SQLIA

- Consider input values are
  - Username = a' or 'a' = 'a
  - Password= a' or 'a' = 'a
- Query will become

  Select * from user where username= 'a' or 'a'='a' and password='a' or 'a'='a';

  Here or 'a' = 'a' is always true this will allow you to access that particular account.

Elements And Expression of Signature

TABLE I. . ELEMENTS AND EXPRESSIONS OF SIGNATURE

Element	Symbol
Alphanumeric	\w
Comment Mark	(\-\-)
Quotation Mark	['"]
Comparison Mark	([=<>][<>!]=)
Logical Keyword	(OR)
Type	(int|char|varchar)
Type Conversion Keyword	(CONVERT|CAST)
SQL Keyword	(SELECT|INSERT|UPDATE|DROP)
UNION Keyword	(UNION | UNION ALL)
Delimiter Mark	;
Delay Tim	\d+
White Space	\s
Bracket	\c
Case Insensitive	/i

Q1. What is the impact of malicious command injection by attacker on web site ? Illustrate with SQL and XSS Injections?

Q2 . Give defense mechanism for sql and xss Injections .

Elements And Expression of Signature

- Usually, the attacker injects the malicious command through either any input forms on the web application or via the URL header.
- Therefore, the transiting incidents along the edge can be taken from the set {Web Page Form Access, URL Header Access};
- and upon successful execution of either incident, the adversary progresses to the next state: Found Injection Place.

  - Incident € {Web Page Form Access, URL Header Access}
  - $SIG_{2,1}$ € {SIG_{Web} , SIG_{URL} }

Tautology Query-Conditional statement

- The Tautology Query attack injects a piece of malicious code into one or more conditional statements so that they always evaluate to true and generate the result according to the evaluated true condition.

- In order to model and identify this attack is Tautology attack; the attacker injected code should meet the defined signature, which is shown as below,
  - Incident € (Tautology Query Statement)
  - SIG $_{T3, T2}$ € (/['"]\s+(OR)\s+\w+\s*([=<>]|[<>!]=)\s*\w+ \-\-/i)
  - When the state of Web Server Execute Tautology Query happened, the only way to transit into SQLIA state is through the edge Tautology Attack.

Tautology Query

- There must be some incidents to indicate the achieving of SQLIA. The set of is {Bypass Authentication, Information Retrieval}.
- Once either of the incidents happen, the attack state achieve.
  - Incident € {Bypass Authentication, Information Retrieval}
  - $SIG_{R,T3}$ € {SIG_{BA} , SIG_{IR} }

Logically Incorrect Query

- Logically Incorrect Query attack is the attacker intent to obtain the error feedback message by injecting incorrect command into the database.
- The database structure and type information can be extracted according to the error message.
- The generated signature is stated as follows,
  - Incident € {Logically Incorrect Query Statement}
  - $SIG_{U3,U2}$ € {/(CONVERT|CAST)\s*\c+\s*(int|char| varchar)/i}
- For the last edge which leading to SQLIA state, it's incident contains Return Error Message from Database and Information Retrieval.
  - Incident € {Return Error Message from Database, Information Retrieval}
  - $SIG_{R,U3}$ € {SIG_{EM} , SIG_{IR} }

UNION Query

- The UNION Query attack is to inject UNION keyword following with another SELECT query statement.
- The result is database returns the dataset that is the union results of the original first query and the injected second query.
- The label of second edge is *UNION Query Injection and the label of top edge is UNION Query Attack.*
  - $SIG_{U3,U2}$ €{/['']\s+(UNION|UNION\s+ALL)\s+(SELECT) /i}
- The label in top edge is UNION Query Attack. The possible incidents are Information Retrieval and Bypass Authentication.
  - Incident €{Information Retrieval, Bypass Authentication}
  - $SIG_{R,U3}$ €{SIG_{IR} , SIG_{BA} }

Piggy-Backed Query

- In Piggy-Backed Query attack, the query be extended by injecting additional queries after the original one.

