CAN304 W12 Database security

Database management system

Database

- 数据库是保存在计算机存储中的一组结构化数据，通常通过专用软件进行访问或操作
- Structured data (tables)，Semi-structured data (XML)，Unstructured data (images)
- 包含数据项和数据项组之间的关系；有时可能包含需要保护的敏感数据

DBMS

数据库管理系统（DBMS）是一套程序，用于：

- 建立和维护数据库
- 为多个用户和应用程序提供特设的查询功能

Relational database

关系数据库 (Relational Databases) 由表的集合组成，每个表都分配有一个唯一的名称。

表：表示数据以及这些数据之间的关系。

Relational Database: Views

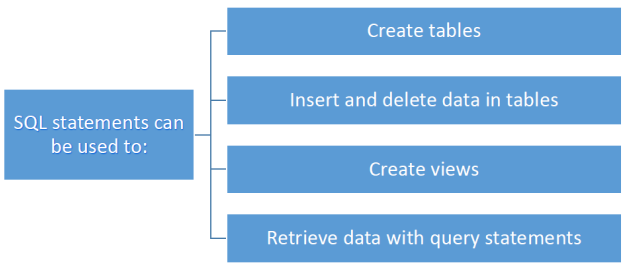
view 是：

- 是由 SQL 查询定义的“虚拟关系”
- 概念上包含查询的结果
- 未预先计算和存储
- 通过在每次使用查询时执行查询来计算

SQL injection attack

Structured Query Language

用于定义关系数据库中的架构、操作和查询数据的标准化语言。



SQL Injection Attacks (SQLi)

- 最普遍和最危险的基于网络安全的威胁之一，旨在利用 Web 应用程序页的性质，向数据库服务器发送恶意 SQL 命令
- 最常见的攻击目标是批量提取数据
- 根据环境的不同，SQL 注入还可能被利用来：修改或删除数据；执行任意操作系统命令；启动拒绝服务 （DoS） 攻击

SQLi Example 1

考虑一个脚本，该脚本通过将预定义的字符串与用户输入的文本组合在一起来构建 SQL 查询：

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

脚本设计者的意图是用户输入城市名称：执行脚本时，提示用户输入城市。

```
• Suppose the user input is $User_Input$
  SELECT *
  FROM OrdersTable
  WHERE ShipCity = ‘$User_Input$’

• $User_Input$ <- Suzhou ’; DROP table OrdersTable--
  SELECT *
  FROM OrdersTable
  WHERE ShipCity = ‘Suzhou’; DROP table OrdersTable--’
```

假如攻击者在输入的时候，多加了一些命令，比如上面的 drop table，那么脚本在执行时会删除表。

注：“--” 代表注释。

SQLi Example 2

考虑打算要求用户输入有效名称和密码的脚本：

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

假设攻击者为名称字段提交“' OR 1=1 --”，现在命令变成了：

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

这导致攻击者一定能成功登录。

Inband Attacks

使用相同的通信通道注入 SQL 代码和检索结果，检索到的数据直接显示在应用程序网页中。包括：

- Tautology
  - 这种形式的攻击在一个或多个条件语句中注入代码，以便它们始终计算为 true
- End-of-line comment
  - 将代码注入特定字段后，通过使用行尾注释，将取消后面的合法代码
- Piggybacked queries
  - 攻击者在预期查询之外添加其他查询，在合法请求之上搭载攻击

Inferential and Out of Band Attacks

- Inferential Attack
  - 没有实际的数据传输，但攻击者能够通过发送特定请求并观察网站/数据库服务器的结果行为来重建信息
- Out-of-Band Attack
  - 使用不同的通道检索数据；当信息检索受到限制，但来自数据库服务器的出站连接松懈时，可以使用此方法

SQLi Countermeasures

Three types:

- Defensive coding
  - 手动防御性编码实践；参数化查询插入；SQL DOM
- Detection
  - Signature based；Anomaly based；Code analysis
- Run-time prevention
  - 在运行时检查查询，查看它们是否符合预期查询的模型

