SENG 460

Practice of Information Security and Privacy

Software Development Security

Overview

- Software Development Introduction
- Security Problems in Software Development
- Secure Software Development Approaches
- Mobile Codes Security Issues
- Web Application Security Issues
- Database Security Issues

Software Development Introduction

- Software Development Life Cycle (SDLC)

The life cycle of software development deals with putting repeatable and predictable processes in place that help ensure functionality, cost, quality, and delivery schedule requirements are met.

- **Requirements gathering:** Determines *why* create this software, *what* the software will do, and *whom* the software will be created for.

- **Design:** Deals with *how* the software will accomplish the goals identified.

- **Development:** Implements software code to meet specifications laid out in the design phase.

- **Testing/Validation:** Validates the software product to ensure that goals are met and the software works as planned.

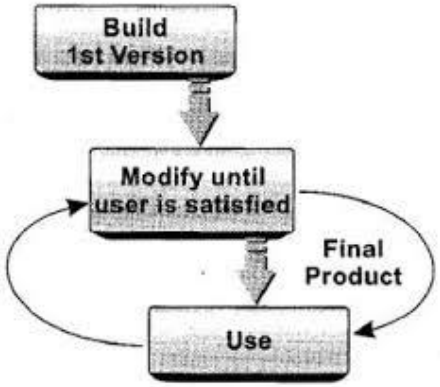
- **Release/Maintenance:** Deploys the software product and ensures that it is properly configured, patched, and monitored.

Software Development Introduction

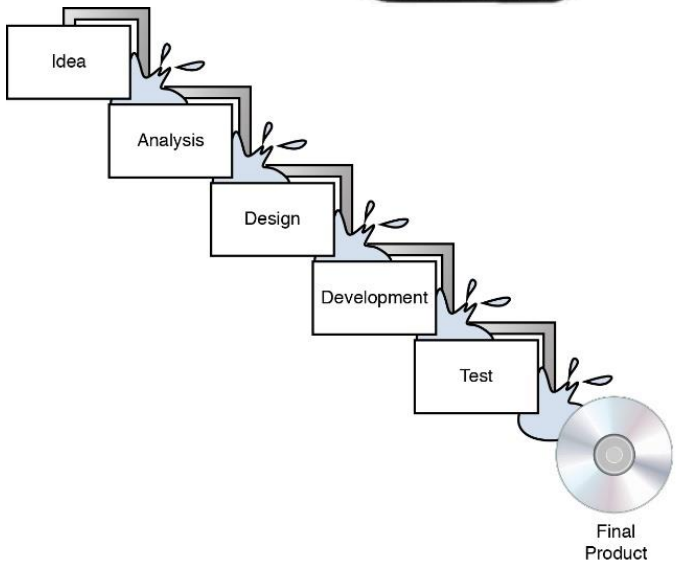
- Software Development Models

There have been several software development models developed over the last 20 or so years. Each model has its own characteristics, pros, cons, SDLC phases, and best use case scenarios.

- **Build and Fix Model:** development takes place immediately with little or no planning involved. Problems are dealt with as they occur, which is usually after the software product is released to the customer.



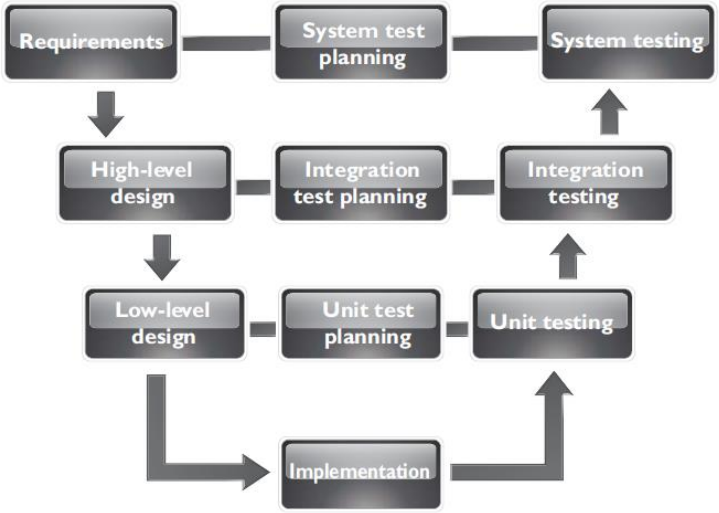
- **Waterfall Model:** uses a linear-sequential life-cycle approach. Each phase must be completed in its entirety before the next phase can begin.



Software Development Introduction

- Software Development Models

- **V-Shaped Model (V-Model):** an extension of the waterfall model. This model emphasizes the *verification and validation* of the product at each phase and provides a formal method of developing test plans at different development levels.



Verification vs. Validation

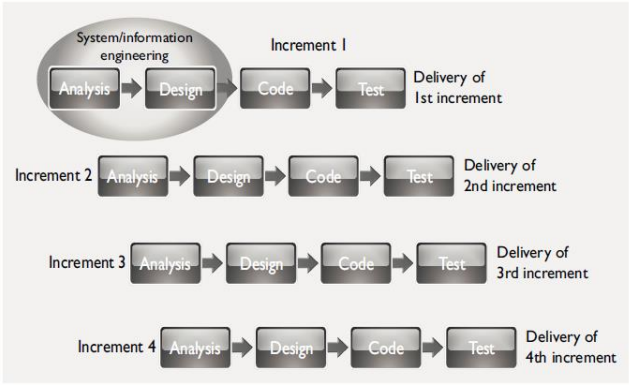
Verification *determines if the product accurately represents and meets the specifications.*
Validation *determines if the product provides the necessary solution for the intended real-world problem.*

Software Development Introduction

- Software Development Models

- **Prototyping Model:** A sample of software code or a model (prototype) can be developed to explore a specific approach to a problem before investing expensive time and resources.
  - ***Rapid prototyping****:* quickly develop a prototype to see if their ideas are feasible and if they should move forward with their current solution.
  - ***Evolutionary prototyping:*** the prototype in this model can be continually improved upon until it reaches the final product stage.
  - ***Operational prototyping:*** an extension of the evolutionary prototype method, but developed and improved upon within a production environment.

