Information Security
CS3002
(Sections BDS-7A/B)
Lecture 17

Instructor: Dr. Syed Mohammad Irteza

Assistant Professor, Department of Computer Science

16 October, 2024

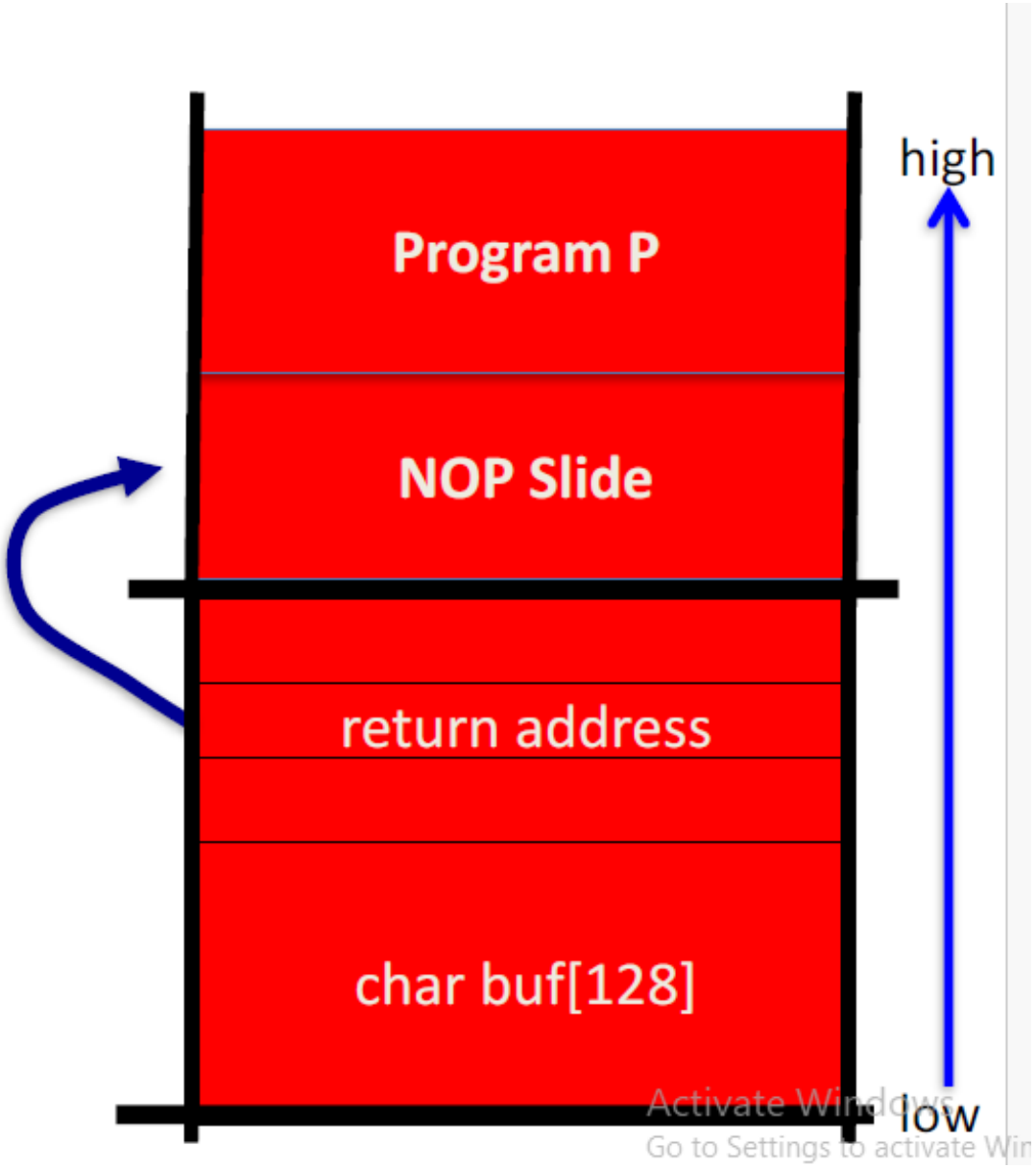
The NOP slide (NOP sled)

Problem: how does attacker determine the ret-address?