Database access control

Database Access Control

DBMS 可以支持一系列管理策略：

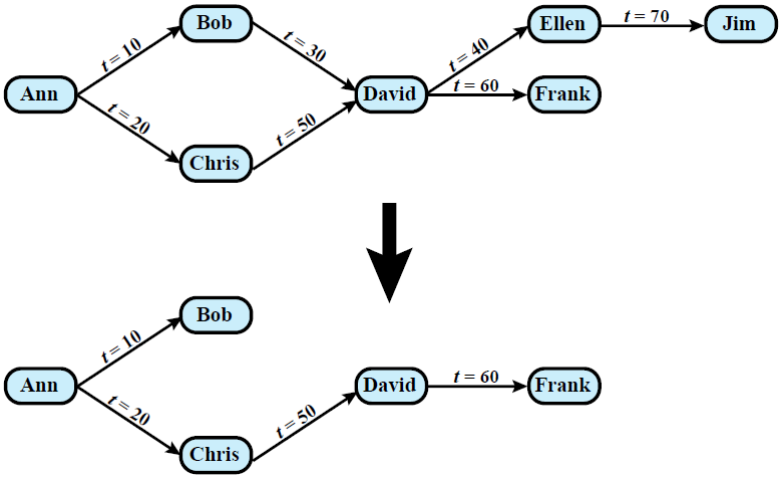
- Centralized administration
  - 少数特权用户可以授予和撤销访问权限
- Ownership-based administration
  - 表的创建者可以授予和撤消对表的访问权限
- Decentralized administration
  - 表的所有者可以向其他用户授予和撤消授权权限，允许他们授予和撤消对表的访问权限

SQL Access Controls

用于管理访问权限的两个命令：Grant，Revoke。

典型的访问权限包括：Select，Insert，Update，Delete，References。

Cascading Authorizations



上图表示 Ann 在 10 时刻授权给 Bob，然后在 20 时刻授权给 Chris ...

现在 Bob 撤销了 David 的特权，结果如下面 (之后被授予的特权也被取消，如 Ellen)。

Role Based Access Control

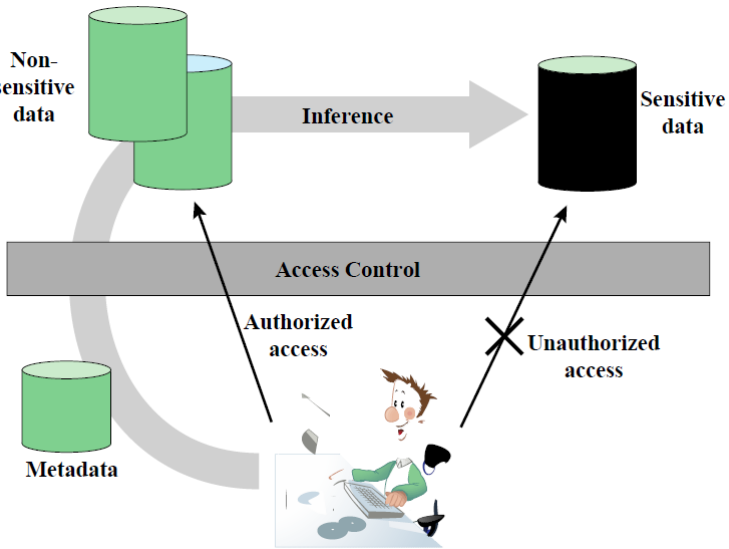
基于角色的访问控制减轻了管理负担，提高了安全性。数据库用户类别：

- Application owner
  - 作为应用程序的一部分拥有数据库对象的最终用户
- End user
  - 通过特定应用程序对数据库对象进行操作，但不拥有任何数据库对象的最终用户
- Administrator
  - 对部分或全部数据库负有管理责任的用户

