture





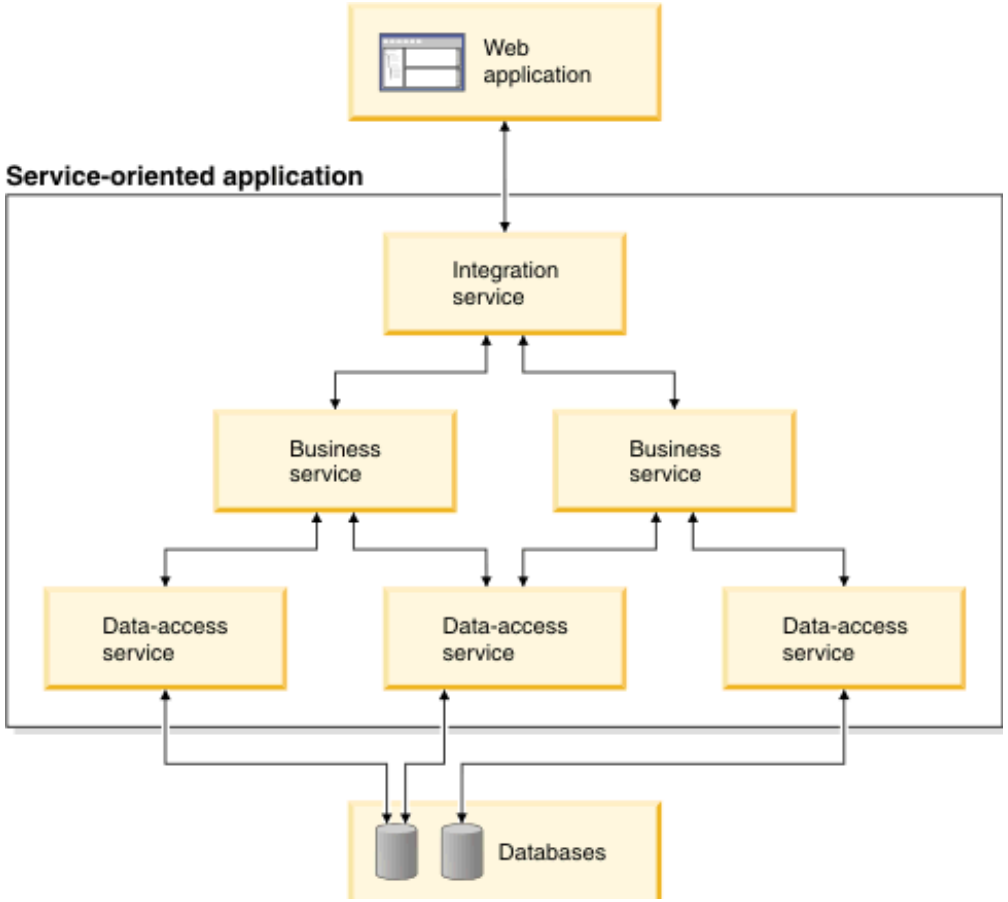


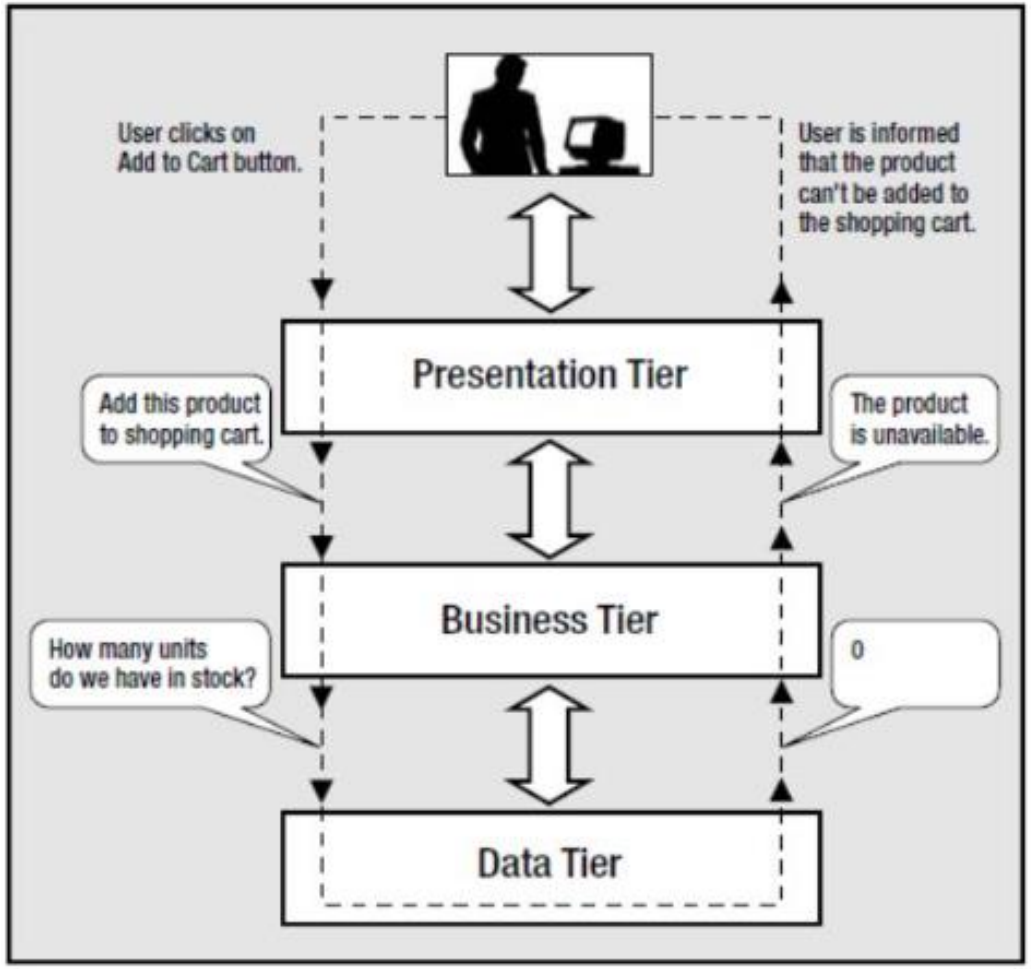
N-Tier Applications



Where's the programming code?

N-Tier Applications





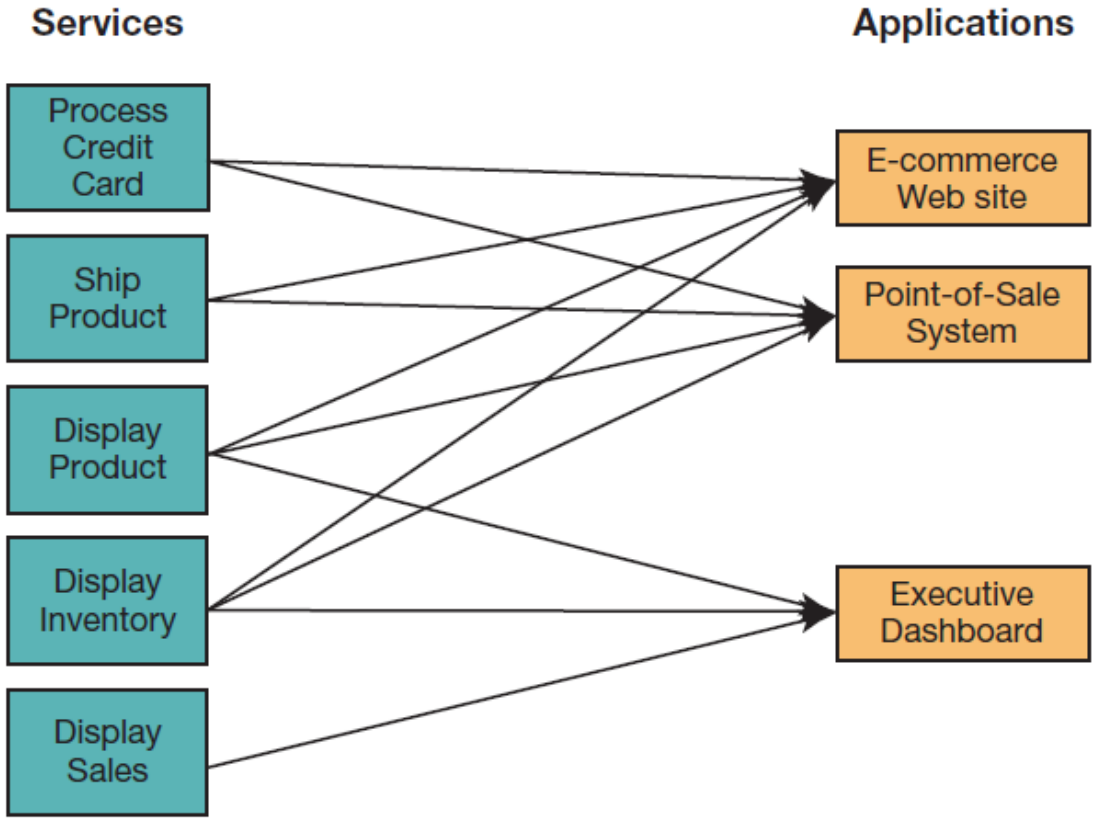
Service Oriented Architecture (SOA)

A **software** architecture

- Business processes broken down into individual components (services)
- Designed to achieve desired results for the service **consumer**
  - Application
  - Another service
  - Person (user)

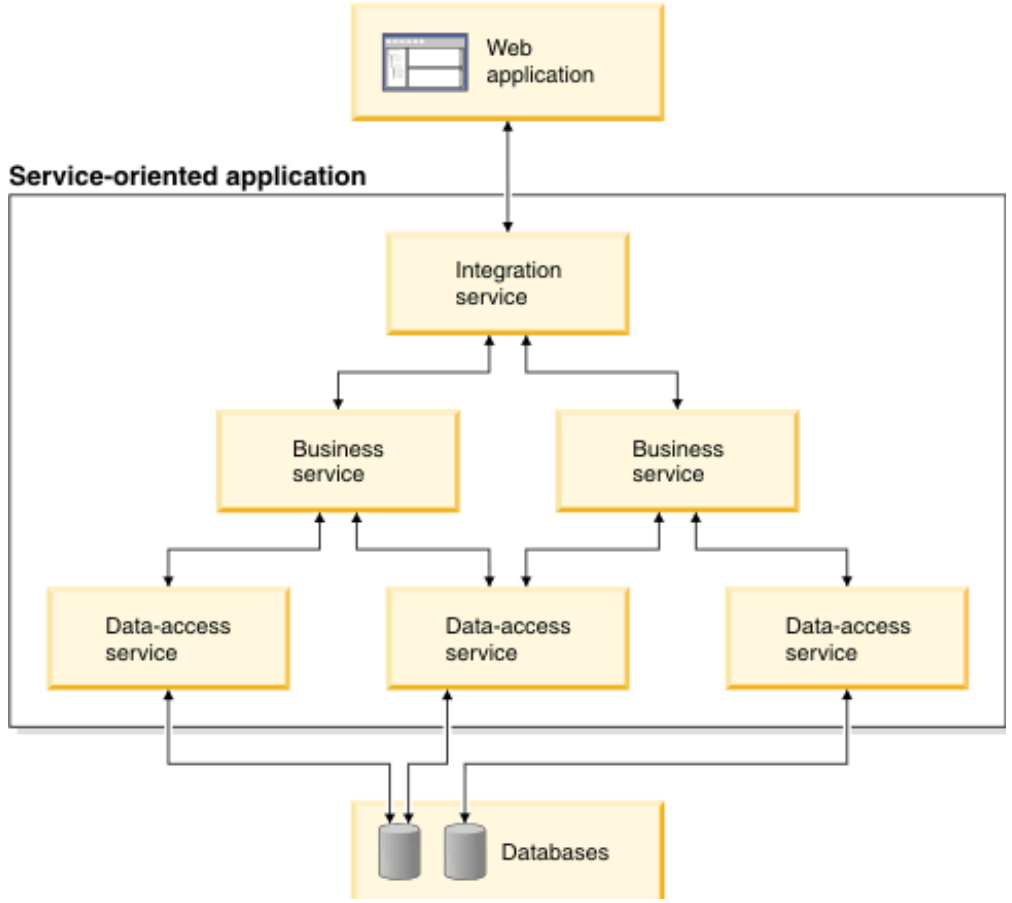
Principles:

- Reusability
- Interoperability
- Componentization



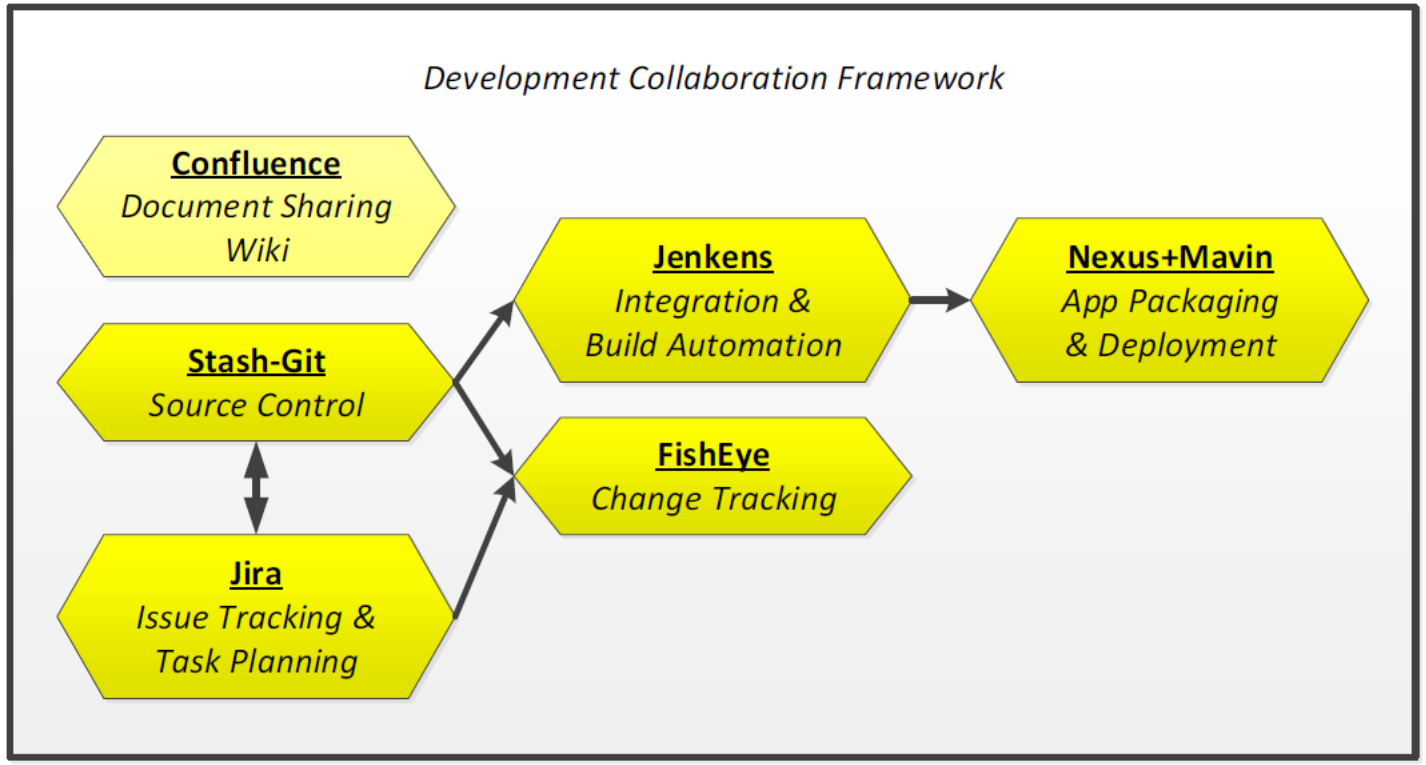
Using SOA, multiple applications can invoke multiple services

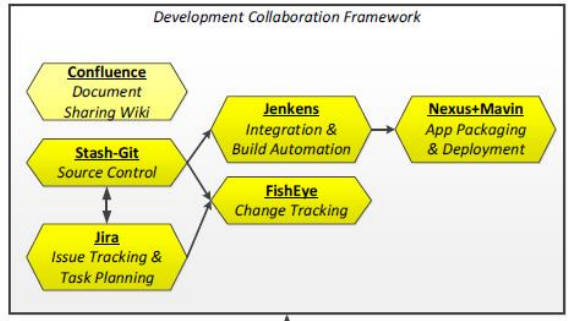
N-Tier Applications using SOA in the cloud

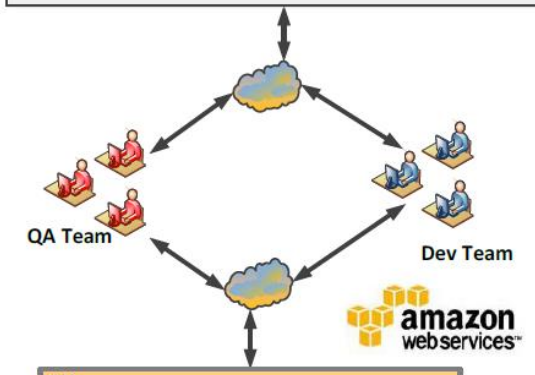


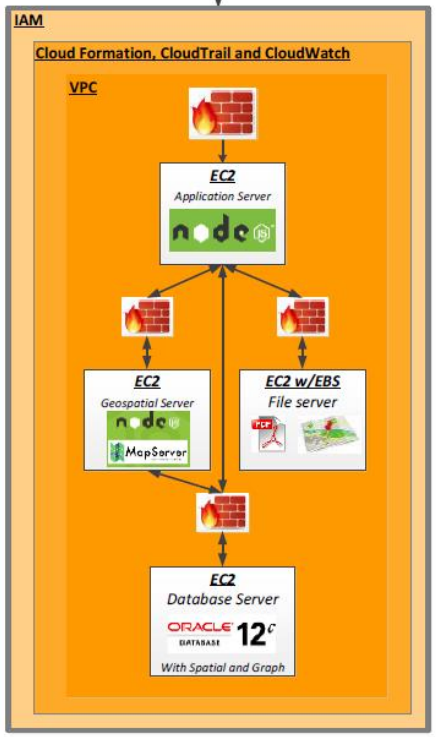
Development Infrastructure Example…

Examples of supporting systems





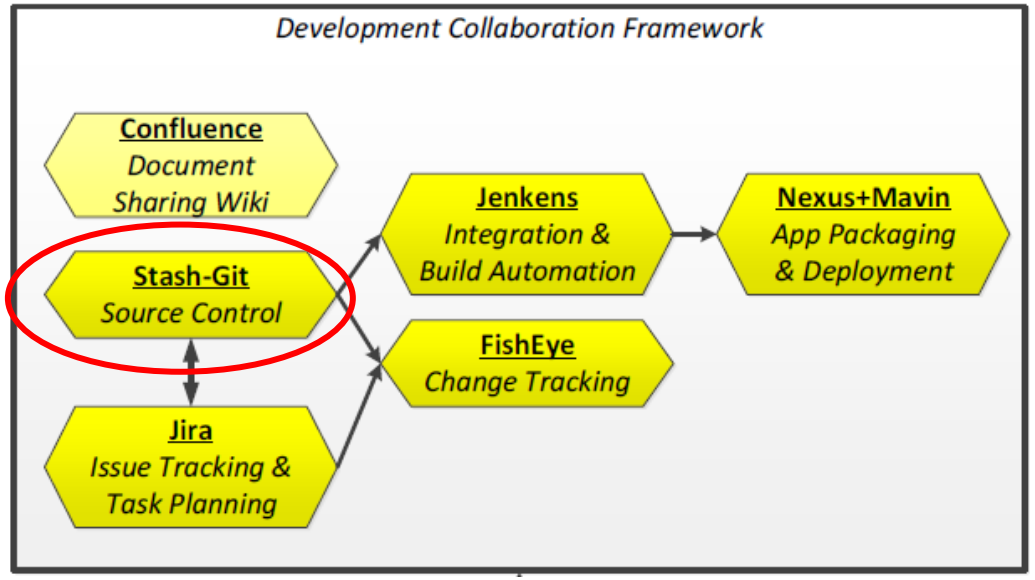




Source Control

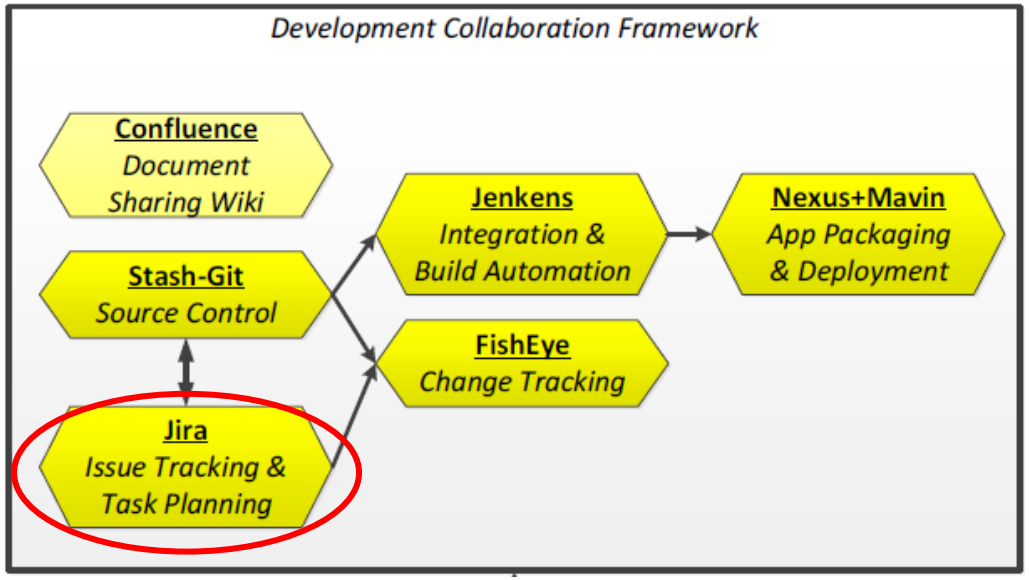
- Web-based hosting of repository service for distributed access and version control of programming code

- Enables maintaining versioned shareable software code and design artifacts with check-in/check-out and maintenance capabilities



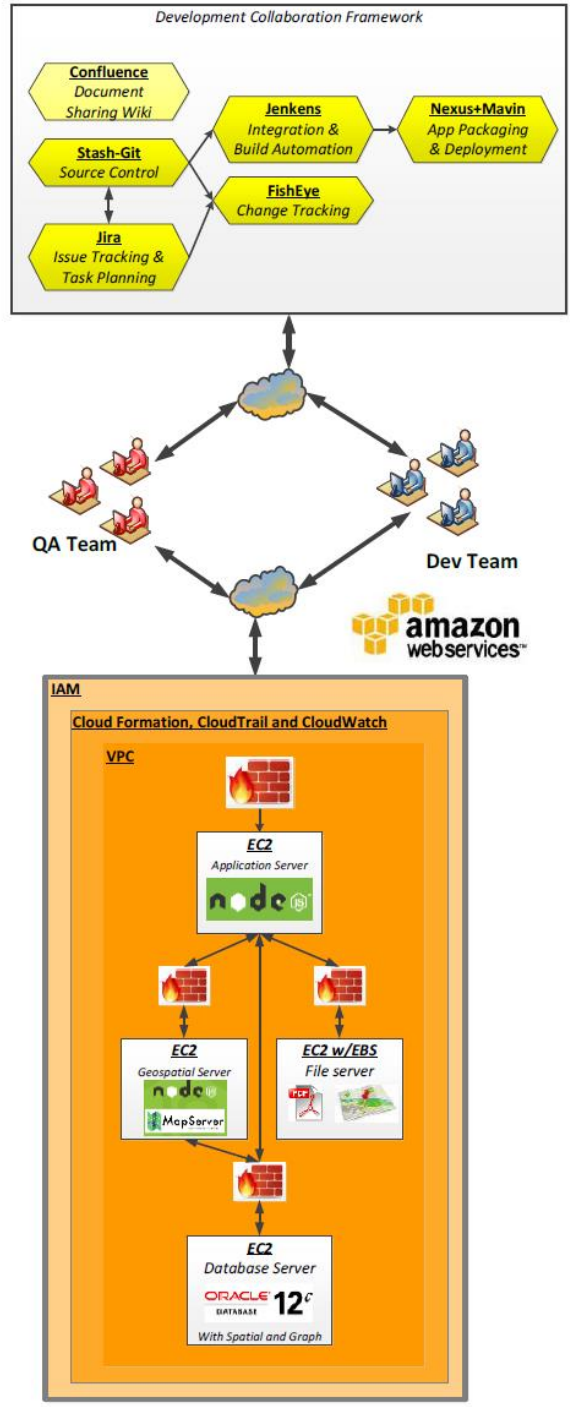
Issue Tracking System

- Enables organization, prioritization, triage, planning and tracking resolution of issues, bugs, and project tasks
- Provides visibility of issues and tasks on “To Do”, “In Progress”, “For Review”, and “Done lists”
- *Integration of Issue Tracking and Source Control Systems enables end-to end traceability of issue and tasks through resolution to source code implementation and issue to source code resolution*
- Change tracking and control enables visualizing and reporting on revisions and changes made to source code and documents by project teammates
- *Enables linking software issue documentation to code differences, sets of changes, full source code and provides a visual audit trail of changes over time in Source Control*