Solution: NOP slide

- Guess approximate stack state when `func()` is called
- Insert many NOPs before program P:

```
nop , xor eax,eax , inc ax
```



NOP slide (NOP sled)

How does a NOP sled work? - Stack Overflow

- Some attacks consist of making the program jump to a specific address and continue running from there.
  - *The injected code has to be loaded previously somehow in that exact location*
- Stack randomization and other runtime differences may make the address where the program will jump impossible to predict
  - *So the attacker places a NOP sled in a big range of memory*
- If the program jumps to anywhere into the sled, it will run all the remaining NOPs, doing nothing, and then will run the payload code, just next to the sled.
- ***The reason the attacker uses the NOP sled is to make the target address bigger***: the code can jump anywhere in the sled, instead of exactly at the beginning of the injected code

Preventing Integer Overflow Attacks

- Prefer using *unsigned integer types* whenever possible.
- Review and test your code by *writing out all casts explicitly* to identify where implicit casts might cause integer overflows.
- Turn on *any options available in your compilers* that can help identify certain types of integer overflows.
- Adopt *secure coding practices such as bounds checking*, input validation, and using safer functions.
- Perform a *bounds check on every value that is user-modifiable* before using it in an arithmetic operation.

Format string attack

To understand the attack, it’s necessary to understand the components that constitute it.

- The **Format Function** is an ANSI C conversion function, like `printf`, `fprintf`, which converts a primitive variable of the programming language into a human-readable string representation.
- The **Format String** is the argument of the Format Function and is an ASCII Z string which contains text and format parameters, like:

```
printf (“The magic number is: %d\n”, 1911);
```

- The **Format String Parameter**, like **%x %s** defines the type of conversion of the format function.

Preventing Format String Vulnerabilities

- Always specify a *format string as part of program, not as an input*. Most format string vulnerabilities are solved by specifying **"%s"** as format string and not using the data string as format string
- If possible, make the ***format string a constant***. Extract all the variable parts as other arguments to the call. Difficult to do with some internationalization libraries
- If the above two practices are not possible, use **defenses such as Format_Guard** . Rare at design time. Perhaps a way to keep using a legacy application and keep costs down. Increase trust that a third-party application will be safe