数据库 RBAC 设施需要提供以下功能：创建和删除角色；定义角色的权限；分配和取消用户对角色的分配。

Inference

通过推理通道间接信息访问 (通过非敏感数据推理出敏感数据)。



Inference techniques：分析表内或表之间属性之间的功能依赖关系；合并具有相同约束的视图。

Inference Detection

- Inference detection during database design
  - 方法通过更改数据库结构或更改访问控制制度来防止推理来删除推理通道
  - 此类别中的技术通常会导致不必要的更严格的访问控制，从而降低可用性
- Inference detection at query time
  - 方法旨在消除查询或一系列查询期间的推理通道违法
  - 如果检测到推理通道，查询将被拒绝或更改

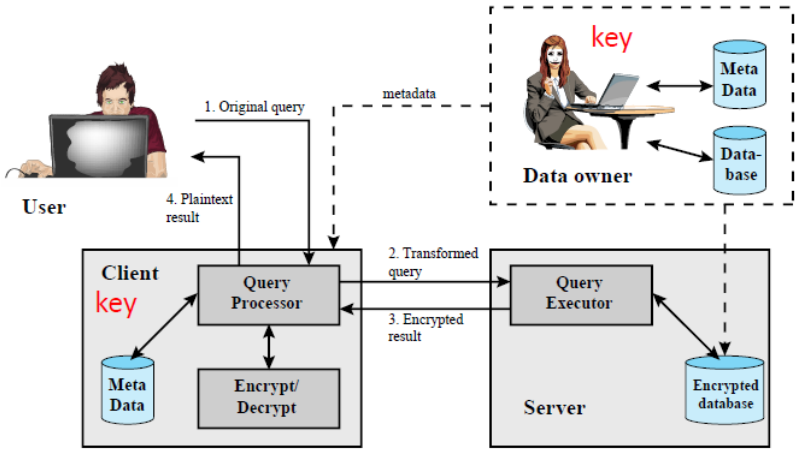
Database encryption

- 加密成为数据库安全的最后一道防线
  - 可应用于整个数据库、记录级别、属性级别或单个字段的级别
- Two disadvantages to encryption
  - Key management：授权用户必须有权访问他们有权访问的数据的解密密钥
  - Inflexibility：当部分或全部数据库被加密时，执行记录搜索变得更加困难

A Database Encryption Scheme

- Data owner - 生成数据以供控制释放的组织
- User - 向系统提供查询的人类实体
- Client - 将用户查询转换为对存储在服务器上的加密数据的查询的前端
- Server - 从数据所有者接收加密数据并使其可用于分发给客户端的组织

假设数据库中的每个单独项目都是单独加密的，所有项目都使用相同的加密密钥。



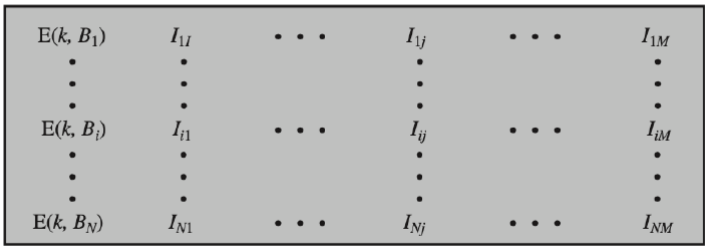
A More Flexible Database Encryption Scheme

数据库表的每条记录（行）都加密为一个块。

$$B_i = (x_{i1} \parallel x_{i2} \parallel \cdots \parallel x_{iM})$$
$$E(k, B_i) = E(k, x_{i1} \parallel x_{i2} \parallel \cdots \parallel x_{iM})$$

为了帮助进行数据检索，属性索引与每个表相关联。为部分或全部属性创建索引值。

$$(x_{i1} \parallel x_{i2} \parallel \cdots \parallel x_{iM}) \rightarrow [E(k, B_i), I_{i1}, I_{i2}, \ldots, I_{iM}]$$



$B_i = (x_{i1} \parallel x_{i2} \parallel \ldots \parallel x_{iM})$

属性值和索引值之间的映射函数构成存储在客户端和数据所有者位置但不存储在服务器上的元数据。