- Consequently, the database receives multiple SQL queries and executes them in sequence.
- It's wise to model the signature with those keywords following with a delimiter which indicates the ending of previous query.
  - Incident € {Piggy-Backed Query Statement}
  - $SIG_{PB3, PB2}$ € {/['"]\s*;\s*(SELECT|INSERT|UPDATE|DELETE |DROP)/i}

Piggy-Backed Query

- The incidents in the top edge, which label is Piggy-Backed Query Attack, can be Information Retrieval, Information Modification and Perform DoS.
  - Incident € {Information Retrieval, Information Modification and Perform DoS}
  - $SIG_{R, PB3} \in \{SIG_{IR}, SIG_{IM}, SIG_{DoS}\}$

Timing Inference Query

- The inference attack implemented according to the obtained result from a true or false evaluation about data.
- The generated signature is stated as follows,
  - Incident €{Timing Inference Query Statement}
  - $SIG_{TI3,\ TI2}$ € {/(WAITFOR)\s+\d+/I}
- Moreover, the label of top edge is Timing Inference Attack. The possible incidents are Information Retrieval, Information Modification and Identify Database Scheme.
  - Incident € {Information Retrieval, Information Modification and Identify Database Scheme}
  - $SIG_{R,\ TI3}$ € {SIG_{IR} , SIG_{IM} , SIG_{DS} }

Art of Attack for SQLIA -- 1

Input from form

Login - ' '

Password – abc ' or '1' = '1'

Resulted Query in program

Select * from User where login = ' ' and password = 'abc ' or '1' = '1'

Injected query returns successful login because in 'or' condition check for 1 = 1 which is all ways true so irrespective of login name and password it will allow user to login into.

Prevention

Filter out 'or 1 = 1' word in login and password field by using regular expression . After submitting form capture each and every field of form check it contain substring or '1' = '1'

str = "or 1 = 1";

isSubString(password, str); .... ture → if or 1 = 1 is present in password

false → othersiwe

Art of Attack for SQLIA -- 2

Input from form

Login = ' ' union select cardNo from
CreditCards where acctNo= 7032 –
Password = asd

Resulted Query in program

Select * from User where Login = ' ' union select cardNo from
CreditCards where acctNo= 7032 –
Password = asd

The original query should return the null set, and the injected query returns data from the "CreditCards" table.

Prevention

Restrict the length of data type in the input. Here in example login name is too long generally login name is 15 to 20 characters validate form on length of each and every field

e.g == if (login.length>15) { alert("The password have no more than 10 characters");
submitOK="false"; }

Art of Attack for SQLIA -- 3

Input from form

Login= ' '

Password = ' '; drop User;

Resulted Query in program

Select from User where Login= ' ' and password = ' ' ; drop User;

The database treats this query string as two queries separated by the query delimiter (";") and executes both. First query will produce null result and second will drop table user.

Prevention

Filter out semicolon (;) from input by validating input .

Art of Attack for SQLIA -- 4

Input from form

Login = ' '

Password= convert(int, select top 1 name from sysobjects where xtype='u'))

Resulted Query in program

Select * from users where login = ' ' and password = convert(int, select top 1 name from sysobjects where xtype='u'));

The injected query extracts the name of the first user table type 'u' from the database's metadata table sysobjects. It then converts this table name to an integer. Because the name of the table is a string, the conversion is illegal and the database returns an error. For example, a SQL Server may return the following error: "Microsoft OLE DB Provider for SQL Server (0x80040E07) Error converting varchar value 'CreditCards' to a column of data type int." From this message, the attacker can 1) see that the database is an SQL Server and 2) discover that the name of the first userdefined table in the database is "CreditCards" (the string that caused the type conversion to occur).

Prevention

Restrict functions as part of input. In form validation write proper rule for restriction of function in input. Access control mechanism should be implemented properly[8].