Database Security

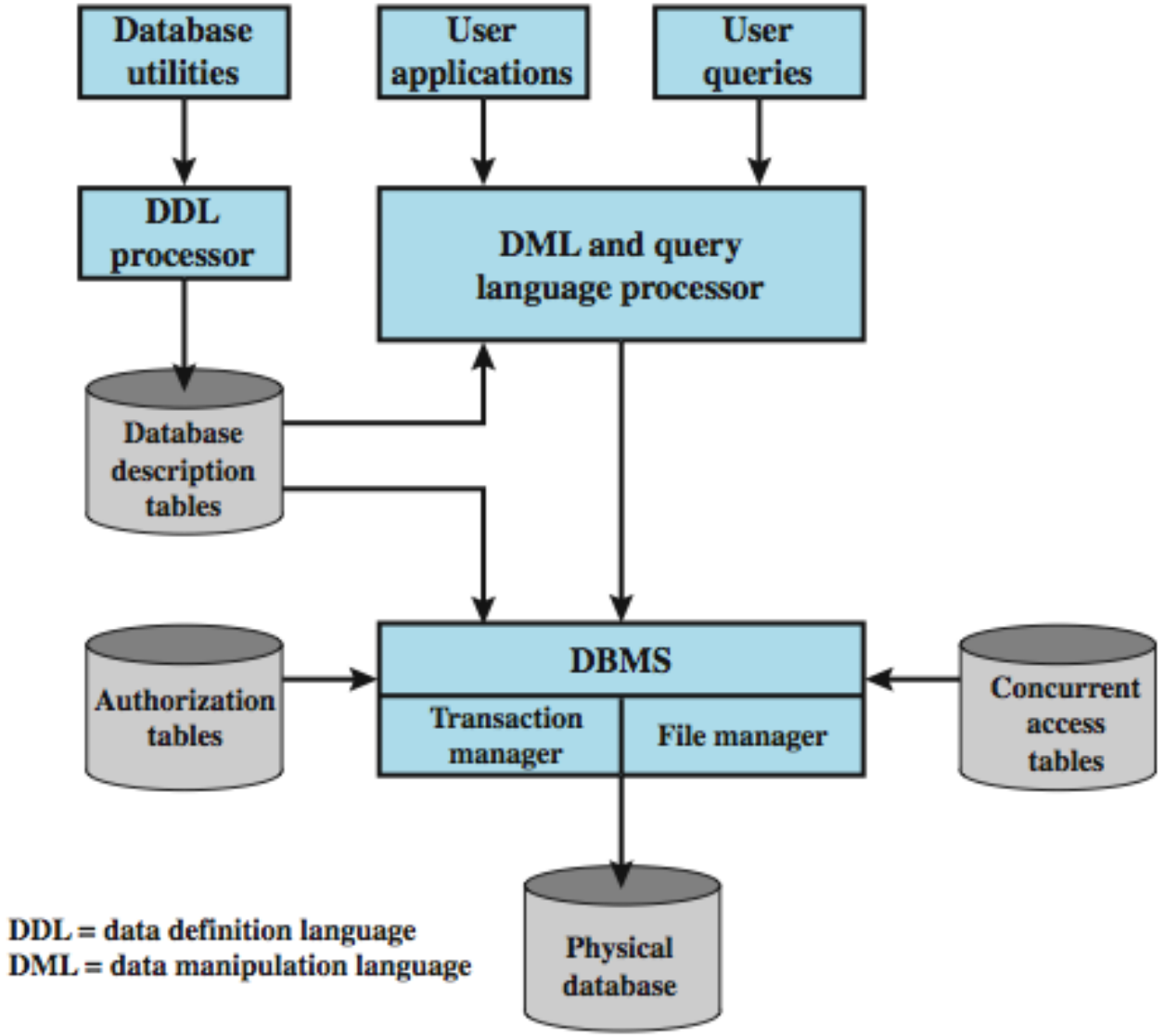
- Basics
- SQL Injection Attack, techniques, types of attack
- Countermeasures, database access control

Database systems

- *Structured collection of data* stored for use by one or more applications
- Contains the *relationships between data items* and *groups of data items*
- Can *sometimes contain sensitive data* that needs to be secured
- *Query language*: Provides a uniform interface to the database

Database Architecture

- Database: structured collection of data stored for use by one or more applications.
- DB also contains the relationships between data items and groups of data items.
- Accompanying the DB is a database management system (DBMS):
  - which is a suite of programs for constructing and maintaining the database etc.



Database Architecture (cont'd)

- Developers make use of a *data definition language (DDL)* to define the database logical structure and procedural properties, which are represented by a set of database description tables.
- A *data manipulation language (DML)* provides a powerful set of tools for application developers.
- *Query languages* are declarative languages designed to support end users.
- The *database management system* makes use of the database description tables to manage the physical database.
- The interface to the database is through a *file manager module* and a *transaction manager module*.

