Security properties

- Confidentiality
  - A guarantee that data has not been disclosed to an unauthorized party

- Authenticity
  - A guarantee that data has not been modified by an unauthorized party

- Availability
  - A guarantee that data is available to authorized parties when needed
Access control

- Users grant privileges to other users

**GRANT privilege ON object TO user [WITH GRANT OPTION]**

privilege = SELECT | INSERT | UPDATE | DELETE ...
Examples

GRANT INSERT, DELETE ON Students TO Emma WITH GRANT OPTION

GRANT SELECT ON Employees TO Bob

GRANT UPDATE(rating) ON Restaurants TO Leah

GRANT REFERENCES(cid) ON Company TO Susan
Examples

GRANT INSERT, DELETE ON Students TO Emma WITH GRANT OPTION

GRANT SELECT ON Employees TO Bob

GRANT UPDATE(rating) ON Restaurants TO Leah

GRANT REFERENCES(cid) ON Company TO Susan

Emma is not allowed to run:

```
SELECT gpa
FROM Students
WHERE sid=12345
```
Examples

GRANT INSERT, DELETE ON Students TO Emma WITH GRANT OPTION

GRANT SELECT ON Employees TO Bob

GRANT UPDATE(rating) ON Restaurants TO Leah

GRANT REFERENCES(cid) ON Company TO Susan

Bob is not allowed to insert, update, or delete Employee records
Examples

GRANT INSERT, DELETE ON Students TO Emma WITH GRANT OPTION

GRANT SELECT ON Employees TO Bob

GRANT UPDATE(rating) ON Restaurants TO Leah

GRANT REFERENCES(cid) ON Company TO Susan

Leah can change the rating of a restaurant, but not its address.
Examples

GRANT INSERT, DELETE ON Students TO Emma WITH GRANT OPTION

GRANT SELECT ON Employees TO Bob

GRANT UPDATE(rating) ON Restaurants TO Leah

GRANT REFERENCES(cid) ON Company TO Susan

Susan is allowed to create a foreign key to the Company’s ID field
Views and security

- David has SELECT rights on table Students
- Creates a VIEW BrightStudents
- Grants SELECT rights on BrightStudents to Dan
Revocation

REVOKE [GRANT OPTION FOR] privileges
    ON object FROM users { RESTRICT | CASCADE }

Administrator says:

REVOKE SELECT ON Students FROM David CASCADE

Dan loses SELECT privileges on BrightStudents
Revocation

Joe:  
    GRANT [....] TO Art ...
Art:  
    GRANT [....] TO Bob ...
Bob:  
    GRANT [....] TO Art ...
Joe:  
    GRANT [....] TO Cal ...
Cal:  
    GRANT [....] TO Bob ...
Joe:  
    REVOKE [....] FROM Art CASCADE

What happens??
Revocation

Everyone will keep the privilege
## SQL injection

- Popular attack on databases through web interfaces
- Commonly ranked in the top security vulnerabilities

<table>
<thead>
<tr>
<th>Rank</th>
<th>Score</th>
<th>ID</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>[1]</td>
<td>93.8</td>
<td><strong>CWE-89</strong></td>
<td>Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')</td>
</tr>
<tr>
<td>[2]</td>
<td>83.3</td>
<td><strong>CWE-78</strong></td>
<td>Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection')</td>
</tr>
<tr>
<td>[3]</td>
<td>79.0</td>
<td><strong>CWE-89</strong></td>
<td>Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')</td>
</tr>
</tbody>
</table>

**A1-Injection**

Injection flaws, such as SQL, OS, and LDAP injection occur when untrusted data is sent to an interpreter as part of a command or query. The attacker’s hostile data can trick the interpreter into executing unintended commands or accessing data without proper authorization.

**A2-Broken Authentication and Session Management**

Application functions related to authentication and session management are often not implemented correctly, allowing attackers to compromise passwords, keys, or session tokens, or to exploit other implementation flaws to assume other users’ identities.
## SQL injection: example

<table>
<thead>
<tr>
<th>patient</th>
<th>doctor</th>
<th>date</th>
<th>diagnosis</th>
</tr>
</thead>
<tbody>
<tr>
<td>fred</td>
<td>Dr. Lee</td>
<td>9/1/2005</td>
<td>cancer</td>
</tr>
<tr>
<td>mary</td>
<td>Dr. Lee</td>
<td>5/2/2004</td>
<td>flu</td>
</tr>
<tr>
<td>fred</td>
<td>Dr. Ash</td>
<td>1/18/2005</td>
<td>diabetes</td>
</tr>
<tr>
<td>joe</td>
<td>Dr. Boul</td>
<td>6/4/2005</td>
<td>flu</td>
</tr>
</tbody>
</table>
First login:

- User: fred
- Password: *************

Then search:

- Search claims: Dr. Lee

SQL:

```
SELECT ... FROM ... WHERE doctor = 'Dr. Lee'
AND patient = 'fred'
```
Search claims:

Dr. Lee’ OR patient=‘mary’;  --

... WHERE doctor = ‘Dr. Lee’ OR patient=‘mary’;
   -- ‘ AND patient = ‘fred’
SQL injection

- These attacks threaten confidentiality
- There are also authenticity attacks
  - Often require knowledge of the schema
  - Can be discovered using error messages

Search claims:

Dr. Lee'; DROP TABLE Patients; --
Database humor

HI, THIS IS YOUR SON’S SCHOOL. WE’RE HAVING SOME COMPUTER TROUBLE.