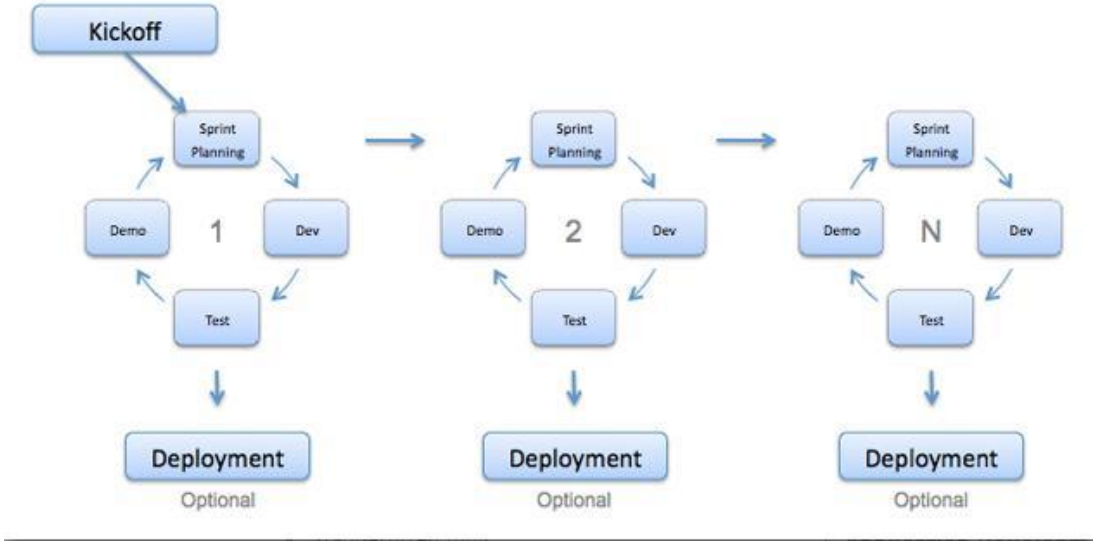
- **Incremental Model:** software development go through “multi-waterfall” cycles. Each cycle results in a deliverable that is an operational product.



Software Development Introduction

- Software Development Models

- **Agile Model:** a collection of values and principles which is based on software development best practises. Unlike traditional software development models which focuses on rigid processes, Agile model focuses on incremental and iterative development methods that promote cross-functional teamwork and continuous feedback mechanisms.



Software Development Introduction

- Software Programming Languages

- **Generation one: machine language**
  - Computer Architecture Specific
  - 1001011010, etc.
- **Generation two: assembly language**
  - Computer Architecture Specific
  - ADD, PUSH, POP, etc.
- **Generation three: high-level language**
  - Computer Architecture Independent
  - C, C++, etc.
- **Generation four: very high-level language**
  - Some are platform-Independent
  - JAVA, C#, etc.
- **Generation five: natural language**
  - The goal is to create software that can solve problems by itself instead of a programmer having to develop code to deal with individual and specific problems.
  - Some languages based on advanced knowledge-based processing and artificial intelligence

Software Development Introduction

- Software Programming Concepts

- **Assembler**: A tool that convert assembly language source code into machine code.

- **Compiler:** A tool that convert high-level language statements into the necessary machine-level format (.exe, .dll, etc.) for specific processors to understand.

- **Interpreter:** a tool converts an intermediate, platform-independent format (e.g. JAVA bytecode) into a machine-level format for execution.

- **Object-oriented Programming (OOP):** *a programming paradigm based on the concept of "objects", which are data structures that contain data, in the form of fields, often known as attributes; and code, in the form of procedures, often known as methods.*
  - ***Message:*** Used for Objects communications. Message is made up of the destination, the method that needs to be performed, and the corresponding arguments.
  - **Polymorphism:** The capability of overriding methods for derived class.
  - **Cohesion:** A measurement that indicates how many different types of tasks a module needs to carry out.
  - **Coupling:** A measurement that indicates how much interaction one module requires for carrying out its tasks.

Software Development Introduction

- Software Testing

There are different types of tests the software should go through because there are different potential flaws the team should be looking for. The following are some of the most common testing approaches:

- **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.

- **System testing** (end-to-end testing) tests a completely integrated system to verify that it meets its requirements

- **Security Testing** Focus on security perspectives of the overall system.

- **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.

Software Development Introduction

- Software Change Control

- **Software Change Control** is the process of controlling the changes that take place during the life cycle of a system and documenting the necessary change control activities.