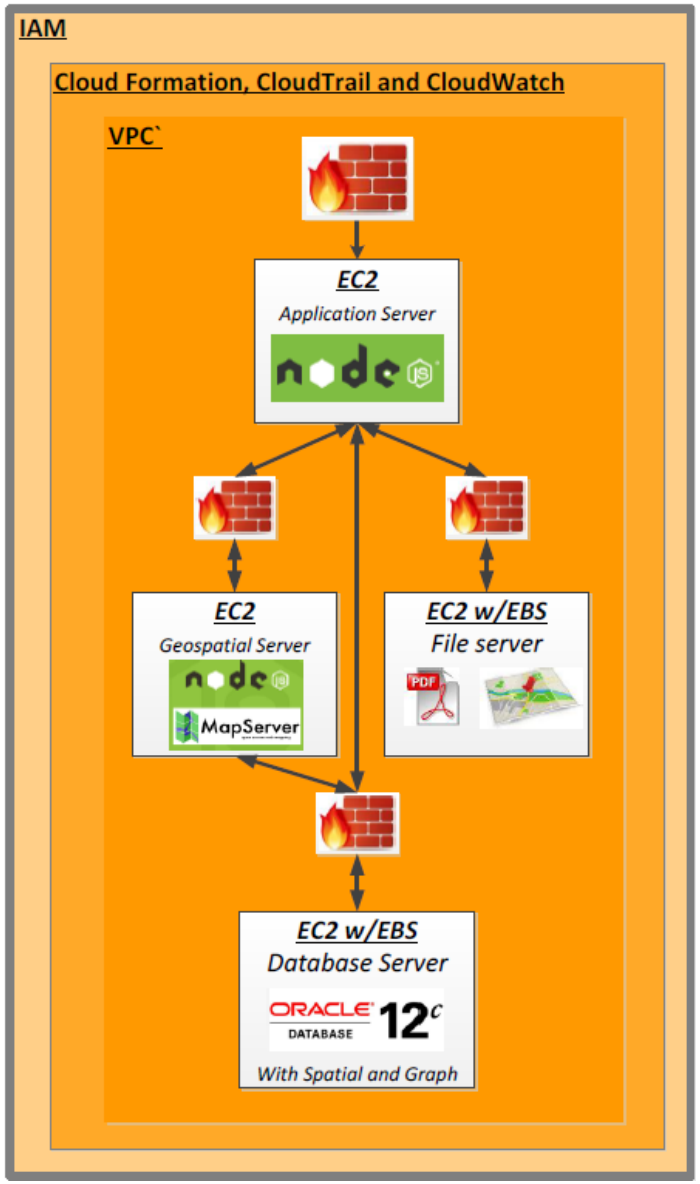
Continuous Integration & Continuous Deployment

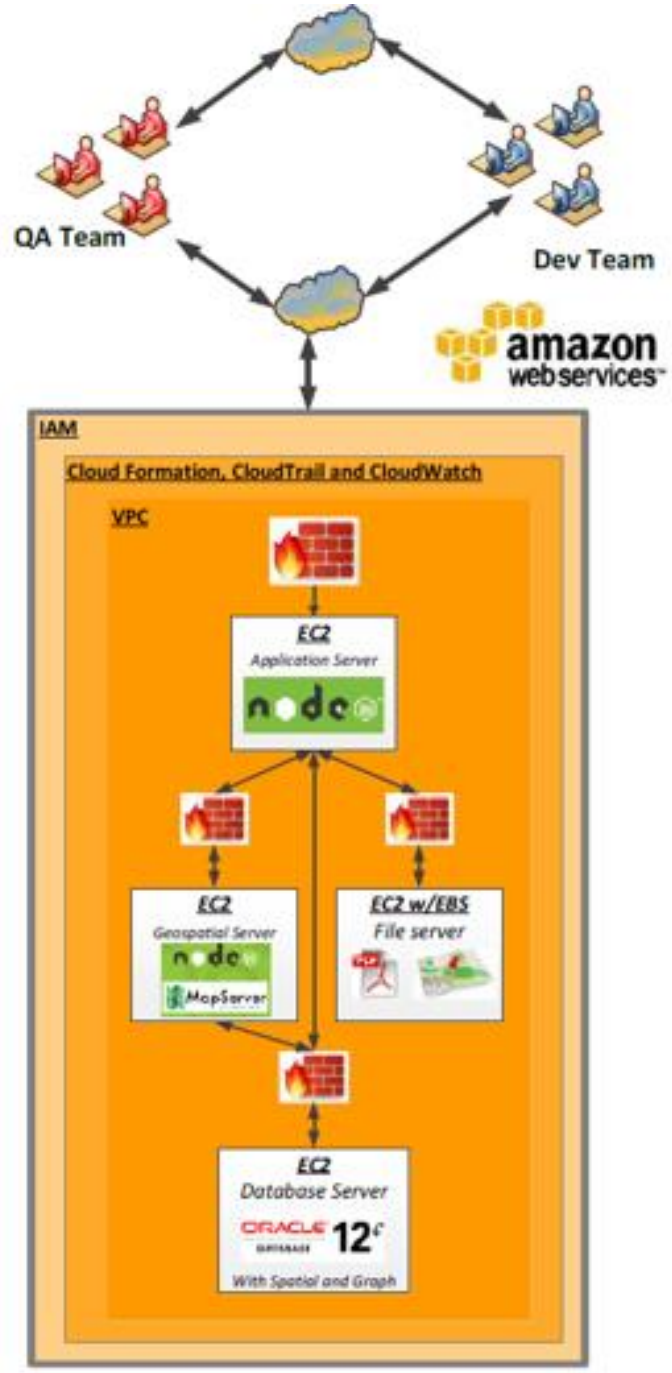
- Helps development team make system builds, triggered by either
  - A commit of updated source code to the version control system
  - Scheduling directive
  - A dependency on the completion of another component's build
  - Developer kicking off the build using a URL to make the request



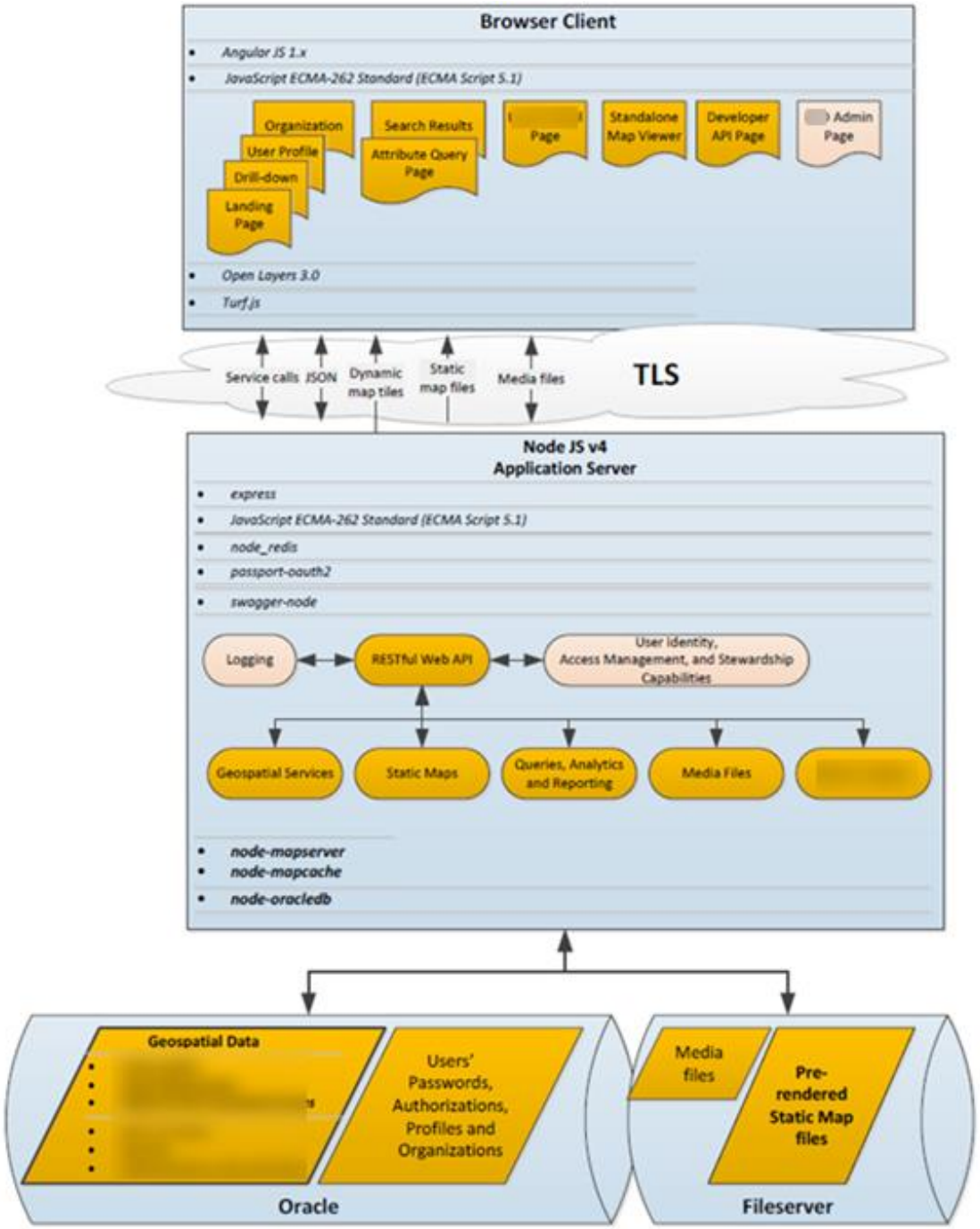
Development Infrastructure Example…

VPC = virtual private cloud





Application 3+ Tier Architecture example



Agenda

- ✓Midterm Exam Review
- ✓Team Project Guidance
- ✓Distributed Systems
  - ✓File Server Architecture
  - ✓Client/Server Architecture
  - ✓N-Tier Architecture
  - ✓Cloud Architecture
  - ✓Service Oriented Architecture (SOA)
- ✓Example Cloud-based N-Tier SOA Application Development System
- Control Stages, Objectives, Application Security Testing
- Additional Best Practices
- Team Project Guidance

Information System Development Control Stages

Control over applications is conducted at every stage and begins at the start of the development of the information system

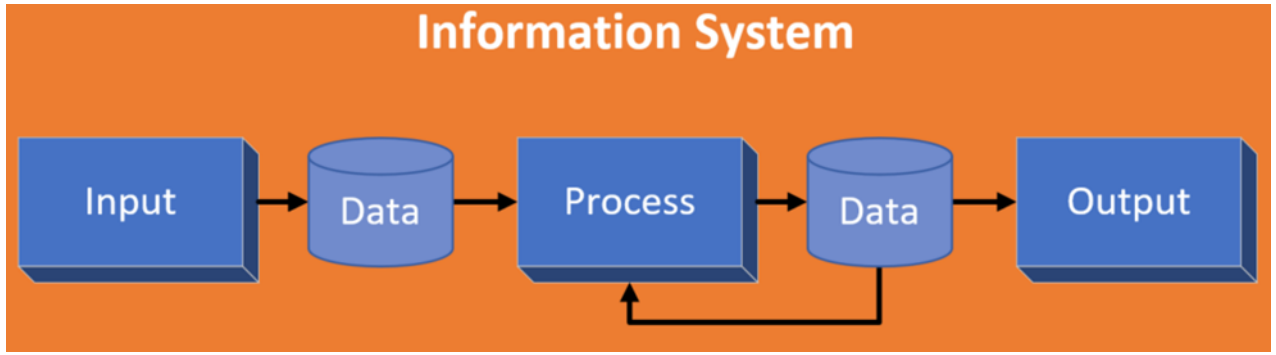
This takes 2 basic forms:

1. Control over the development process itself
2. Ensuring adequate business controls are built into the finished product

Major control stages would include:

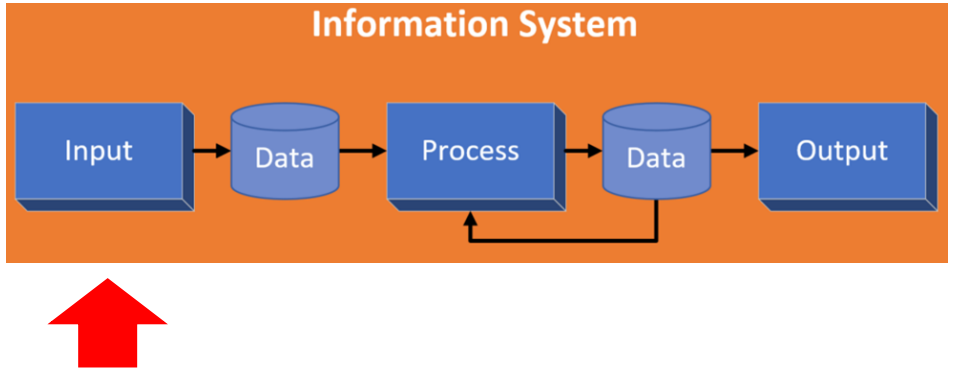
- System design
- System development 
- System operation
- System utilization

Control Objectives for Business Information Systems



1. Input control objectives
2. Processing control objectives
3. Output control objectives

Control Objectives for Business Information Systems



Input control objectives

- All transactions are
  - initially and completely recorded
  - completely and accurately entered into the system
  - entered only once

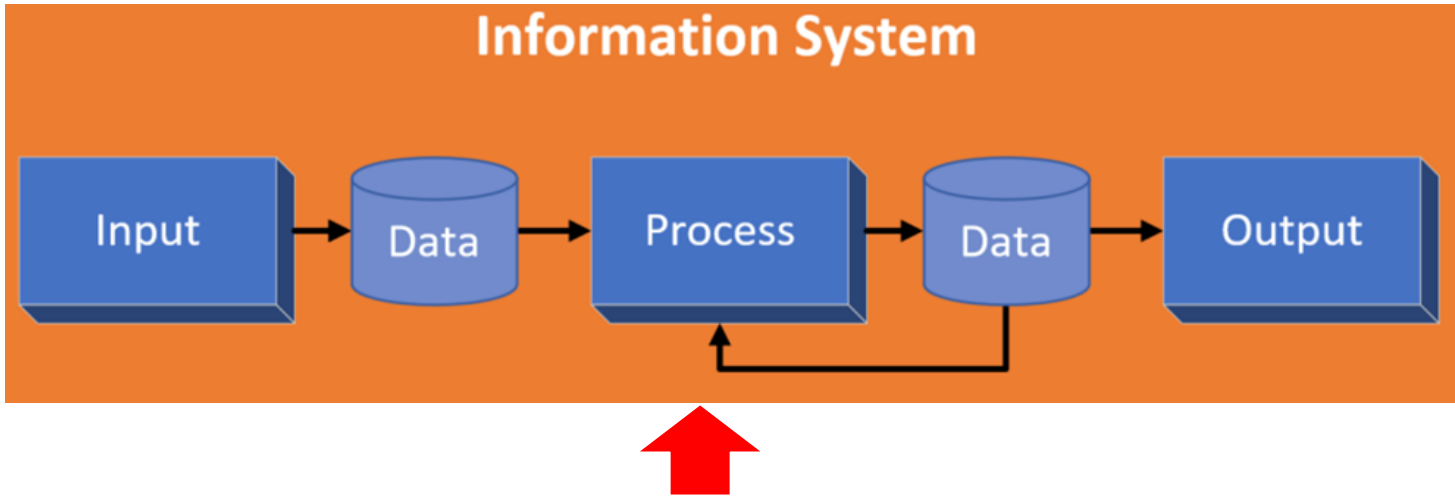
- Controls in this area may include:
  - Pre-numbered documents
  - Control total reconciliation
  - Data validation
  - Activity logging
  - Document scanning and retention for checking
  - Access authorization
  - Document cancellation (e.g. after entry)

Control Objectives for Business Information Systems

Processing control objectives

- Approved transactions are accepted by the system and processed
- All rejected transactions are reported, corrected, and re-input
- All accepted transactions are processed only once
- All transactions are accurately processed
- All transactions are completely processed

- Controls over processing may include:
  - Control totals
  - Programmed balancing
  - Reasonableness tests
  - Segregation of duties
  - Restricted access
  - File labels
  - Exception reports
  - Error logs
  - Concurrent update control

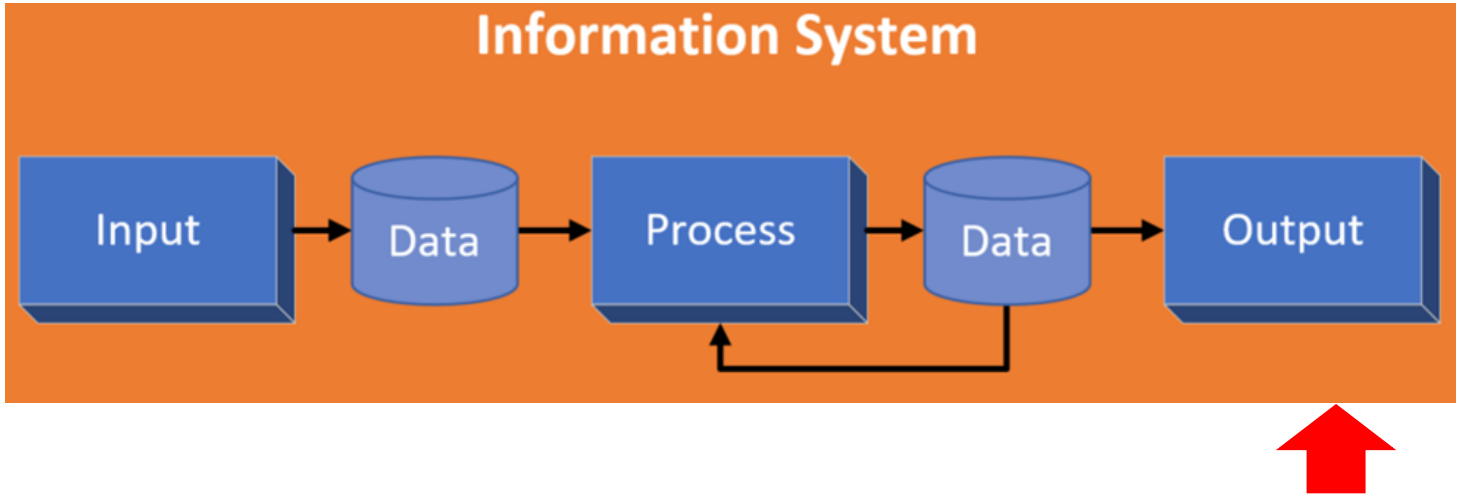


Control Objectives for Business Information Systems

Output control objectives focus on

- Hardcopy
- File outputs and output record sets stored in tables
- Online query files and outputs stored in tables

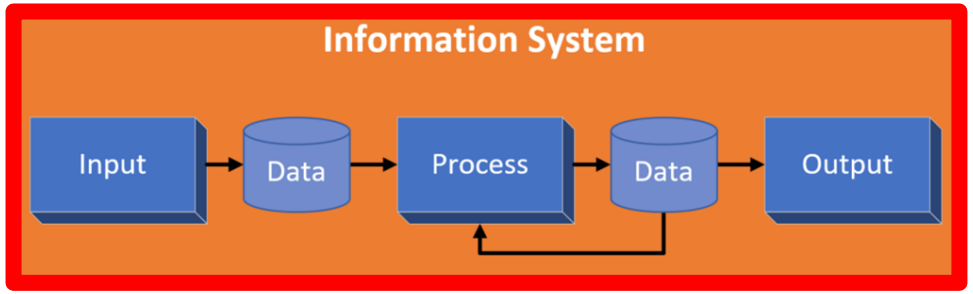
- Controls over output may include:
  - Assurance that the results of input and processing are output
  - Output is available to only authorized personnel
  - Complete audit trail
  - Output distribution logs



Control Objectives for Business Information Systems

Computer program control objectives focus on

- Integrity of programs and processing
- Prevention of unwanted changes



Typical computer program controls include:

- Ensuring adequate design and development
- Ensuring adequate testing
- Controlled transfer of programs (among machines, from version control, …)
- Ongoing maintainability of systems
- Use of formal SDLC
- User involvement
- Adequate documentation
- Formalized testing plan
- Planned conversion
- Use of post-implementation reviews (see CISA chapter)
- Establishment of a quality assurance (QA) function
- Involvement of internal auditors

Testing of these controls require IT auditors to seek evidence regarding their adequacy and effectiveness….

PPTM - People, Processes, Tools, and Measures

A brainstorming framework for examining security of an application from the macro-level, based on

People – describes every aspect of the application that deals with a human

- Make sure the right people are involved in planning, design, implementation or operations, and the right stakeholders are involved
- E.g. If the application involves end users, ensure:
  - The application has controls around providing and removing access
  - End users have been involved with the planning and design of components they will (to ensure usability)

Process – Describes every aspect of the application that is involved in a policy, procedure, method, or course of action

- Review the interaction of the application with interfacing systems and verify compliance with security models
  - E.g. Ensure that firewalls are in place to protect the application from external applications, users, business partners, …
  - Policies and procedures should be written to support how the application is intended to be used
  - Adequate documentation should exist to support technicians who need to maintain the application

Tools – Describe every aspect of the application that deals with concrete technology or product

- Ensure appropriate hardware and environment exist to support the application
- Ensure the application interfaces with recommended technologies appropriate for your intended policies and procedures
- Verify that the application and infrastructure are tested and audited appropriately

Measures – Describe every aspect of the application that is quantifiable conceptually, such as the business purpose or application performance

- E.g. verify that the application meets well-documented and well-thought out acceptance criteria
- E.g. if the application is intended to solve a quantifiable business problem verify that it does indeed solve the problem
- Verify that the logs are meaningful and that you can measure the performance of the application

STRIDE

Threat	Desired property
Spoofing	Authenticity
Tampering	Integrity
Repudiation	Non-repudiability
Information disclosure	Confidentiality
Denial of Service	Availability
Elevation of Privilege	Authorization

A "simplified threat-risk model" which is easy to remember

Spoofing Identity

- Is a key risk for applications with many users and a single execution context at the application and database tiers
- Users should not be able to become any other user or assume the attributes of another user

Tampering with Data

- Data should be stored in a secure location, with access appropriately controlled
- The application should carefully check data received from the user and validate that it is "sane" (i.e. relevant and valid) and applicable before storing or using it
- Data entered in the client (e.g. browser) should be checked and validated on the server and not in the client where the validation checks might be tampered with
- Application should not send and calculate data in the client where the user can manipulate the data, but in the server-side code

Repudiation

- Determine if the application requires nonrepudiation controls, such as web access logs, audit trails at each tier, or the same user context from top to bottom
- Users may dispute transactions if there is insufficient auditing or record-keeping of their activity

Denial of Service

- Application designers should be aware that their applications are at risk of denial of service attacks
- Use of expensive resources (e.g. large files, heavy-duty searches, long queries) should be reserved for authenticated and authorized users and should not be available to anonymous users.
- Every facet of the application should be engineered to perform as little work as possible, to use fast and few database queries, and to avaoid exposing large files or unique links per user to per user to prevent simple denial-of-service attacks

Elevation of Privilege

- If an application provides distinct user and administrative roles, ensure that the user cannot elevate his or her role to a more highly privileged one
- All actions should be controlled through an authorization matrix to ensure that only the permitted roles can access privileged functionality. It is not sufficient, for example, to not display privileged-role links

OWASP (Open Web Application Security Project) Frameworks

- Vulnerabilities

  - API Abuse
  - Authentication Vulnerability
  - Authorization Vulnerability
  - Availability Vulnerability
  - Code Permission Vulnerability
  - Code Quality Vulnerability
  - Configuration Vulnerability
  - Cryptographic Vulnerability
  - Encoding Vulnerability
  - Environmental Vulnerability
  - Error Handling Vulnerability
  - General Logic Error Vulnerability
  - Input Validation Vulnerability
  - Logging and Auditing Vulnerability
  - Password Management Vulnerability
  - Path Vulnerability
  - Sensitive Data Protection Vulnerability
  - Session Management Vulnerability
  - Unsafe Mobile Code
  - Use of Dangerous API