Art of Attack for SQLIA -- 5

Input from form

Login='legalUser' and ASCII(substring ((select top 1 name sysobjects), 1,1)) > X WAITFOR 10 - - '
Password = ' ';

Resulted Query in program

Select acct from users where login='legalUser' and ASCII(substring ((select top 1 name sysobjects), 1,1)) > X WAITFOR 10 - - ' and password = ;

In the attack, the SUBSTRING function is used to extract the first character of the database's first table's name, which is then converted into an ASCII value and compared with the value of X. If the value is greater, the attacker will be able to observe a 10 s delay in the database response. The attacker can continue this way and use a binary-search strategy to identify the value of each character in the table's name.

Prevention

Filter special characters from input by validating form .

Current Information Security Approaches

- 1. Pro-active Protection: current approaches mainly adopt the pro-active protection hardware or software, e.g., firewalls, IDS, etc.
- 2. Data Protection server: for data protection, most of the enterprises adopt data protection servers and multi-level secure mechanisms, e.g., encryption data storage, password management, etc.
- 3. The use of "Outside-In" Protection: many approaches adopt gateways, demilitarized zone (DMZ), and proxies to limit the accessibilities.

- **Development of secure IS**

Q1: How should an IS be developed in order to be secure?(Programming language vulnerabilities like java or dbms weakness, ESAPI framework or java network programimg)

- **Access to IS**
- Q2 How can people's access to information be controlled?(Accsess control by DBMS, OS, Implementation strategy-authentication and authorization)
- **Secure communication:**

Q3 How can secure communication between people be ensured?(TCP/IP OR CRYPTOGRAPHY)

- **Security management:** How should information security be managed? (DID Mechanism)

Model of Software Life Cycle Processes for Secure Information Systems

Model consist of

- Comprehensive Set of Tasks
- List of ISO standards underlying the Tasks
- The Regular Sequence of the Tasks

Comprehensive Set of Tasks

We clarified the following criteria for engineering tasks related to security.

- *Developing the security facilities or creating documents* related to security
- *Managing the documents*
- *Verifying, validating, and reviewing the documents*
- *Modifying the documents*
- *Referring the documents*
- *Analyzing and managing surrounding environment* of systems

23 tasks from ISO/IEC 12207 tasks according to these criteria are extracted

Tasks of the Model

Tasks from ISO/IEC 12207

- Infrastructure Management
- Human Resource Management
- Quality Management
- Risk Management
- Information Management
- Software Requirement Analysis
- Software Architectural Design
- Software Detailed Design

Tasks from ISO/IEC 12207

- Software Qualification Testing
- Software Installation
- Software Integration
- Software Construction
- Software Configuration Management
- Software Operation
- Software Maintenance
- Software Disposal

ISO standards underlying the Tasks

Tasks	ISO Standards
Infrastructure Management	13335, 27001, 27002, 27006
Human Resource Management	13335, 27001, 27002, 27006
Quality Management	13335, 27001, 27002, 27006
Risk Management	15408
Information Management	11770
Software Requirements Analysis	15408, 19790
Software Architectural Design	15408, 15446, 15816, 15945, 15947, 18028
Software Detailed Design	7064, 9796, 9798, 10118, 13888, 14888, 18014
Software Implementation	9797, 10116, 10118, 15946, 18031, 18032, 18033