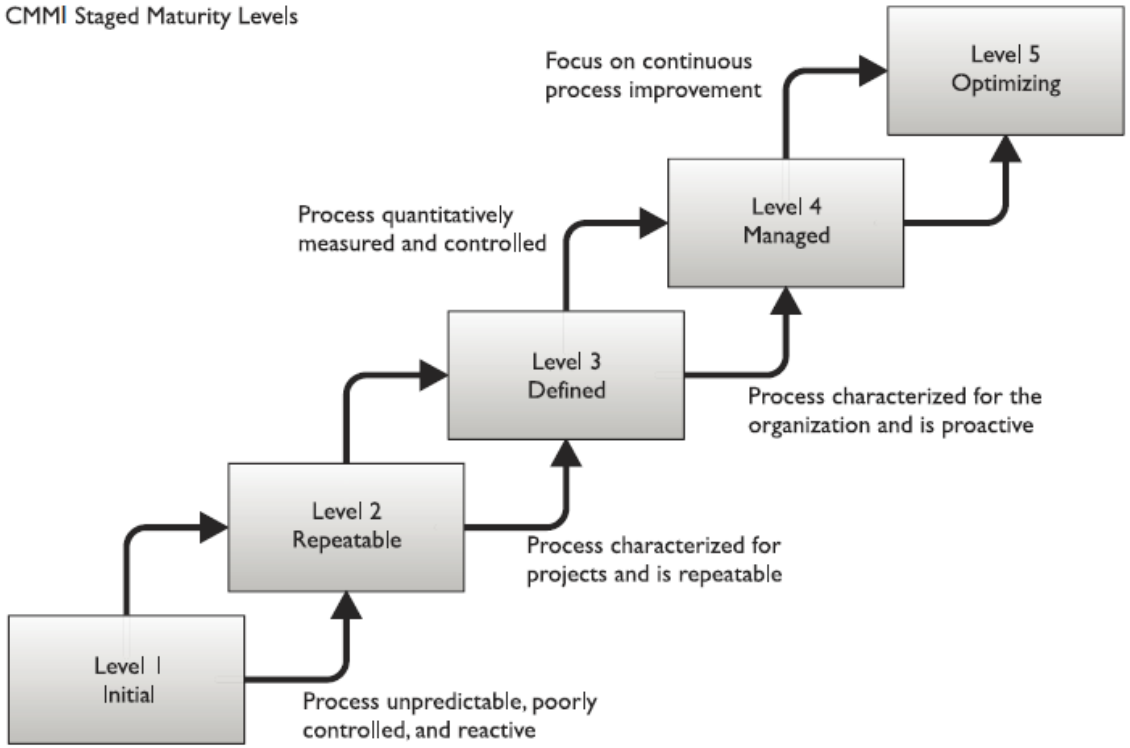
1. Make a formal request for a change.
2. Analyze the request.
3. Record the change request.
4. Submit the change request for approval.
5. Develop the change.
6. Report results to management.

Software Versioning *deals with keeping track of file revisions, which makes it possible to "roll back" to a previous version of the file.*

Software Development Introduction

- Evaluation of Software Development Process

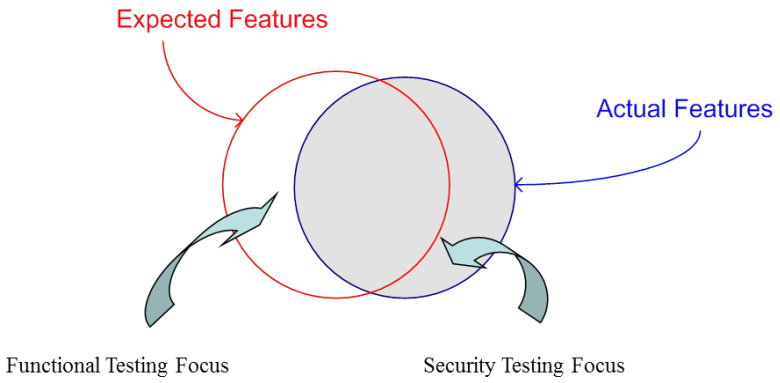
- **Capability Maturity Model Integration (CMMI):** a model which describes procedures, principles, and practices that underlie software development process maturity. software vendors would use the model to help improve their processes and customers would use the model to assess the vendors' practices.



Security Problems in Software Development

- Why security issues created in software development

- Software systems are getting more and more complicated, software developers make false assumptions when building the individual components.
- Most software developers do not have complete insight to security vulnerability issues.
- Developers often implement hidden features for self-testing purposes, which can create security issues if not cleaned up properly.
- No security related procedures are practised in many software development processes.
- Software vendors have time-to-market constraints, thus do not take time for proper security architecture, design, and testing steps.



Security Problems in Software Development

- Sample Security Vulnerability in Design

A real world example

Business Logic:

1. All application actions will be logged with identity information in audit trail.
2. An administrator can create users and assign privileges.
3. Only administrators can delete entries in the audit trail.
4. Any deletion action will be recorded in the audit trail.

Audit Log:

1. Admin A logon
2. Admin A create Admin B
3. Admin B do something bad
4. Admin B delete 3 log entries

Assumption: An attempt by an administrator to cleanse the audit logs altogether would always leave one last entry that would point the finger of suspicion at them.

Attack:

1. Logon as an administrator A.
2. Create a user B and assign B with administrative privileges.
3. Using B to perform malicious actions.
4. Using B to delete log entries generated by steps 1 ~3.

Security Problems in Software Development

- Sample Security Vulnerability in Implementation

```
public Response checkLogin(Session session) {
    try {
        String uname = session.getParameter("username");
        String passwd = session.getParameter("password");
        User user = db.getUser(uname, passwd);
        if (user == null) {
            // invalid credentials
            session.setMessage("Login failed.");
            return doLogin(session);
        }
    }
    catch (Exception e) {
        Trace.log("Exception in checkLogin: " + e.getMessage());
    }

    // valid user
    session.setMessage("Login successful.");
    return doMainMenu(session);
}
```