- Principles

  - Apply defense in depth (complete mediation)
  - Use a positive security model (fail-safe defaults, minimize attack surface)
  - Fail securely
  - Run with least privilege
  - Avoid security by obscurity (open design)
  - Keep security simple (verifiable, economy of mechanism)
  - Detect intrusions (compromise recording)
  - Don't trust infrastructure
  - Don't trust services
  - Establish secure defaults (psychological acceptability)

- Top 10 Web Application Security Risks

Application Security Testing

Static application security testing (SAST)

- Can be thought of as testing the application from the inside out
- By examining its source code, byte code or application binaries for conditions indicative of a security vulnerability

Dynamic application security testing (DAST)

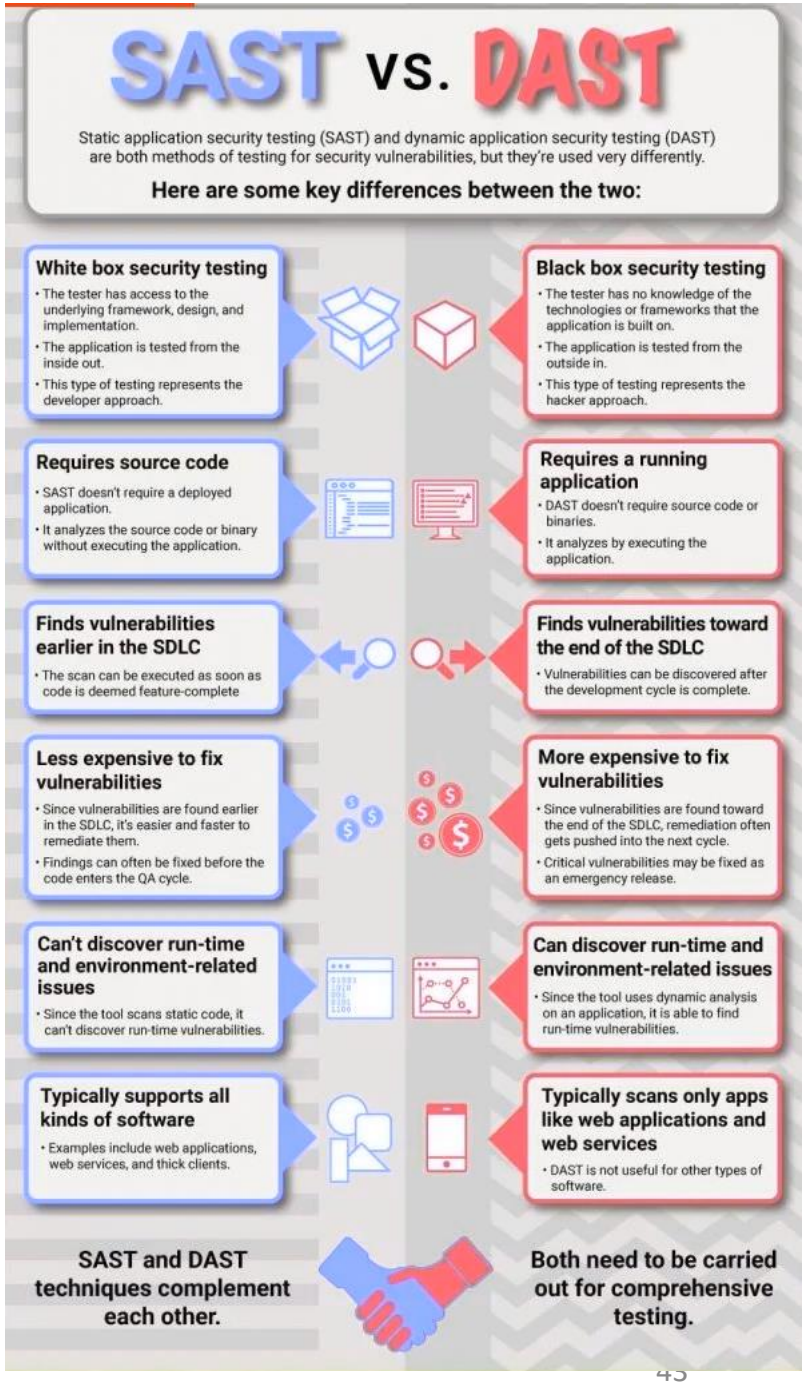
- Can be thought of as testing the application from the outside in
- By examining the application in its running state, and trying to poke it and prod it in unexpected ways in order to discover security vulnerabilities

Interactive application security testing (IAST)

- Can be thought of as testing the application from the outside in
- By examining the application in its running state, and trying to poke it and prod it in unexpected ways in order to discover security vulnerabilities

Software Composition Analysis (SCA)

- Software Composition (or Component) Analysis is the process of identifying potential areas of risk from the use of third-party and open-source software components
- SCA is a form of Cyber Supply Chain Risk Management

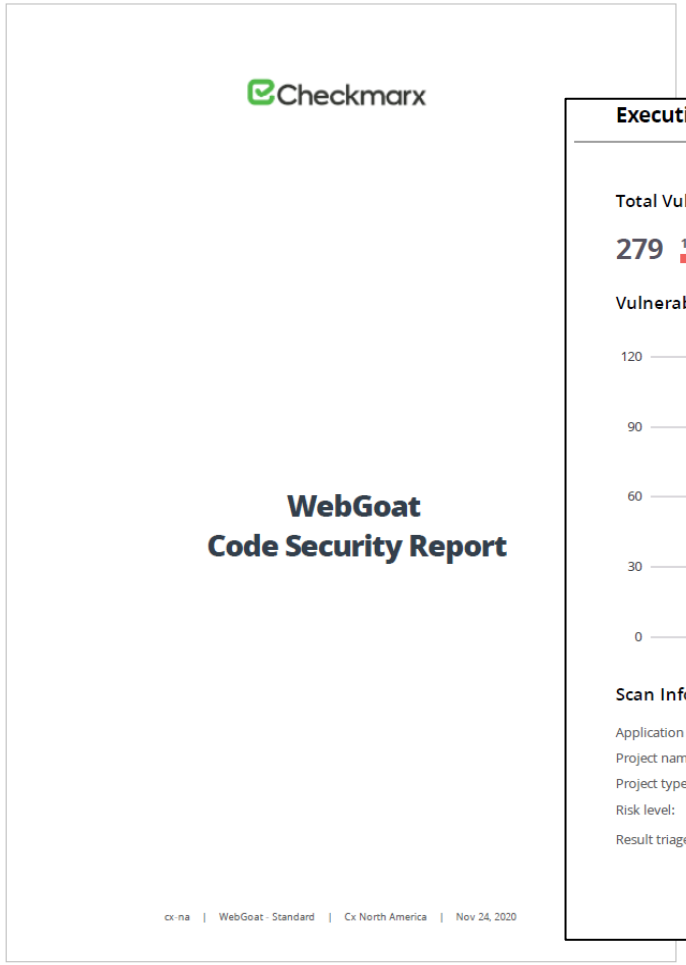


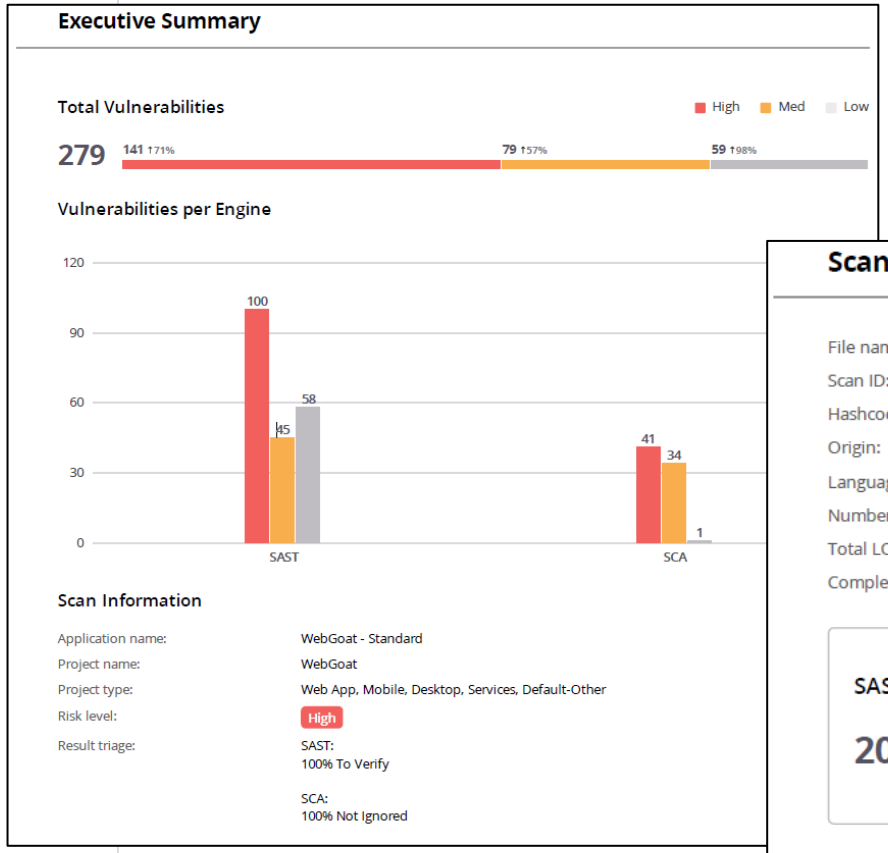
Automated application security testing tools

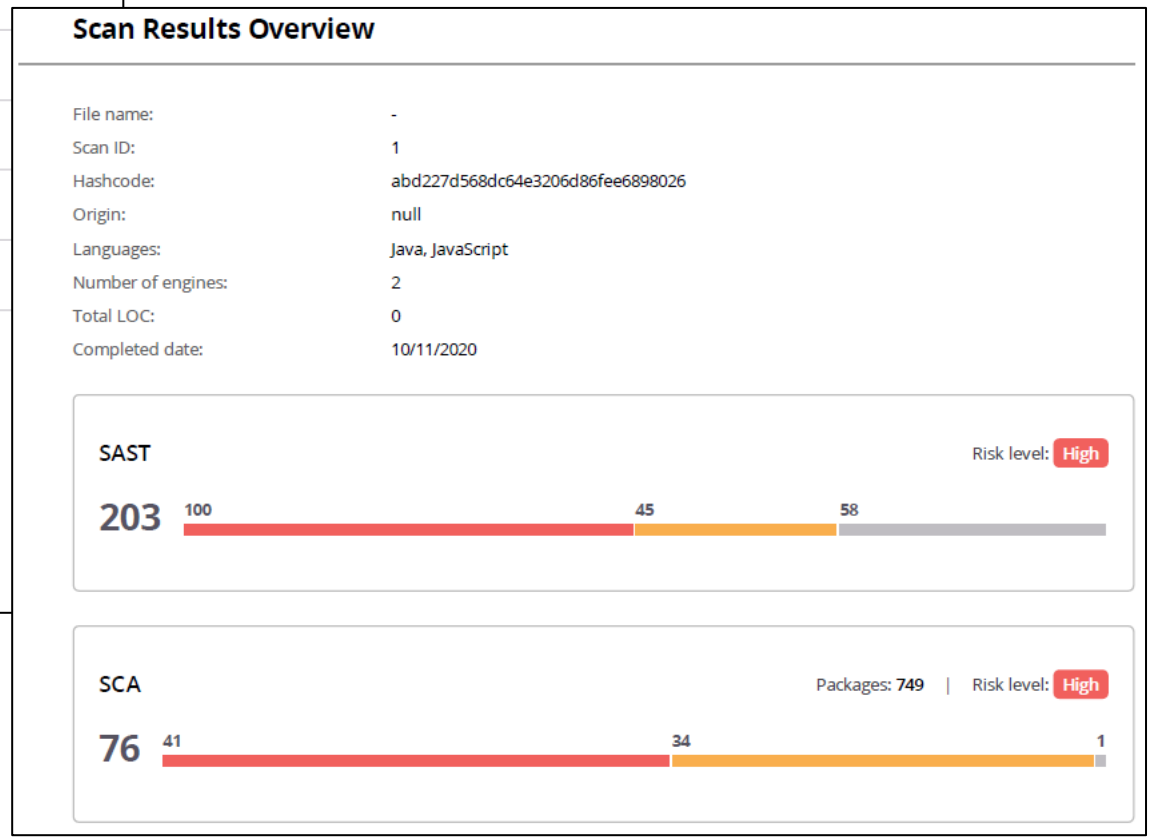
Some vendors provide SAST tools, others provide DAST tools, others provide SCA tools