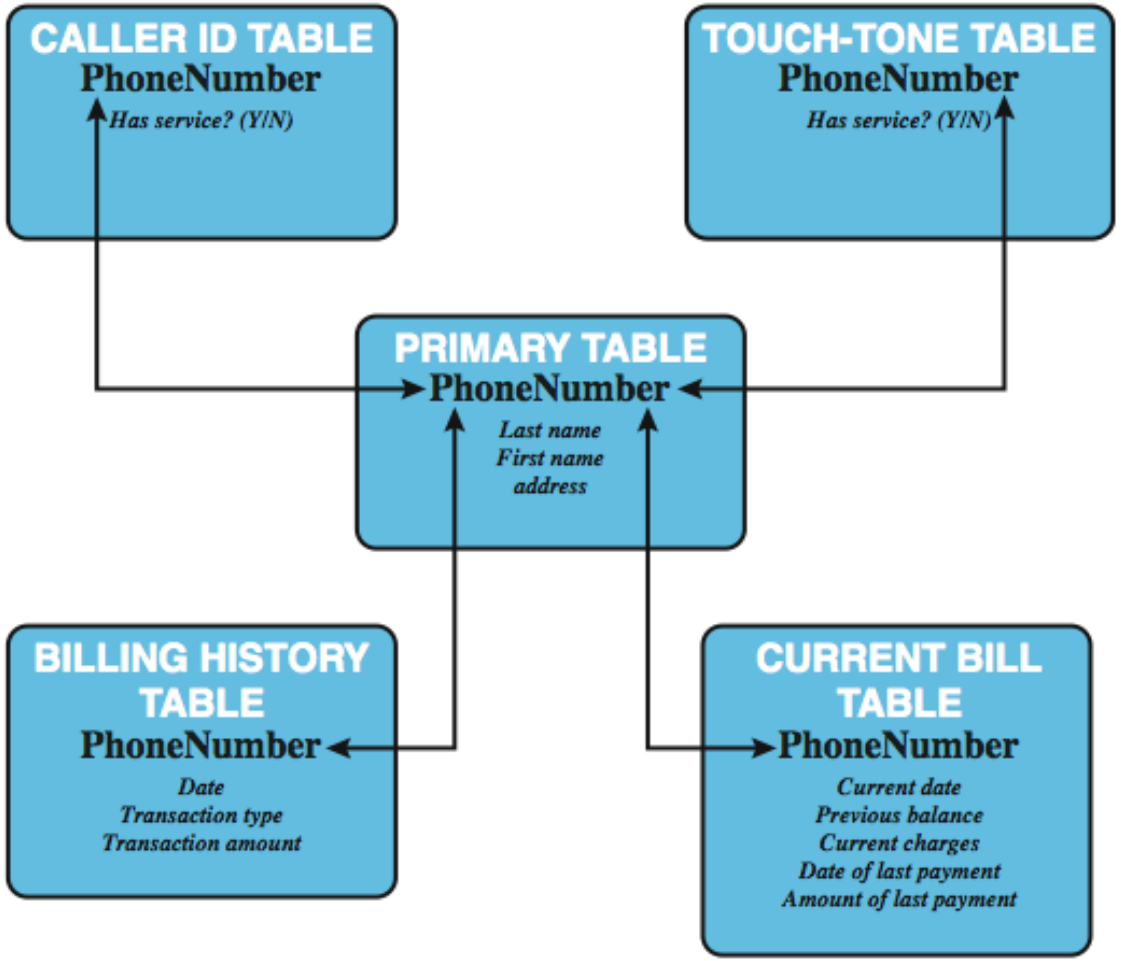
Database Architecture (cont'd)

- In addition to the *database description table*, two other tables support the DBMS.
- The DBMS uses *authorization tables* to ensure the user has permission to execute the query language statement on the database.
- The *concurrent access table* prevents conflicts when simultaneous, conflicting commands are executed.
- *Database systems provide efficient access* to large volumes of data and are vital to the operation of many organizations.
- Because of their complexity and criticality, *database systems generate security requirements that are beyond the capability of typical OS-based security mechanisms* or standalone security packages.