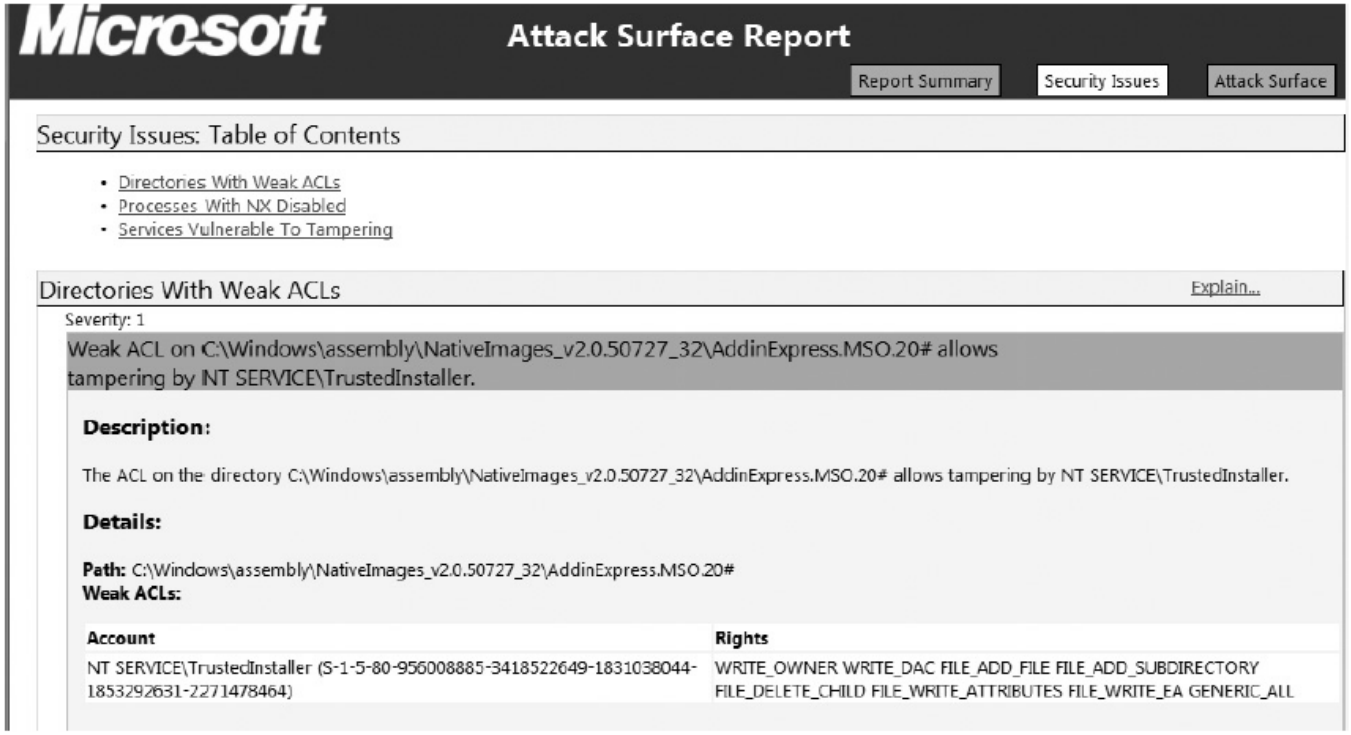
Secure Software Development Approaches

- **Requirements gathering**
  - Security requirements
  - Security risk assessment
  - Privacy risk assessment
  - Risk-level acceptance
- **Design**
  - Attack surface analysis
  - Threat modeling
- **Development**
  - Computer-aided Software Engineering (CASE) tools
  - Static Code Vulnerability Scan
- **Testing/Validation**
  - Dynamic Vulnerability Scan
  - Manual Security Testing
- **Release/Maintenance**
  - Final security review

Secure Software Development Approaches

- Attack surface analysis

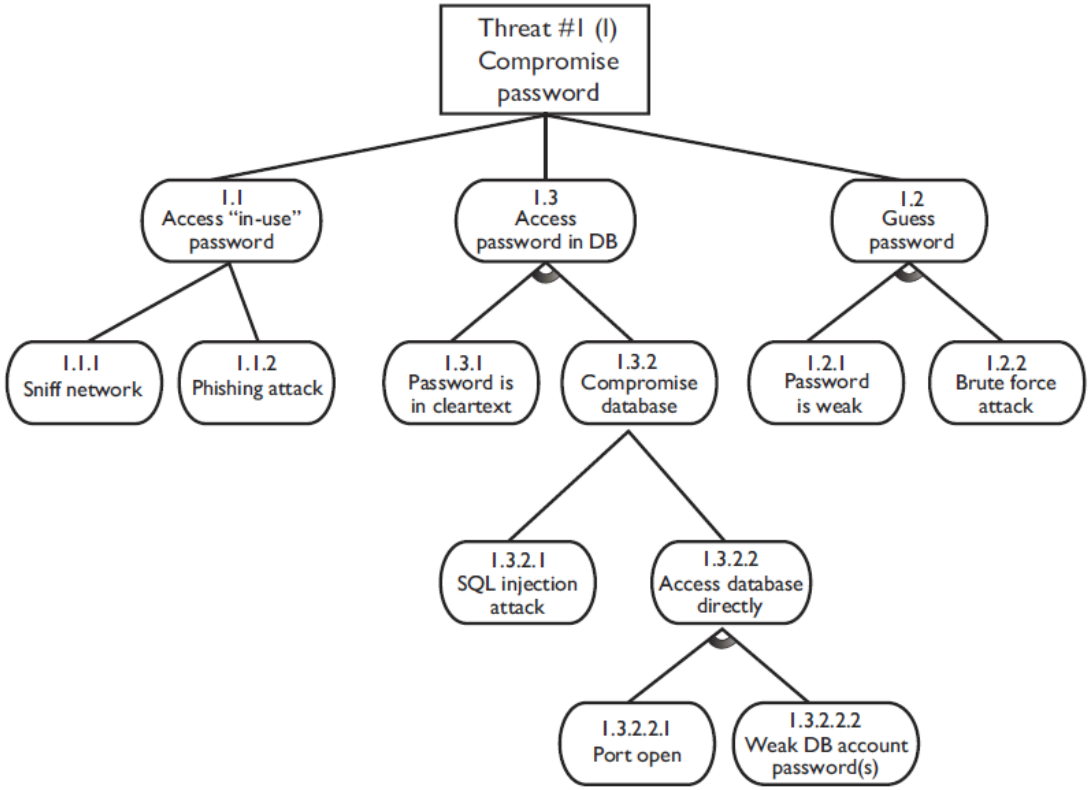
An ***attack surface*** is what is available to be used by an attacker against the product itself. The aim of an ***attack surface analysis*** is to identify and reduce the amount of code and functionality accessible to untrusted users.