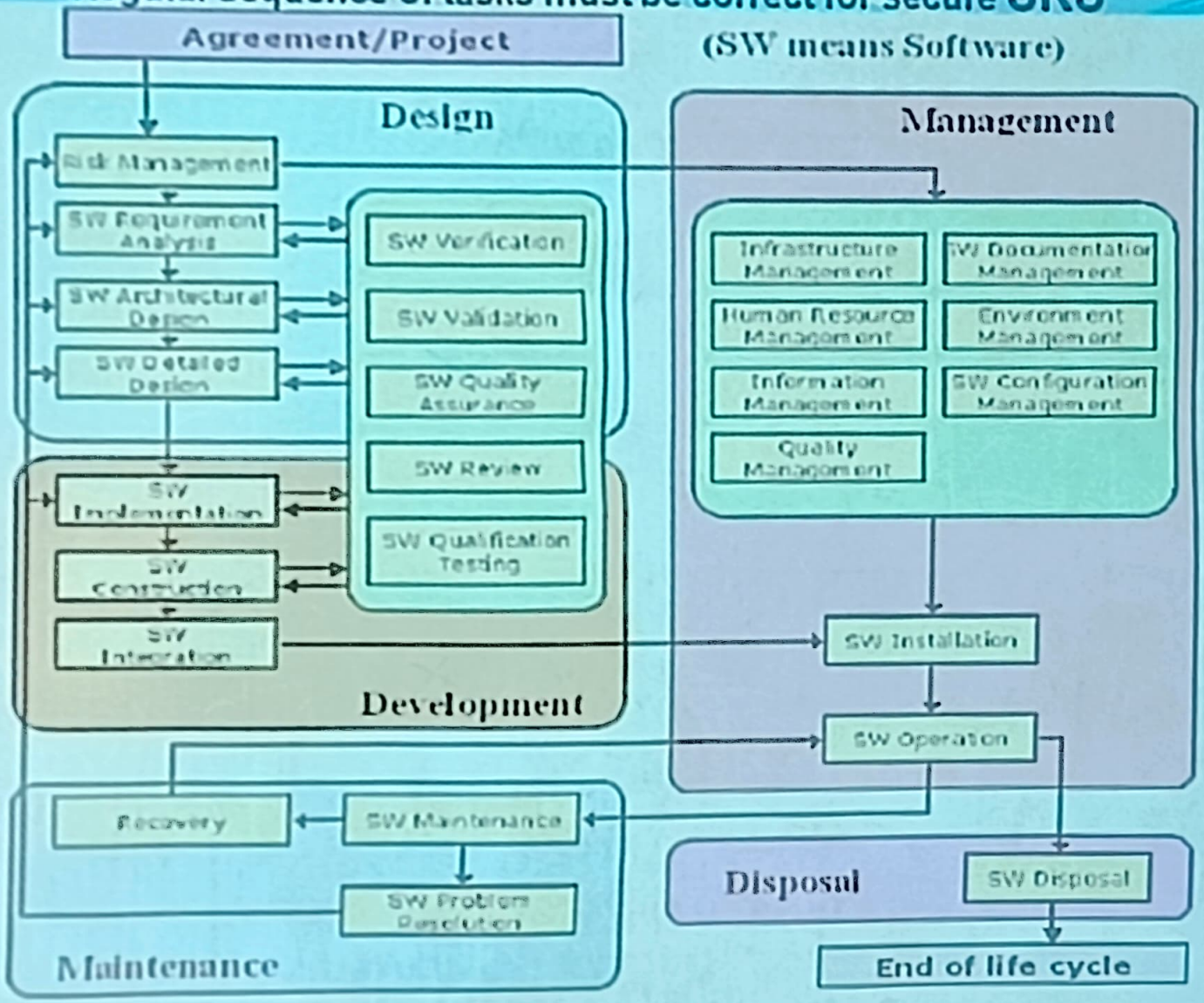
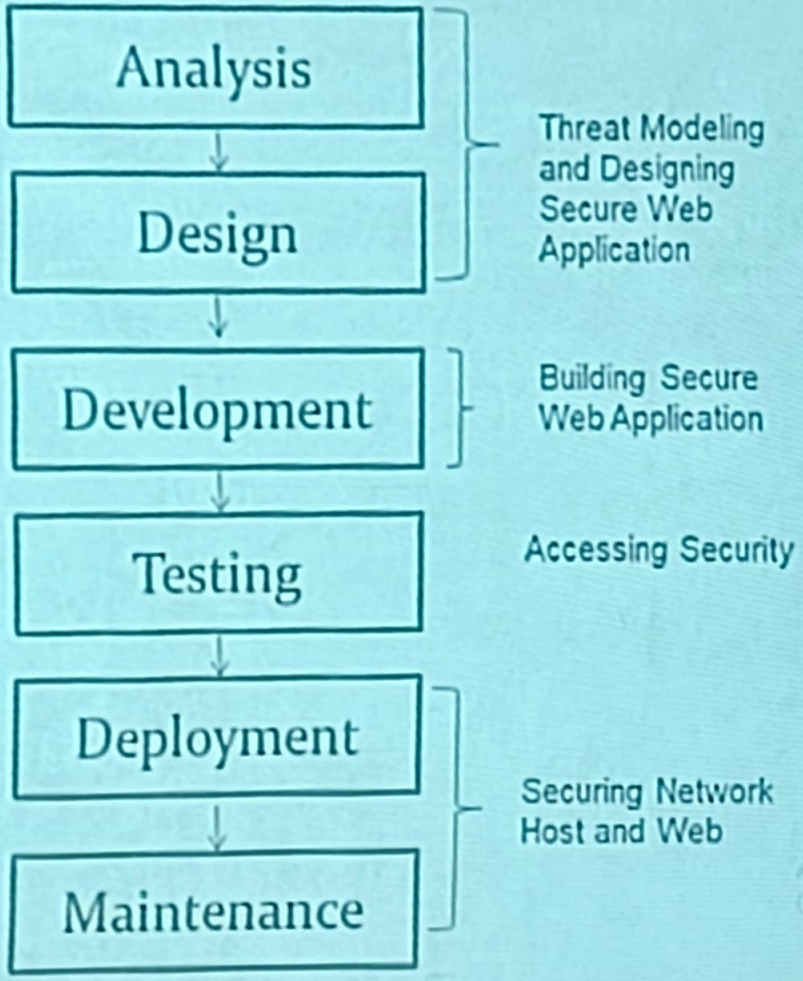