Relational databases

- Table of data consisting of rows and columns
  - Each *column holds a particular type of data*
  - Each *row contains a specific value for each column*
  - Ideally has *one column where all values are unique*, forming an identifier/key for that row
- Enables the creation of *multiple tables linked together by a unique identifier* that is present in all tables
- Use a *relational query language* to access the database
  - Allows the user to request data that fit a given set of criteria

A relational database example



Relational Database Elements

Department Table

Did	Dname	Dacctno
4	human resources	528221
8	education	202035
9	accounts	709257
13	public relations	755827
15	services	223945

primary key

Employee Table

Ename	Did	SalaryCode	Eid	Ephone
Robin	15	23	2345	6127092485
Neil	13	12	5088	6127092246
Jasmine	4	26	7712	6127099348
Cody	15	22	9664	6127093148
Holly	8	23	3054	6127092729
Robin	8	24	2976	6127091945
Smith	9	21	4490	6127099380

foreign key

primary key

(a) Two tables in a relational database

Dname	Ename	Eid	Ephone
human resources	Jasmine	7712	6127099348
education	Holly	3054	6127092729
education	Robin	2976	6127091945
accounts	Smith	4490	6127099380
public relations	Neil	5088	6127092246
services	Robin	2345	6127092485
services	Cody	9664	6127093148

(b) A view derived from the database

Structured Query Language (SQL)

- originally developed by IBM in the mid-1970s
- standardized language to define, manipulate, and query data in a relational database
- several similar versions of ANSI/ISO standard

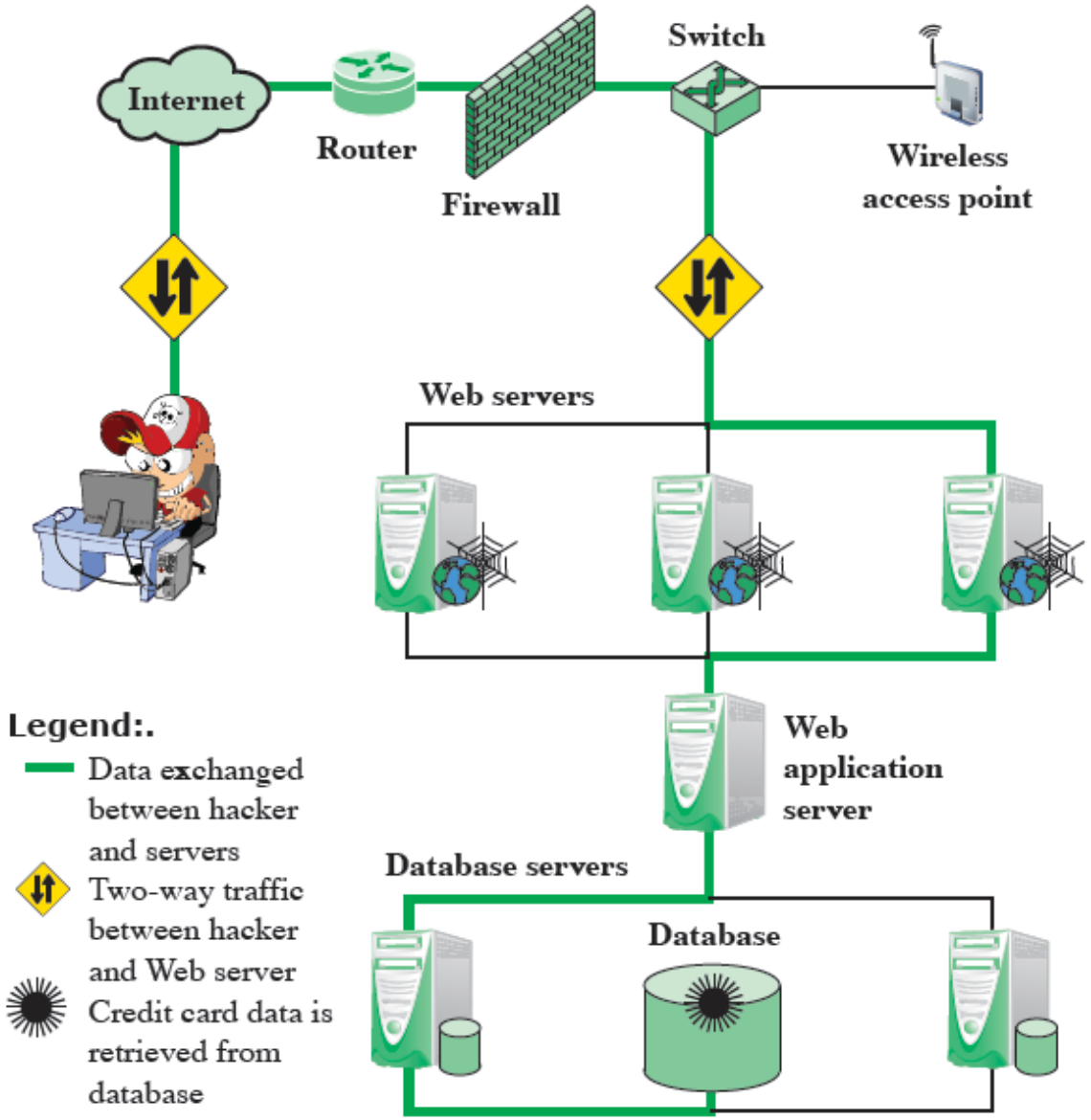
```
CREATE TABLE department (
 Did INTEGER PRIMARY KEY,
 Dname CHAR (30),
 Dacctno CHAR (6)
)

CREATE TABLE employee (
 Ename CHAR (30),
 Did INTEGER,
 SalaryCode INTEGER,
 Eid INTEGER PRIMARY KEY,
 Ephone CHAR (10),
 FOREIGN KEY (Did) REFERENCES department (Did)
)
```

```
CREATE VIEW newtable (Dname, Ename, Eid, Ephone)
AS SELECT D.Dname, E.Ename, E.Eid, E.Ephone
FROM Department D, Employee E
WHERE E.Did = D.Did
```