Some vendors provide combinations of these tools

Automated application security testing tools provide vulnerability reports







SAST Compliance Report Examples

OWASP Top 10 2013

Vulnerabilities: **152**

Top 10 vulnerability types:

1. Reflected_XSS_All_Clients (39)
2. SQL_Injection (23)
3. Client_DOM_Open_Redirect (16)
4. Stored_XSS (10)
5. XSRF (9)
6. Use_of_Cryptographically_Weak_PRNG (8)
7. Heap_Inspection (8)
8. Use_of_Hard_coded_Cryptographic_Key (8)
9. Client_JQuery_Deprecated_Symbols (7)
10. Use_Of_Hardcoded_Password (7)

OWASP Top 10 2017

Vulnerabilities: **154**

Top 10 vulnerability types:

1. Reflected_XSS_All_Clients (39)
2. SQL_Injection (23)
3. Use_Of_Hardcoded_Password (13)
4. Stored_XSS (10)
5. Use_of_Hard_coded_Cryptographic_Key (8)
6. Use_of_Cryptographically_Weak_PRNG (8)
7. Heap_Inspection (8)
8. Use_Of_Hardcoded_Password (7)
9. Log_Forging (7)
10. Client_JQuery_Deprecated_Symbols (7)

PCI DSS v3.2

Vulnerabilities: **126**

Top 10 vulnerability types:

1. Reflected_XSS_All_Clients (39)
2. SQL_Injection (23)
3. Stored_XSS (10)
4. XSRF (9)
5. Use_of_Hard_coded_Cryptographic_Key (8)
6. Use_of_Cryptographically_Weak_PRNG (8)
7. Log_Forging (7)
8. Use_Of_Hardcoded_Password (7)
9. Client_Potential_XSS (3)
10. HttpOnlyCookies (3)

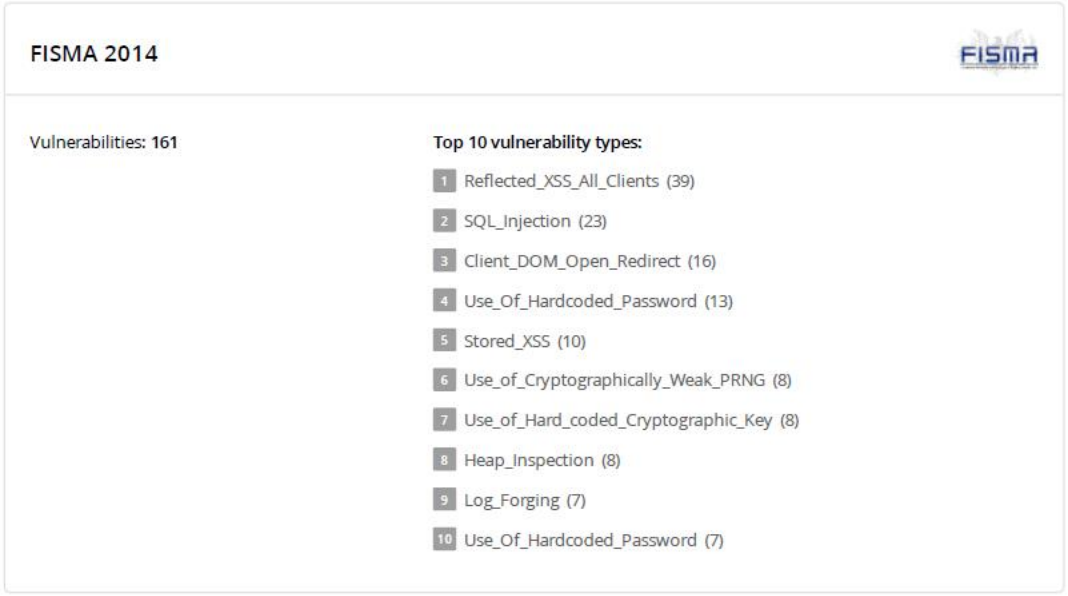
OWASP Mobile Top 10 2016

Vulnerabilities: **66**

Top 10 vulnerability types:

1. SQL_Injection (23)
2. Side_Channel_Data_Leakage (17)
3. Use_of_Hard_coded_Cryptographic_Key (8)
4. Log_Forging (7)
5. Use_Of_Hardcoded_Password (7)
6. Inadequate_Encryption_Strength (2)
7. Deserialization_of_Untrusted_Data (2)

SAST Compliance Report Examples

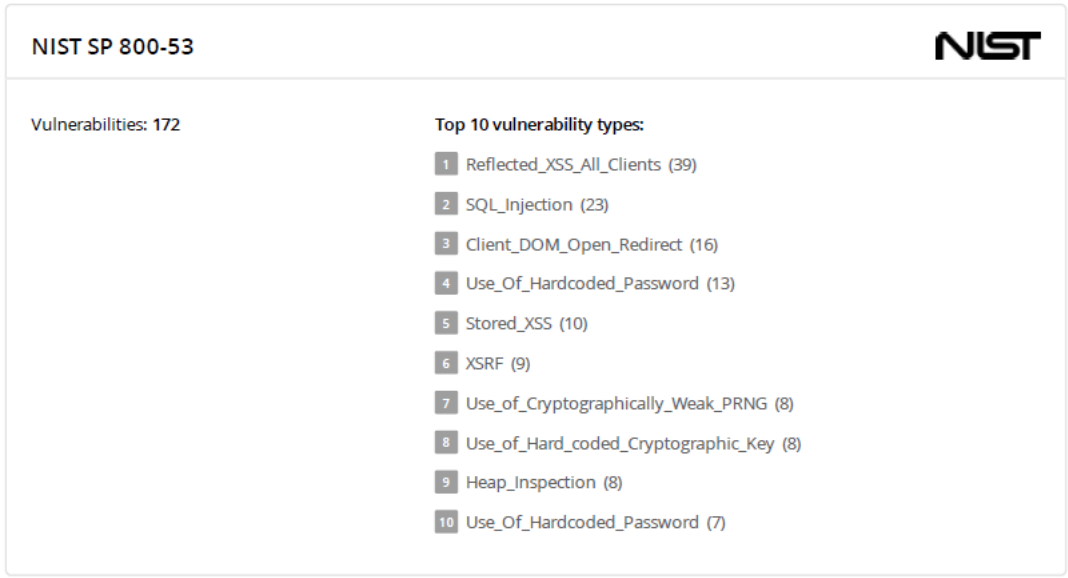


FISMA 2014

Vulnerabilities: **161**

Top 10 vulnerability types:

1. Reflected_XSS_All_Clients (39)
2. SQL_Injection (23)
3. Client_DOM_Open_Redirect (16)
4. Use_Of_Hardcoded_Password (13)
5. Stored_XSS (10)
6. Use_of_Cryptographically_Weak_PRNG (8)
7. Use_of_Hard_coded_Cryptographic_Key (8)
8. Heap_Inspection (8)
9. Log_Forging (7)
10. Use_Of_Hardcoded_Password (7)



NIST SP 800-53

Vulnerabilities: **172**

Top 10 vulnerability types:

1. Reflected_XSS_All_Clients (39)
2. SQL_Injection (23)
3. Client_DOM_Open_Redirect (16)
4. Use_Of_Hardcoded_Password (13)
5. Stored_XSS (10)
6. XSRF (9)
7. Use_of_Cryptographically_Weak_PRNG (8)
8. Use_of_Hard_coded_Cryptographic_Key (8)
9. Heap_Inspection (8)
10. Use_Of_Hardcoded_Password (7)

SAST Report Details

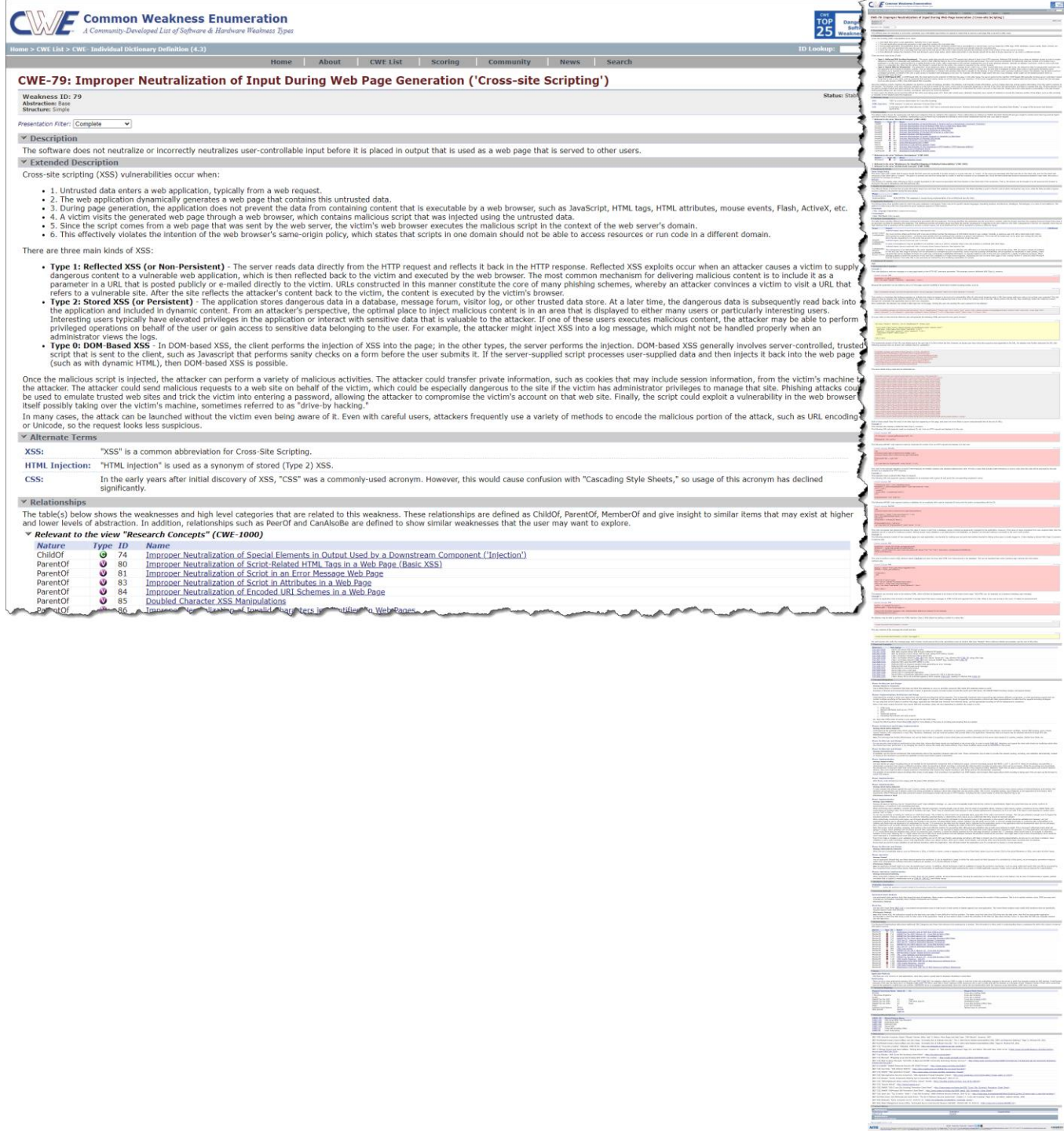
JavaScript

Client_DOM_XSS (1)

A successful XSS exploit would allow an attacker to rewrite web pages and insert malicious scripts which would alter the intended output. This could include HTML fragments, CSS styling rules, arbitrary JavaScript, or references to third party code. An attacker could use this to steal users' passwords, collect personal data such as credit card details, provide false information, or run malware. From the victim's point of view, this is performed by the genuine website, and the victim would blame the site for incurred damage. An additional risk with DOM XSS is that, unlike reflected or stored XSS, tainted values do not have to go through the server. Since the server is not involved in sanitization of these inputs, server-side validation is not likely to not be aware XSS attacks have been occurring, and any server-side security solutions, such as a WAF, are likely to be ineffective in DOM XSS mitigation.