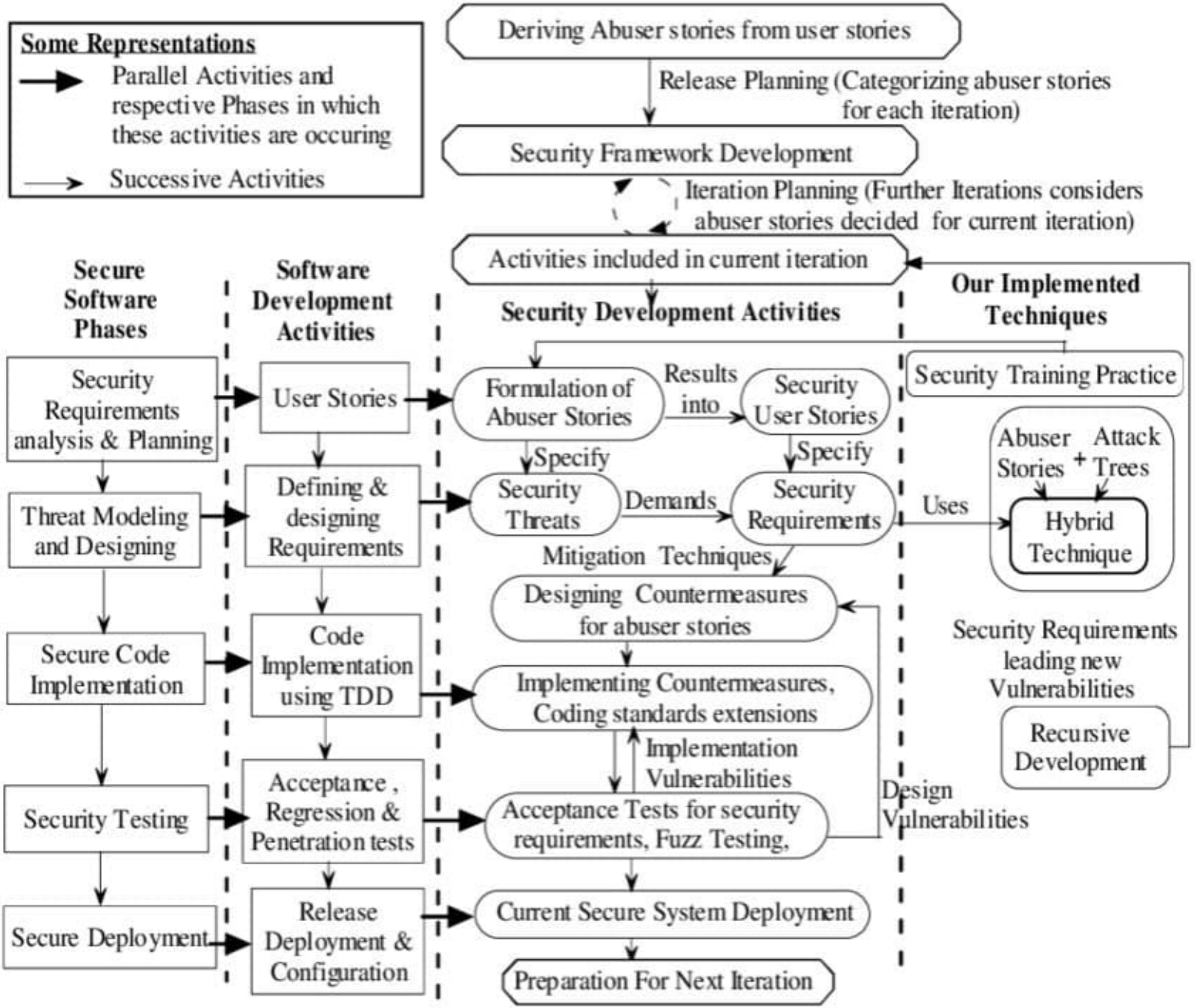