SQL injection attacks

- One of the *most prevalent and dangerous network-based security threats*
- Sends *malicious SQL commands* to the database server
- Depending on the environment *SQL injection can also be exploited to*:
  - *Modify or delete data*
  - *Execute arbitrary operating system commands*
  - *Launch denial-of-service (DoS) attacks*

A typical injection attack



Injection attack steps

1. Hacker finds a *vulnerability in a custom Web application and injects an SQL command* to a database by sending the command to the Web server. *The command is injected into traffic that will be accepted by the firewall.*
2. The Web server *receives the malicious code* and sends it to the Web application server.
3. The *Web application server receives* the malicious code from the Web server and sends it to the *database server*.
4. The database server *executes the malicious code on the database*. The database *returns data from credit cards table*.
5. The Web application server *dynamically generates a page with data including credit card details* from the database.
6. The Web server then sends the *credit card details to the hacker*

Sample SQL injection

- The SQLi attack typically works by prematurely terminating a text string and appending a new command

```
SELECT fname
FROM student
where fname is  ‘user prompt’;
```

What if user enters the following input?

```
User: John’; DROP table Course;--
```

Sample SQL injection

- ```
  var ShipCity;
  ShipCity = Request.form (“ShipCity”);
  var sql = “select * from OrdersTable where ShipCity = ‘ ” +
  ShipCity + “ ’ ”;
  ```

- ```
  SELECT * FROM OrdersTable WHERE ShipCity = ‘Redmond’
  ```

- Suppose, however, the user enters the following:
  ```
  Redmond’; DROP table OrdersTable --
  ```

- This results in the following SQL query:
  ```
  SELECT * FROM OrdersTable WHERE ShipCity =
  ‘Redmond’; DROP table OrdersTable --
  ```