Secure Software Development Approaches

- Threat Modeling

Threat modeling is a systematic approach used to understand how different threats could be realized and how a successful compromise could take place. If



Secure Software Development Approaches

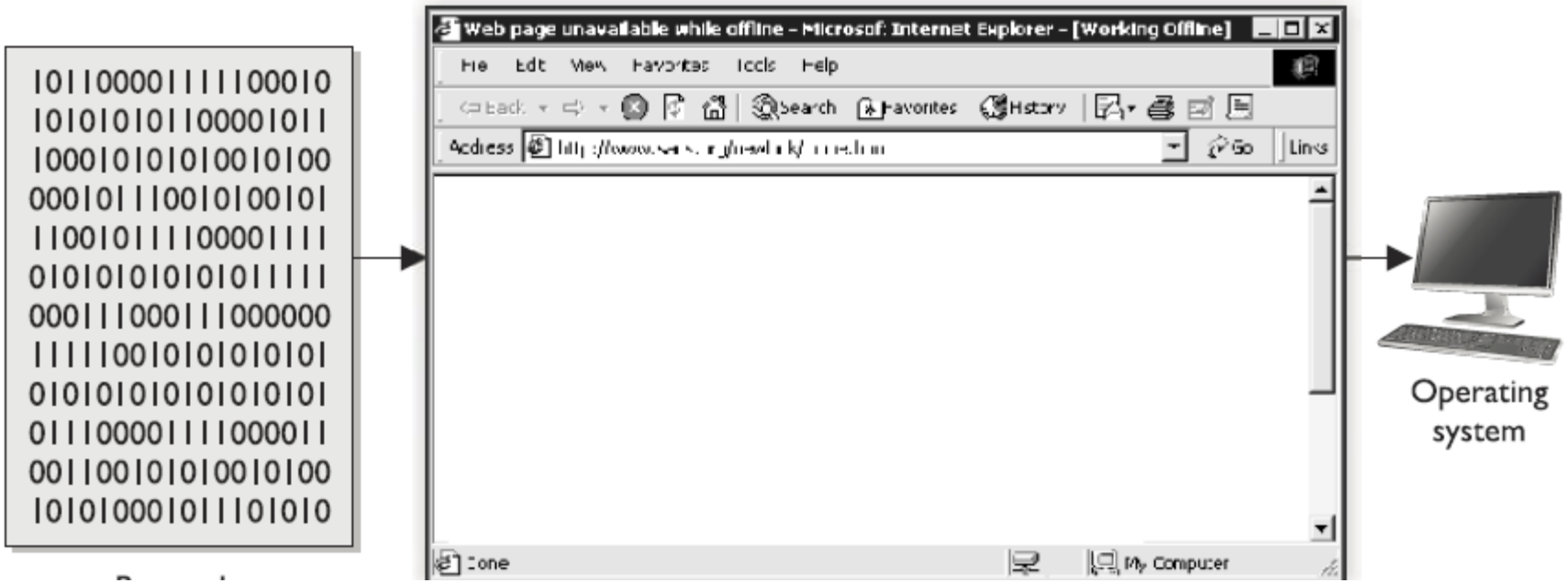
- Best Practices

- **Web Application Security Consortium (WASC)** is an organization that provides best practices for web-based applications.
- **Open Web Application Security Project (OWASP)** is another group that monitors attacks, specifically web attacks.
- **Build Security In (BSI)** initiative promotes a process-agnostic approach that makes security recommendations with regard to architectures, testing methods, code reviews, and management processes.
- **International Electrotechnical Commission (IEC)** created the 27034 standard, which is part of a larger body of standards called the ISO/IEC 27000 series. These standards provide guidance to organizations in integrating security into the development and maintenance of software applications.

Mobile Codes Security Issues

Code that can be transmitted across a network, to be executed by a system or device on the other end, is called ***mobile code***.