Figure 2. The regular sequence of the tasks

Development of Security Framework for Requirement Elicitation

Applying security in product life cycle




Some Representations
Parallel Activities and respective Phases in which these activities are occuring
Successive Activities
Deriving Abuser stories from user stories
Release Planning (Categorizing abuser stories for each iteration)
Security Framework Development
Iteration Planning (Further Iterations considers abuser stories decided for current iteration)
Activities included in current iteration
Secure Software Phases
Software Development Activities
Security Development Activities
Our Implemented Techniques
Security Requirements analysis & Planning
User Stories
Formulation of Abuser Stories
Results into
Security User Stories
Security Training Practice
Specify
Specify
Abuser Stories + Attack Trees
Threat Modeling and Designing
Defining & designing Requirements
Security Threats
Demands
Security Requirements
Uses
Hybrid Technique
Mitigation Techniques
Designing Countermeasures for abuser stories
Secure Code Implementation
Code Implementation using TDD
Implementing Countermeasures, Coding standards extensions
Security Requirements leading new Vulnerabilities
Recursive Development
Implementation Vulnerabilities
Security Testing
Acceptance , Regression & Penetration tests
Acceptance Tests for security requirements, Fuzz Testing,
Design Vulnerabilities
Secure Deployment
Release Deployment & Configuration
Current Secure System Deployment
Preparation For Next Iteration

Security Framework

Phase I:Security Requirements analysis & Planning

1.Identify critical assests, 2. Formulation of abuser stories in every phases of life

1. Identify critical assests: Given phase explores user stories describing features and functional requirements of the software to be built.

- Keeping security in mind, developer will identify critical assests of the system for describing threats in future and in addition addresses specific security objectives like goal, constraints with user stories.