New | 244648 | Row 1

State:	**To Verify**
Source node:	**location**
Source file:	**/WebGoat-develop/webgoat-container/src/main/resources/static/js/libs/backbone-min.js**
Sink node:	**location**
Sink file:	**/WebGoat-develop/webgoat-container/src/main/resources/static/js/libs/backbone-min.js**
Compliances:	**OWASP Top 10 2013, OWASP Top 10 2017, PCI DSS v3.2, FISMA 2014, NIST SP 800-53**
CWE:	**CWE-79**
Notes:	-



Common Weakness Enumeration
A Community-Developed List of Software & Hardware Weakness Types

CWE-79: Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')

Description

The software does not neutralize or incorrectly neutralizes user-controllable input before it is placed in output that is used as a web page that is served to other users.

Extended Description

Cross-site scripting (XSS) vulnerabilities occur when:

1. Untrusted data enters a web application, typically from a web request.
2. The web application dynamically generates a web page that contains this untrusted data.
3. During page generation, the application does not prevent the data from containing content that is executable by a web browser, such as JavaScript, HTML tags, HTML attributes, mouse events, Flash, ActiveX, etc.
4. A victim visits the generated web page through a web browser, which contains malicious script that was injected using the untrusted data.
5. Since the script comes from a web page that was sent by the web server, the victim's web browser executes the malicious script in the context of the web server's domain.
6. This effectively violates the intention of the web browser's same-origin policy, which states that scripts in one domain should not be able to access resources or run code in a different domain.

There are three main kinds of XSS:

- **Type 1: Reflected XSS (or Non-Persistent)** - The server reads data directly from the HTTP request and reflects it back in the HTTP response. Reflected XSS exploits occur when an attacker causes a victim to supply dangerous content to a vulnerable web application, which is then reflected back to the victim and executed by the web browser. The most common mechanism for delivering malicious content is to include it as a parameter in a URL that is posted publicly or e-mailed directly to the victim. URLs constructed in this manner constitute the core of many phishing schemes, whereby an attacker convinces a victim to visit a URL that refers to a vulnerable site. After the site reflects the attacker's content back to the victim, the content is executed by the victim's browser.
- **Type 2: Stored XSS (or Persistent)** - The application stores dangerous data in a database, message forum, visitor log, or other trusted data store. At a later time, the dangerous data is subsequently read back into the application and included in dynamic content. From an attacker's perspective, the optimal place to inject malicious content is in an area that is displayed to either many users or particularly interesting users. Interesting users typically have elevated privileges in the application or interact with sensitive data that is valuable to the attacker. If one of these users executes malicious content, the attacker may be able to perform privileged operations on behalf of the user or gain access to sensitive data belonging to the user. For example, the attacker might inject XSS into a log message, which might not be handled properly when an administrator views the logs.
- **Type 0: DOM-Based XSS** - In DOM-based XSS, the client performs the injection of XSS into the page; in the other types, the server performs the injection. DOM-based XSS generally involves server-controlled, trusted script that is sent to the client, such as Javascript that performs sanity checks on a form before the user submits it. If the server-supplied script processes user-supplied data and then injects it back into the web page (such as with dynamic HTML), then DOM-based XSS is possible.

Once the malicious script is injected, the attacker can perform a variety of malicious activities. The attacker could transfer private information, such as cookies that may include session information, from the victim's machine to the attacker. The attacker could send malicious requests to a web site on behalf of the victim, which could be especially dangerous to the site if the victim has administrator privileges to manage that site. Phishing attacks could be used to emulate trusted web sites and trick the victim into entering a password, allowing the attacker to compromise the victim's account on that web site. Finally, the script could exploit a vulnerability in the web browser itself possibly taking over the victim's machine, sometimes referred to as "drive-by hacking."

In many cases, the attack can be launched without the victim even being aware of it. Even with careful users, attackers frequently use a variety of methods to encode the malicious portion of the attack, such as URL encoding or Unicode, so the request looks less suspicious.

Alternate Terms

XSS:	"XSS" is a common abbreviation for Cross-Site Scripting.
HTML Injection:	"HTML injection" is used as a synonym of stored (Type 2) XSS.
CSS:	In the early years after initial discovery of XSS, "CSS" was a commonly-used acronym. However, this would cause confusion with "Cascading Style Sheets," so usage of this acronym has declined significantly.

Relationships

The table(s) below shows the weaknesses and high level categories that are related to this weakness. These relationships are defined as ChildOf, ParentOf, MemberOf and give insight to similar items that may exist at higher and lower levels of abstraction. In addition, relationships such as PeerOf and CanAlsoBe are defined to show similar weaknesses that the user may want to explore.

Relevant to the view "Research Concepts" (CWE-1000)

Nature	Type	ID	Name
ChildOf	B	74	Improper Neutralization of Special Elements in Output Used by a Downstream Component ('Injection')
ParentOf	V	80	Improper Neutralization of Script-Related HTML Tags in a Web Page (Basic XSS)
ParentOf	V	81	Improper Neutralization of Script in an Error Message Web Page
ParentOf	V	83	Improper Neutralization of Script in Attributes in a Web Page
ParentOf	V	84	Improper Neutralization of Encoded URI Schemes in a Web Page
ParentOf	V	85	Doubled Character XSS Manipulations
ParentOf	V	86	Improper Neutralization of Invalid Characters in Identifiers in Web Pages

DAST Report

TINFOIL SECURITY

June 17, 2020
http://demo.testfire.net

Security Analysis - June 17, 2020

SCAN SUMMARY

This site was checked for **65** classes of vulnerabilities, with up to hundreds of tests for each vulnerability class. This site is considered to be **Very Unsafe** as of **June 17, 2020**.

VULNERABILITY CLASSES

The following types of vulnerabilities were looked for over the **27** URLs found during this security scan.

Allowed HTTP methods	ASP.NET DEBUG Method Enabled
Blind SQL Injection (timing attack)	Buffer Overflow
Clickjacking	Code Injection
Credit card number disclosure	Cross-Site Request Forgery
Cross-Site Scripting in attribute of HTML element	Cross-Site Scripting in event attribute of HTML element
Cross-Site Scripting in HTML "script" tag	Cross-Site Scripting in HTML tag
Cross-Site Scripting in HTML "vbscript" tag	Cross-Site Scripting (XSS)
Cross-Site Scripting (XSS) in path	CVS/SVN user disclosure
Directory listing is enabled.	Disclosed e-mail address
Disclosed US Social Security Number	DOM Based Cross-Site Scripting
File Inclusion	Found a CAPTCHA protected form
Found an HTML object	Found Robots.txt
Found Stacktrace	FrontPage Extensions Enabled
HTTP PUT is enabled	Insecure Cookies
LDAP Injection	Misconfiguration in LIMIT directive of .htaccess file
Missing Subresource Integrity Protection	Mixed Resource
Non HTTP-Only Cookies	OpenSSL Heartbeat Extension Memory Leak (Heartbleed)
Operating system command injection	Outdated TLS Supported
Password field with autocomplete	Password Submission via GET
Path Traversal	Permissive CORS Policy
Persistent Cross-Site Scripting (XSS)	Private IP address disclosure
Remote file inclusion	Response splitting
Scriptless Cross-Site Scripting in attribute of HTML element	Server-Side Include Injection
Session Cookie Expiration	Session Fixation
Session ID Entropy	Shellshock
Spammable contact form	SQL Injection
SSLv3 Enabled	Strutshock (CVE-2017-5638)
The TRACE HTTP method is enabled	TLS Fallback is not Supported
TLS Vulnerable to POODLE	Unencrypted HTTP Basic Authentication
Unencrypted password form	Unvalidated redirect
WebDAV	XML External Entity Injection
XPath Injection	YAML Injection
YAML Injection (timing)	

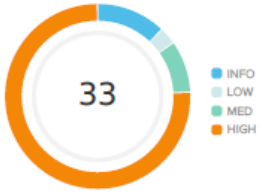
SCAN OVERVIEW

STATUS ON 06/17/2020

Your website is

Very Unsafe

NUMBER OF VULNERABILITIES



Total Vulnerabilities

WHAT'S THE WORST THAT COULD HAPPEN?

With your current vulnerabilities a hacker could potentially infiltrate your website, steal your user's cookies, log the keys they type, and pretend to be them on your website. And that's just the tip of the iceberg. Data breaches like this, once disclosed, can often lead to a 20% loss in your customer base. We highly recommend you fix these vulnerabilities quickly and with much vengeance.

LOGIN STATUS

Login Successful: Yes

SITEMAP

http://demo.testfire.net/
http://demo.testfire.net/admin/admin.jsp
http://demo.testfire.net/bank/apply.jsp
http://demo.testfire.net/bank/ccApply
http://demo.testfire.net/bank/customize.jsp
http://demo.testfire.net/bank/doTransfer
http://demo.testfire.net/bank/main.jsp
http://demo.testfire.net/bank/queryxpath.jsp
http://demo.testfire.net/bank/showAccount
http://demo.testfire.net/bank/showTransactions
http://demo.testfire.net/bank/transaction.jsp
http://demo.testfire.net/bank/transfer.jsp
http://demo.testfire.net/default.jsp
http://demo.testfire.net/disclaimer.htm
http://demo.testfire.net/doSubscribe
http://demo.testfire.net/feedback.jsp
http://demo.testfire.net/index.jsp
http://demo.testfire.net/search.jsp
http://demo.testfire.net/sendFeedback
http://demo.testfire.net/status_check.jsp

VULNERABILITY: CROSS-SITE REQUEST FORGERY

DETAILS

Severity	High
URL	http://demo.testfire.net/admin/admin.jsp
Variable	addAccount
Element	form

INJECTION

Matched by Regular Expression

```
<form id="addAccount" name="addAccount" action="" method="post"> <tr> <td colspan="4"> <h2>Add an account to an existing user</h2> </td> </tr> <tr> <th> Users: </th> <th> Account Types: </th> <th> </th> <th> </th> </tr> <tr> <td> <select name="username" id="username" size="1"> <option value="admin">admin</option> <option value="jdoe">jdoe</option> <option value="jsmith">jsmith</option> <option value="sspeed">sspeed</option> <option value="tuser">tuser</option> </select> </td> <td> <select name="accttypes"> <option value="Checking">Checking</option> <option value="Savings" selected>Savings</option> <option value="IRA">IRA</option> </select> </td> <td></td> <td><input type="submit" value="Add Account"></td> </tr> </form>
```

DESCRIPTION

Cross-Site Request Forgery (CSRF) allows an attacker to execute actions on behalf of an unwitting user who is already authenticated with your web application. If successful, user data and user actions can be compromised. If the user who is attacked with CSRF happens to be an administrator, the entire web application should be considered compromised. CSRF occurs when a user submits data to a form or input he/she did not intend; usually an attacker will accomplish this by sending them a link or convincing them to input to a different form that looks similar and posts to the same place.

HOW TO FIX

A unique token that guarantees freshness of submitted data must be added to all web application elements that can affect business logic.

REFERENCES

Wikipedia - http://en.wikipedia.org/wiki/Cross-site_request_forgery

CGI Security - http://www.cgisecurity.com/csrf-faq.html

OWASP - https://wiki.owasp.org/index.php/Cross-Site_Request_Forgery_(CSRF)

Application Security Assessment and Recommendations

Issue Types 21

TOC

	Issue Type	Number of Issues
H	Authentication Bypass Using HTTP Verb Tampering	3
H	Cross-Site Request Forgery	23
H	Cross-Site Scripting	2
H	Microsoft FrontPage Extensions Site Defacement	3
H	Missing Secure Attribute in Encrypted Session (SSL) Cookie	5
H	RC4 cipher suites were detected	1
M	Alternate Version of File Detected	45
M	Body Parameters Accepted in Query	9
M	Browser Exploit Against SSL/TLS (a.k.a. BEAST)	1
M	Cacheable SSL Page Found	67
M	Direct Access to Administration Pages	1
M	Drupal "keys" Path Disclosure	1
M	Insecure "OPTIONS" HTTP Method Enabled	1
M	Microsoft FrontPage Server Extensions Vital Information Leakage	2
M	Microsoft IIS Missing Host Header Information Leakage	1
M	Missing "Content-Security-Policy" header	5
M	Missing Cross-Frame Scripting Defence	4
M	Query Parameter in SSL Request	185
M	Temporary File Download	3
M	Unencrypted __VIEWSTATE Parameter	20
M	Web Application Source Code Disclosure Pattern Found	1

Fix Recommendations 19

TOC

	Remediation Task	Number of Issues
H	Review possible solutions for hazardous character injection	2
M	Add the 'Secure' attribute to all sensitive cookies	5
M	Change server's supported ciphersuites	2
M	Configure your server to allow only required HTTP methods	3
M	Set proper permissions to the FrontPage extension files	3
M	Validate the value of the "Referer" header, and use a one-time-nonce for each submitted form	23
L	Always use SSL and POST (body) parameters when sending sensitive information.	185
L	Apply configuration changes according to Q218180	1
L	Apply proper authorization to administration scripts	1
L	Config your server to use the "Content-Security-Policy" header	5
L	Config your server to use the "X-Frame-Options" header	4
L	Contact the vendor of your product to see if a patch or a fix has been made available recently	1
L	Disable WebDAV, or disallow unneeded HTTP methods	1
L	Do not accept body parameters that are sent in the query string	9
L	Modify FrontPage extension file permissions to avoid information leakage	2
L	Modify your Web.Config file to encrypt the VIEWSTATE parameter	20
L	Prevent caching of SSL pages by adding "Cache-Control: no-store" and "Pragma: no-cache" headers to their responses.	67
L	Remove old versions of files from the virtual directory	48
L	Remove source code files from your web-server and apply any relevant patches	1

This report contains the results of a web application security scan performed by IBM Security AppScan Standard.