In-band attacks

- **Tautology**: This form of attack injects code in one or more conditional statements so that they always evaluate to true
- **End-of-line comment**: After injecting code into a particular field, legitimate code that follows are nullified through usage of end of line comments
- **Piggybacked queries**: The attacker adds additional queries beyond the intended query, piggy-backing the attack on top of a legitimate request

Sample SQL injection: tautology

```
$query= “
SELECT info FROM user WHERE name =
`$_GET[“name”]’ AND pwd = `GET[“pwd”]`
”;
```

Attacker enters: `’ OR 1=1 --`

```
SELECT info FROM users WHERE name = ‘ ’ OR 1=1 -- AND pwd = ‘ ’
```

Inferential attack (gathering info)

- There is no actual transfer of data, but the attacker is *able to reconstruct the information by sending particular requests and observing the resulting behavior* of the Website/database server
  - Illegal/logically incorrect queries: lets an attacker gather important information about the type and structure of the backend database of a Web application
    - The vulnerability leveraged by this attack is that the default error page returned by application servers is often overly descriptive.
  - Blind SQL injection: Allows attackers to infer the data present in a database system even when the system is sufficiently secure to not display any erroneous information back to the attacker
  - The attacker asks the server true/false questions to observe the functionality in each case.

SQLi countermeasures

- Defensive coding: Stronger data validation
  - type checking, to check that inputs that are supposed to be numeric contain no characters other than digits
  - pattern matching to try to distinguish normal input from abnormal input
- Detection
  - Signature based
  - Anomaly based
  - Code analysis
- Runtime prevention: Check queries at runtime to see if they conform to a model of expected queries

Detection

- **Signature based:** This technique attempts to match specific attack patterns. Such an approach must be constantly updated and may not work against self-modifying attacks.

- **Anomaly based:** This approach attempts to define normal behavior and then detect behavior patterns outside the normal range. A number of approaches have been used.
  - In general terms, there is a training phase, in which the system learns the range of normal behavior, followed by the actual detection phase.

- **Code analysis:** Code analysis techniques involve the use of a test suite to detect SQLi vulnerabilities.
  - The test suite is designed to generate a wide range of SQLi attacks and assess the response of the system.

Database Access Control

- DBMS provide *access control* for database
- It operates on assumption that *user is already authenticated*

- DBMS provides specific access rights to portions of the database
  - e.g. create, insert, delete, update, read, write
  - to entire database, tables, selected rows or columns
  - possibly dependent on contents of a table entry

- can support a range of policies:
  - centralized administration
  - ownership-based administration
  - decentralized administration

Possible Policies

- **Centralized administration:** A small number of privileged users may grant and revoke access rights.
- **Ownership-based administration:** The owner (creator) of a table may grant and revoke access rights to the table.
- **Decentralized administration:** In addition to granting and revoking access rights to a table, the owner of the table may grant and revoke authorization to other users, allowing them to grant and revoke access rights to the table.

- As with any access control system a database access control system distinguishes different access rights, including create, insert, delete, update, read, and write.
- Some DBMSs provide considerable control over the granularity of access rights.
- Example of access policy:
  - In a personnel database, some users may be limited to seeing salary information only up to a certain maximum value. And a department manager may only be allowed view salary information for employees in his or her department.

SQL Access Controls

- Does the user have access to the entire database or just portions of it?
- Two commands:
  - `GRANT {privileges | role} [ON table] TO {user | role | PUBLIC} [IDENTIFIED BY password] [WITH GRANT OPTION]`
    - e.g. `GRANT SELECT ON ANY TABLE TO john`
  - `REVOKE {privileges | role} [ON table] FROM {user | role | PUBLIC}`
    - e.g. `REVOKE SELECT ON ANY TABLE FROM john`
  - `WITH GRANT OPTION: whether grantee can grant “GRANT” option to other users`
- Typical access rights are:
  - `SELECT, INSERT, UPDATE, DELETE, REFERENCES`

Cascading Authorizations