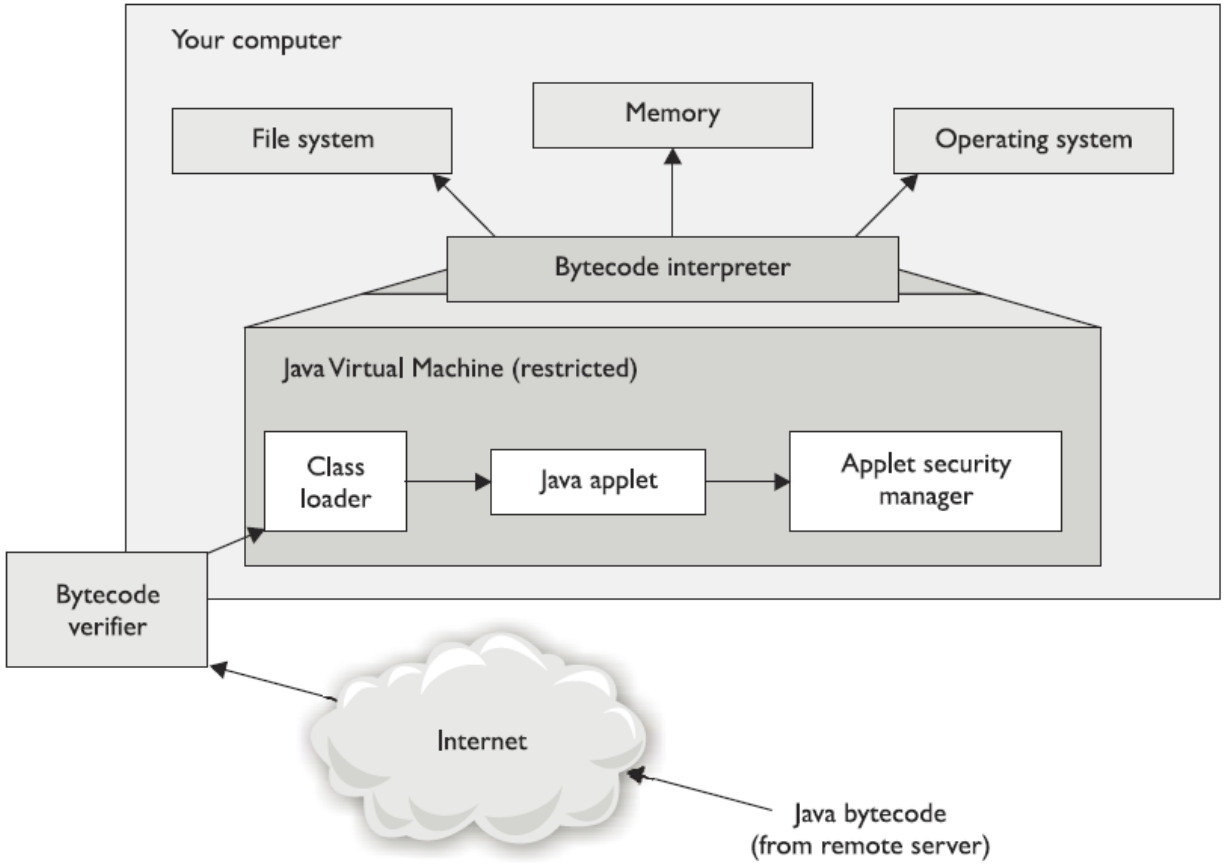
- JAVA Applet
- JAVA Script
- Adobe Flash Component
- Microsoft ActiveX Component



Mobile Codes Security Issues

- JAVA Applet Security

- Applet can be easily reverse-engineered
- Applet security manager often run by default rules



Mobile Codes Security Issues

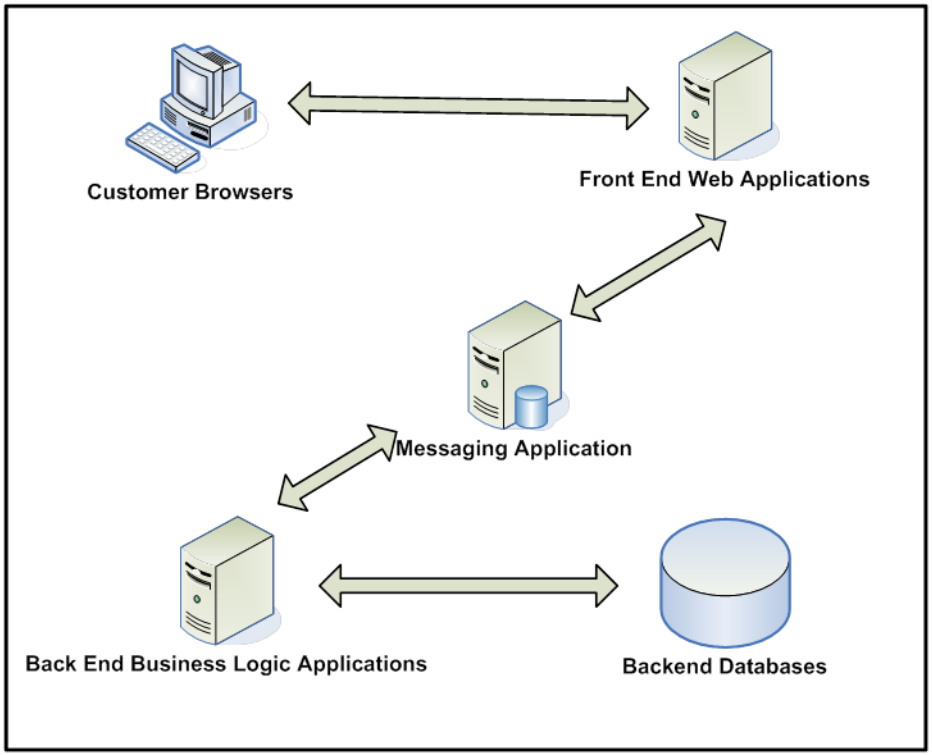
- ActiveX Security

- Uses Authenticode technology, which relies on digital certificates and trusting certificate.
- Allow web browsers to execute other software applications within the browser.
- Once allowed to run, they are able to download further ActiveX components without user authentication.
- Completely self-sufficient programs that can be executed in Windows environment.
- Shared the privilege levels of the current user on a system.

Web Application Security Issues

- A Typical Architecture of Today's Online Systems

Web-based applications are applications remotely accessible through HTTP protocol. The most common ports related to web application are 80 (HTTP) and 443 (HTTPS).



- Web Applications are facing directly external attacks .
- Web applications understand the most application logic from presentation perspective.
- Back end applications usually rely on front end web applications to implement many security controls.