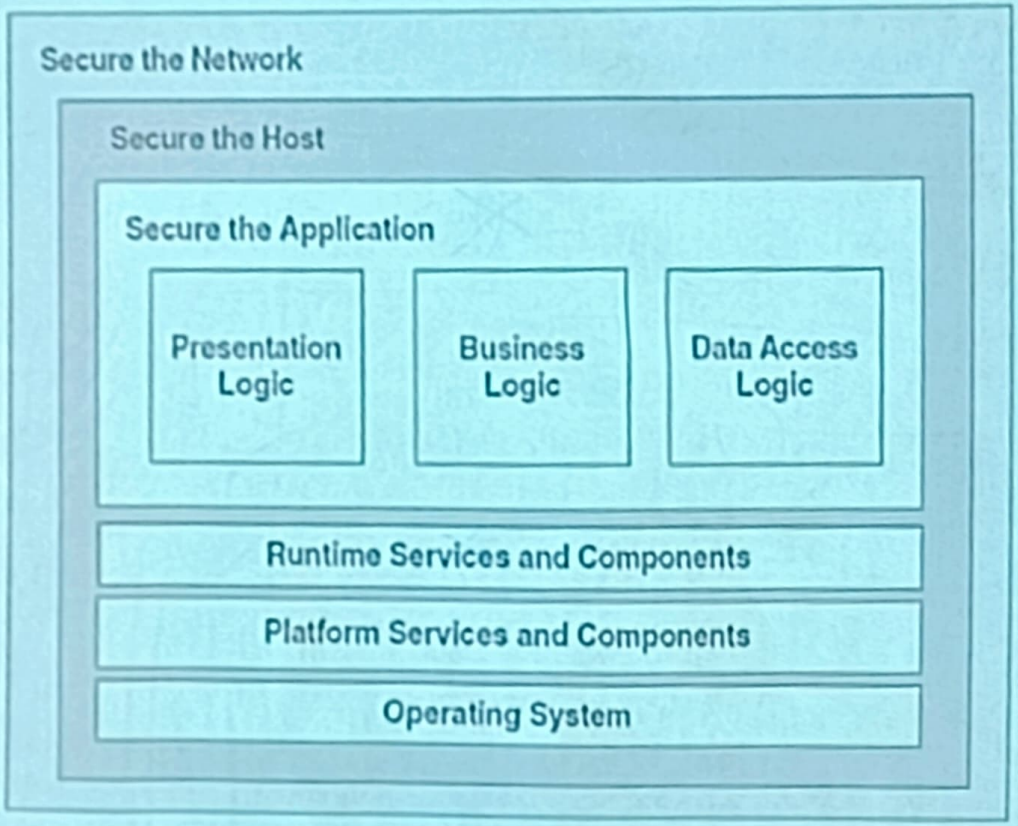
Threats in Software Development Life Cycle

Phases	Threats	
Analysis	• Business goals e.g. allowing 24hrs banking via web can yield DOS • System Boundary assessment: Every legitimate system entry or exit point is a threat, as well as other possibly illegitimate access points to a system. • analysis on abuse of privileges by insiders can yield a lot of threat and vulnerability information	
Design	Input Validation	BOF, XSS,SQLIA, Canonicalization
	Authentication	Network eavesdropping, brute force attacks, dictionary attacks, cookies reply, credential theft
	Authorization	Elevation of privilege , disclosure of confidential data, data tampering , luring attacks
	Configuration mgmt	Unauthorized access to administration interfaces ; unauthorized access to configuration store, retrieval of clear text configuration data; lack of individual accountability, over privileged process and service accounts

Development	Input Validation	BOF, XSS,SQLIA, Canonicalization
	Authentication	Network eavesdropping, brute force attacks, dictionary attacks, cookies reply, credential theft
	Authorization	Elevation of privilege, disclosure of confidential data, data tampering, luring attacks
	Configuration mgmt	Unauthorized access to administration interfaces; unauthorized access to configuration store, retrieval of clear text configuration data; lack of individual accountability, over privileged process and service accounts
	Sensitive mgmt	Access sensitive data in storage, network eavesdropping , data tampering
	Session mgmt	Session hijacking, session replay, man in the middle
	Cryptography	Poor key generation or key mgmt, weak or custom encryption
	Parameter manipulation	Query string manipulation, form field manipulation, cookie manipulation, HTTP header manipulation
	Exception mgmt	Information disclosure, denial of service

	Sensitive mgmt	Access sensitive data in storage , network eavesdropping , data tampering
	Session mgmt	Session hijacking, session replay, man in the middle
	Cryptography	Poor key generation or key mgmt, weak or custom encryption
	Parameter manipulation	Query string manipulation , form field manipulation, cookie manipulation , HTTP header manipulation
	Exception mgmt	Information disclosure, denial of service
	Auditing & logging	User denies performing an operation; attacker exploits an application without trace, attacker covers his or her tracks

The Holistic Approach



- Web application security must be addressed across application tiers and at multiple layers.

- An attacker can exploit weaknesses at any layer. For this reason, the we discuss here a holistic approach to application security and applies it at all three layers.