High severity issues:	79
Medium severity issues:	198
Total security issues included in the report:	277
Total security issues discovered in the scan:	308

Issues Sorted by Issue Type

- Authentication Bypass Using SQL Injection 2
- Blind SQL Injection 4
- Cross-Site Request Forgery 24
- Cross-Site Scripting 3
- HTTP PUT Method Site Defacement 20
- Inadequate Account Lockout 1
- Microsoft FrontPage Extensions Site Defacement 3
- Missing Secure Attribute in Encrypted Session (SSL) Cookie 1
- Phishing Through URL Redirection 1
- WebDAV MKCOL Method Site Defacement 20
- Alternate Version of File Detected 50
- Cacheable SSL Page Found 26
- Hidden Directory Detected 7
- Microsoft FrontPage Configuration Information Leakage 1
- Microsoft FrontPage Server Extensions Vital Information Leakage 2
- Microsoft IIS Missing Host Header Information Leakage 1
- Query Parameter in SSL Request 66
- Temporary File Download 32
- Unencrypted __VIEWSTATE Parameter 11
- Web Application Source Code Disclosure Pattern Found 2

Application Security Vulnerability Assessment Report

AppScan example

Advisories

- Authentication Bypass Using SQL Injection
- Blind SQL Injection
- Cross-Site Request Forgery
- Cross-Site Scripting
- HTTP PUT Method Site Defacement
- Inadequate Account Lockout
- Microsoft FrontPage Extensions Site Defacement
- Missing Secure Attribute in Encrypted Session (SSL) Cookie
- Phishing Through URL Redirection
- WebDAV MKCOL Method Site Defacement
- Alternate Version of File Detected
- Cacheable SSL Page Found
- Hidden Directory Detected
- Microsoft FrontPage Configuration Information Leakage
- Microsoft FrontPage Server Extensions Vital Information Leaka
- Microsoft IIS Missing Host Header Information Leakage
- Query Parameter in SSL Request
- Temporary File Download
- Unencrypted __VIEWSTATE Parameter
- Web Application Source Code Disclosure Pattern Found

H Authentication Bypass Using SQL Injection 2 TOC

Issue 1 of 2 TOC

Authentication Bypass Using SQL Injection	
Severity:	High
URL:	https://www.r...
Entity:	UserName (Parameter)
Risk:	It may be possible to bypass the web application's authentication mechanism
Causes:	Sanitation of hazardous characters was not performed correctly on user input
Fix:	Review possible solutions for hazardous character injection

Reasoning: The test result seems to indicate a vulnerability because when four types of request were sent - a valid login, an invalid login, an SQL attack, and another invalid login - the responses to the two invalid logins were the same, while the response to the SQL attack seems similar the response to the valid login.

Issue 2 of 2 TOC

Authentication Bypass Using SQL Injection	
Severity:	High
URL:	https://www. ...x
Entity:	Password (Parameter)
Risk:	It may be possible to bypass the web application's authentication mechanism
Causes:	Sanitation of hazardous characters was not performed correctly on user input
Fix:	Review possible solutions for hazardous character injection

Reasoning: The test result seems to indicate a vulnerability because when four types of request were sent - a valid login, an invalid login, an SQL attack, and another invalid login - the responses to the two invalid logins were the same, while the response to the SQL attack seems similar the response to the valid login.

Authentication Bypass Using SQL Injection

TOC

Test Type:

Application-level test

Threat Classification:

Insufficient Authentication

Causes:

Sanitation of hazardous characters was not performed correctly on user input

Security Risks:

It may be possible to bypass the web application's authentication mechanism

Affected Products:

CWE:

566

References:

"Web Application Disassembly with ODBC Error Messages" (By David Litchfield)
SQL Injection Training Module

Technical Description:

The application uses a protection mechanism that relies on the existence or values of an input, but the input can be modified by an untrusted user in a way that bypasses the protection mechanism.

When security decisions such as authentication and authorization are made based on the values of user input, attackers can bypass the security of the software.

Suppose the query in question is:

```
SELECT COUNT(*) FROM accounts WHERE username='$user' AND password='$pass'
```

Where $user and $pass are user input (collected from the HTTP request which invoked the script that constructs the query - either from a GET request query parameters, or from a POST request body parameters). A regular usage of this query would be with values $user=john, $password=secret123. The query formed would be:

```
SELECT COUNT(*) FROM accounts WHERE username='john' AND password='secret123'
```

The expected query result is 0 if no such user+password pair exists in the database, and >0 if such pair exists (i.e. there is a user named 'john' in the database, whose password is 'secret123'). This would serve as a basic authentication mechanism for the application. But an attacker can bypass this mechanism by submitting the following values: $user=john, $password=' OR '1'='1.

Technical Description:

The application uses a protection mechanism that relies on the existence or values of an input, but the input can be modified by an untrusted user in a way that bypasses the protection mechanism.

When security decisions such as authentication and authorization are made based on the values of user input, attackers can bypass the security of the software.

Suppose the query in question is:

```
SELECT COUNT(*) FROM accounts WHERE username='$user' AND password='$pass'
```

Where $user and $pass are user input (collected from the HTTP request which invoked the script that constructs the query - either from a GET request query parameters, or from a POST request body parameters). A regular usage of this query would be with values $user=john, $password=secret123. The query formed would be:

```
SELECT COUNT(*) FROM accounts WHERE username='john' AND password='secret123'
```

The expected query result is 0 if no such user+password pair exists in the database, and >0 if such pair exists (i.e. there is a user named 'john' in the database, whose password is 'secret123'). This would serve as a basic authentication mechanism for the application. But an attacker can bypass this mechanism by submitting the following values: $user=john, $password=' OR '1'='1.

The resulting query is:

```
SELECT COUNT(*) FROM accounts WHERE username='john' AND password='' OR '1'='1'
```

This means that the query (in the SQL database) will return TRUE for the user 'john', since the expression 1=1 is always true. Therefore, the query will return a positive number, and thus the user (attacker) will be considered valid without having to know the password.

Agenda

- ✓Midterm Exam Review
- ✓Team Project Guidance
- ✓Distributed Systems
  - ✓File Server Architecture
  - ✓Client/Server Architecture
  - ✓N-Tier Architecture
  - ✓Cloud Architecture
  - ✓Service Oriented Architecture (SOA)
- ✓Example Cloud-based N-Tier SOA Application Development System
- ✓Control Stages, Objectives, Application Security Testing
- Additional Best Practices

Additional best practices for secure application development

1. Defense-in-Depth
2. Positive Security Model
3. Fail Safely
4. Run with Least Privilege
5. Avoid Security by Obscurity
6. Keep Security Simple
7. Use Open Standards
8. Keep, manage and analyze logs to detect Intrusions
9. Never Trust External Infrastructure and Services
10. Establish Secure Defaults

Characteristics which can help in quickly spotting common weaknesses and poor controls

Defense In Depth

Layered approaches provide more security over the long term than one complicated mass of security architecture

- **Sequences of routers, firewalls and intrusion detection/protection monitoring devices used to examine data packets, reduce undesired traffic and protect the inner information systems**
- **Access Control Lists (ACLs),** for example, on the networking routers and firewall equipment to allow only necessary traffic to reach the application

    - *Quickly eliminating access to services, ports, and protocols significantly lowers the overall risk of compromise to the system on which the application is running*

Positive Security Model

- Positive security models use “whitelist” to allow only what is on the list, excluding everything else by default
  - “Deny by default”
  - A challenge for antivirus programs
- In contrast with negative (blacklist) security models that allow everything by default, eliminating only the items known to be bad
  - Problems:
    - Blacklist must be kept up to date
    - Even if blacklist is updated, an unknown vulnerability can still exist
    - Attack surface is much larger than with a positive security model

Fail Safely

- An application failure can be dealt with in one of 3 ways:
  - Allow
  - Block
  - Error
- In general, application errors should all fail in the same way:
  - Disallow the operation (as viewed by the user) and provide no or minimal information on the failure
  - Do not provide the end user with additional information that may help in compromising the system
    - Put the error information in the logs, but do not provide to the user to use in compromising the system

Run with Least Privilege

- Principle of Least Privilege mandates that accounts have the least amount of privilege possible to perform their activity
- This includes:
  - User rights
  - Resource permissions such as CPU limits, memory capacity, network bandwidth, file system permissions, and database permissions

Avoid Security by Obscurity

- Obfuscating data (hiding it) instead of encrypting it is a very weak security mechanism
  - If a human can figure out how to hide the data a human can learn how to recover the data
- Never obfuscate critical data that can be encrypted or never stored in the first place

Keep Security Simple

- Simple security mechanisms are easy to verify and easy to implement correctly
- Avoid complex security mechanisms if possible
  - *“The quickest method to break a cryptographic algorithm is to go around it”*
- Do not confuse complexity with layers: Layers are good; complexity isn’t

Use Open Standards

- Open security standards provide increased portability and interoperability
- IT infrastructure is often a heterogeneous mix of platforms, open standards helps ensure compatibility between systems as the application grows
- Open standards are often well known and scrutinized by peers in the security industry to ensure they remain secure

Keep, manage and analyze logs to help detect intrusions

- Applications should have built-in logging that is protected and easily read
- Logs help you troubleshoot issues, and just as important – help you to track down when or how an application might have been compromised

Never Trust External Infrastructure and Services

- Many organizations use the processing capabilities of third-party partners that more than likely have differing security policies and postures than your organization
- It is unlikely that you can influence or control an external third party
- Implicitly trusting externally run systems is dangerous!

Establish Secure Defaults

- New applications should arrive or be presented to users with the most secure default settings possible that still allow business to function
- This may require training end users or communications messages
- End result is a significantly reduced attack surface
  - *Especially when application is pushed out across a large population*

Test Areas for Auditing Applications

1. Input Controls, Process Controls, and Output Controls
   - Review and evaluate controls built into system transactions for i data
   - Determine the need for error/exception reports related to data integrity and evaluate whether this need has been filled
2. Interface Controls
   - Review and evaluate the controls in place over data feeds to and from interfacing systems
   - If the same data is kept in multiple databases and/or systems, ensure that periodic sync processes are executed to detect any inconsistencies in the data
3. Audit Trails
   - Review and evaluate the audit trails present in the system and the controls over those audit trails
   - Ensure that the system provides a means of tracing a transaction or piece of data from the beginning to the end of the process enabled by the system

Test Areas for Auditing Applications

4. Software Change Controls
   - Ensure that the application software cannot be changed without going through a standard checkout/staging/testing/approval process after it is placed into production
   - Evaluate controls regarding code checkout and versioning
   - Evaluate controls regarding the testing of application code before it is placed into a production environment
   - Evaluate controls regarding batch scheduling
5. Backup and Recovery
   - Determine whether a Business Impact Analysis (BIA) has been performed on the application to establish backup and recovery needs
   - Ensure that appropriate backup and recovery controls are in place
   - Ensure appropriate recovery controls are in place

Test Areas for Auditing Applications

6. Data Retention and User Involvement
   - Evaluate controls regarding the application's data retention
   - Evaluate overall user involvement and support for the Application
7. Identity, Authentication, and Access Controls…
8. Host Hardening…

Agenda

- ✓Midterm Exam Review
- ✓Team Project Guidance
- Distributed Systems
  - File Server Architecture
  - Client/Server Architecture
  - N-Tier Architecture
  - Cloud Architecture
  - Service Oriented Architecture (SOA)
- Example Cloud-based N-Tier SOA Application Development System
- Control Stages, Objectives, Application Security Testing
- Additional Best Practices