Web Application Security Issues

- Same Origin Policy (SOP)

Browser same origin policy (SOP) is a key concept applying to client-side web security:

- The browser SOP identifies each web site using its origin, which is a unique combination of protocol, domain, port, and creates a context for each origin.
- The principal intent for SOP is to allow interactions between resources served in the same origin context, whilst almost completely preventing any interactions between different origin contexts.
  - Same-origin policy for cookies
  - Same-origin policy for DOM access
  - Same-origin policy for XMLHttpRequest
  - …..

Web Application Security Issues

- Web Application Attack Types (From OWASP)

- CSRF
- Cache Poisoning
- Code Injection
- Command Injection
- Comment Injection Attack
- Cross Site Tracing
- Cross-Site Request Forgery (CSRF)
- Cross-User Defacement
- Cross-site Scripting (XSS)
- Cryptanalysis
- Custom Special Character Injection
- Direct Dynamic Code Evaluation
- Direct Static Code Injection
- Double Encoding
- SQL Injection
- Server-Side Includes (SSI) Injection
- Session Prediction
- Session fixation
- Session hijacking attack
- Setting Manipulation
- Special Element Injection
- Man-in-the-browser attack
- Man-in-the-middle attack
- Mobile code: invoking untrusted mobile code
- Mobile code: non-final public field
- Mobile code: object hijack

Web Application Security Issues

- The Essential Security Problem of Web Applications

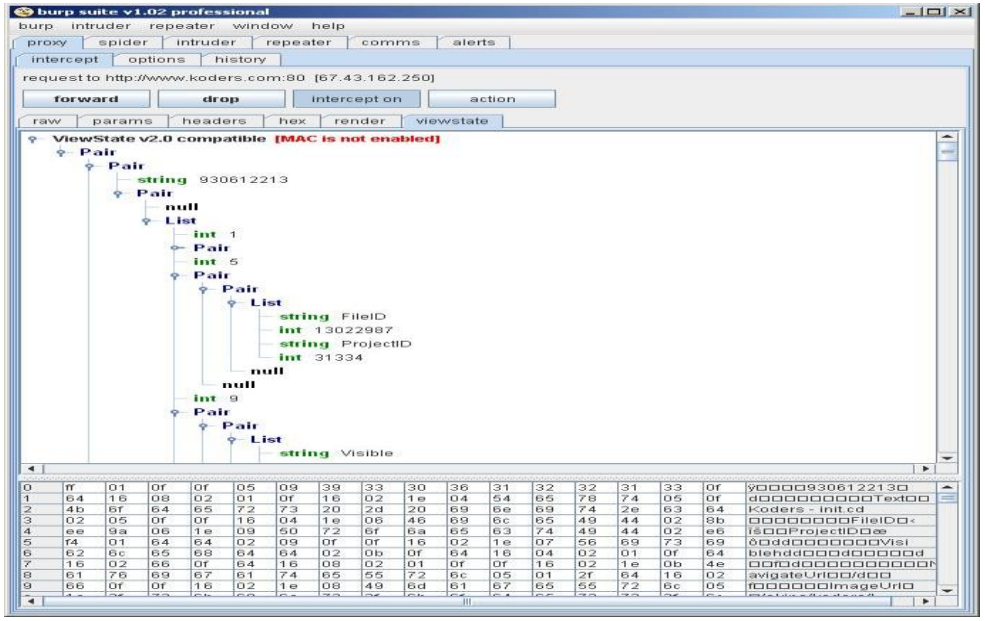
- Users have complete control over the client end:
  - Can submit arbitrary input
  - Can modify all data passing between browser and server
  - Can send requests and parameters in any sequence
  - Can use tools alongside / instead of the browser
  - …….

In principle, web application should not reply on any client-side security controls!

Web Application Security Issues

- Attacking Client-Side Controls

- Some web applications stores application state on client side:
  - Hidden form fields
    - <input name="price" type="hidden" value="1224.95">
  - ASP .NET viewState form parameter
    - <input type="hidden" name="__VIEWSTATE" id="__VIEWSTATE" value="/wEPDwUKMTIxNDIyOTM0Mg8WAh4FcHJpY2UFBzEyMjQuOTVkZA==" />



Web Application Security Issues

- Attacking Session Management

Vulnerabilities in session management mechanisms largely fall into two categories:

- Weaknesses in the generation of session tokens.
  - *'user=daf;app=admin;date=01/12/06' revealed from*

    '757365723d6461663b6170703d61646d696e3b646174653d30312f31322f3036'

- Weaknesses in the handling of session tokens.
  - Disclosure of Tokens Within the URL
    - *http://www.webjunction.org/do/Navigation;jsessionid=F27ED2A6AAE4C6DA409A3044E79B8B48?category=327*
  - Session token can be sent to malicious party through XSS vulnerability

Web Application Security Issues

- Code Injection Attack

Code injection arises because crafted input can “break out” of the data context and get interpreted as code.

SQL Injection Attack:

A typical SQL query used by a search function:

```
Select title,publisher from books where publisher = 'Wiley'
```

A malicious user can submit the search term:

```
Wiley' or 1=1--
```

resulting in the query:

```
Select title,publisher from books where publisher = 'Wiley' or 1=1--'
```

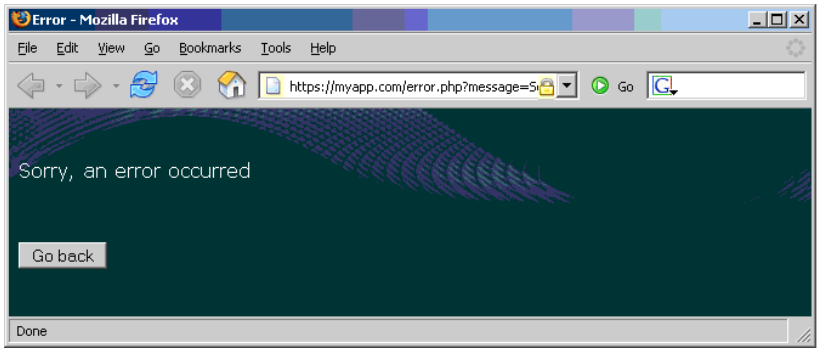
The query returns all title and publisher items in the books table, because 1=1 is always true.

Web Application Security Issues

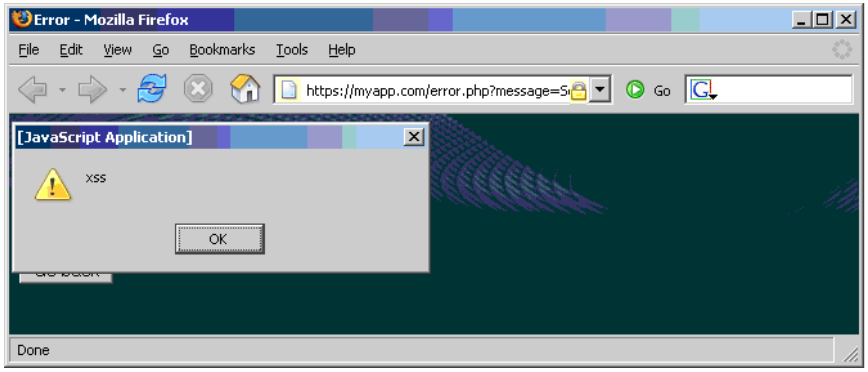
- Attacking the Same Origin Policy Control

Reflected Cross Site Scripting Attack:

https://myapp.com/error.php?message=Sorry%2c+an+error+occurred



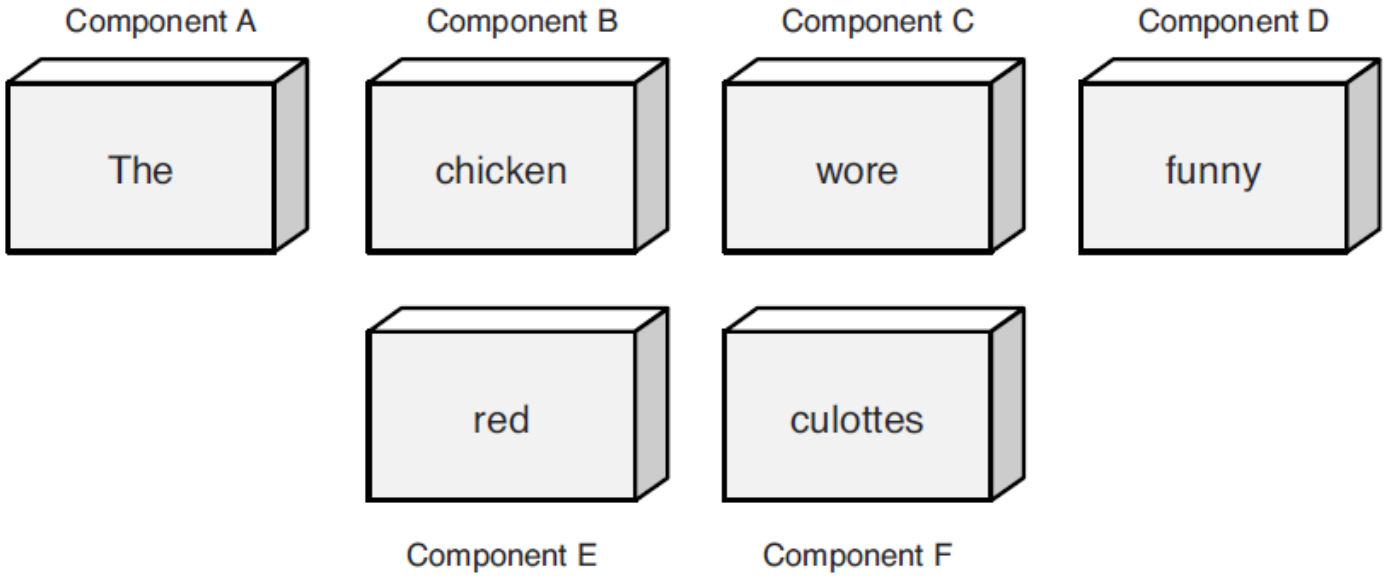
https://myapp.com/error.php?message=<script>alert('xss');</script>



Database Security Issues

- Aggregation Threat

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



Database Security Issues

- Inference Threat

Inference is the ability to derive information not explicitly available. Inference is the intended result of aggregation. The inference problem happens when a subject deduces the full story from the pieces he learned of through aggregation. This is seen when data at a lower security level indirectly portrays data at a higher level.

AS an example, if a clerk were restricted from knowing the planned movements of troops based in a specific country, but did have access to food shipment requirements forms and tent allocation documents, he could figure out that the troops were moving to a specific place because that is where the food and tents are being shipped. The food shipment and tent allocation documents were classified as confidential, and the troop movement was classified as top secret. Because of the varying classifications, the clerk could access and ascertain top-secret information he was not supposed to know.

Database Security Issues

- **Database Security Controls**

  - **Content-dependent access control rules:** access is determined by the sensitivity of the data.
  - **Context-dependent access control rules:** access is determined by multiple factors such as location, time of day, and previous access history.
  - **Database Views:** Refers to the given set of data a user or group can see when they access the database.
  - **Database Locks:** Used when one user is accessing a record that prevents another user from accessing the record at the same time to prevent edits until the first user finishes.
  - **Polyinstantiation:** A process used to prevent data inference violations by enabling a relation to contain multiple tuples with the same primary keys with each instance distinguished by a security level. It prevents low-level database users from inferring the existence of higher level data.