OH, DEAR – DID HE BREAK SOMETHING? IN A WAY –

DID YOU REALLY NAME YOUR SON Robert’); DROP TABLE Students;-- ?

OH, YES. LITTLE BOBBY TABLES, WE CALL HIM.

WELL, WE’VE LOST THIS YEAR’S STUDENT RECORDS. I HOPE YOU’RE HAPPY.

AND I HOPE YOU’VE LEARNED TO SANITIZE YOUR DATABASE INPUTS.

http://xkcd.com/327/
Solutions

- Input validation
  - Check content, length, format
  - Generally a pain... May reject good input

- Stored procedures with parameters

- Move policy implementation from apps to DB
Private data analysis

- Derive useful information from a collection of data without violating the privacy of individual participants

- Oldest form: statistical database
  - Permit statistic queries (e.g., average, count)
Statistical databases

SELECT count(*)
FROM Patients
WHERE age=42
    AND diagnosis='schizophrenia'

SELECT name
FROM Patients
WHERE age=42
    AND diagnosis='schizophrenia'

OK

not OK
Approaches

- **Query restriction**
  - SDB (restricted) queries
  - Exact responses / denials

- **Data perturbation**
  - SDB (perturbed) queries
  - Responses

- **Output perturbation**
  - SDB (restricted) queries
  - Perturbed responses
“Anonymized” data publishing

- In the mid-90s, the Massachusetts Group Insurance Commission (GIC), released hospital visit data of all state employees.

<table>
<thead>
<tr>
<th>Name</th>
<th>SSN</th>
<th>Gender</th>
<th>Zip</th>
<th>DOB</th>
<th>Diagnosis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Joe</td>
<td>123-456</td>
<td>Male</td>
<td>1244</td>
<td>9/1/1957</td>
<td>Cancer</td>
</tr>
<tr>
<td>Mary</td>
<td>987-654</td>
<td>Female</td>
<td>0115</td>
<td>5/2/1983</td>
<td>Flu</td>
</tr>
<tr>
<td>Bob</td>
<td>716-902</td>
<td>Male</td>
<td>1312</td>
<td>1/18/1978</td>
<td>Diabetes</td>
</tr>
<tr>
<td>Charlie</td>
<td>726-918</td>
<td>Male</td>
<td>1126</td>
<td>6/4/1988</td>
<td>Flu</td>
</tr>
</tbody>
</table>

**Sensitive attribute**

**Identifier**
Additional data source

- Latanya Sweeney (a CS grad student at the time) paid $20 to buy the voter registration list for Cambridge MA.

<table>
<thead>
<tr>
<th>name</th>
<th>party</th>
<th>gender</th>
<th>zip</th>
<th>dob</th>
</tr>
</thead>
<tbody>
<tr>
<td>joe</td>
<td>Democrat</td>
<td>male</td>
<td>1244</td>
<td>9/1/1957</td>
</tr>
<tr>
<td>mary</td>
<td>Democrat</td>
<td>female</td>
<td>0115</td>
<td>5/2/1973</td>
</tr>
<tr>
<td>bob</td>
<td>Republican</td>
<td>male</td>
<td>1312</td>
<td>1/18/1978</td>
</tr>
<tr>
<td>charlie</td>
<td>Democrat</td>
<td>male</td>
<td>1126</td>
<td>6/4/1965</td>
</tr>
</tbody>
</table>
Re-identification

- William Weld (the MA governor at the time) lives in Cambridge and is in VOTER
- 6 people in INSURED share his date of birth
- Only 3 were male
- Weld was the only one with that zip
- Sweeney retrieved the governor’s health records, diagnoses, and prescriptions
Pseudo-identifiers

Latanya Sweeney’s finding:

In 2000, 87% of the US population could be uniquely identified based on zip, date of birth, and gender.
K-anonymity

- Intuition: hide individuals in groups of sufficient size

- Alter data so that:
  - At least k individuals share the same values of the pseudo-identifiers in the database
  - Attribute suppression and generalization
### K-anonymity example

|  | Non-Sensitive | | Sensitive | | Non-Sensitive | | Sensitive |
|---|---|---|---|---|---|---|---|---|
|  | Zip Code | Age | Nationality | Condition | Zip Code | Age | Nationality | Condition |
| 1 | 13053 | 28 | Russian | Heart Disease | 1 | 130** | < 30 | * | Heart Disease |
| 2 | 13068 | 29 | American | Heart Disease | 2 | 130** | < 30 | * | Heart Disease |
| 3 | 13068 | 21 | Japanese | Viral Infection | 3 | 130** | < 30 | * | Viral Infection |
| 4 | 13053 | 23 | American | Viral Infection | 4 | 130** | < 30 | * | Viral Infection |
| 5 | 14853 | 50 | Indian | Cancer | 5 | 1485* | ≥ 40 | * | Cancer |
| 6 | 14853 | 55 | Russian | Heart Disease | 6 | 1485* | ≥ 40 | * | Heart Disease |
| 7 | 14850 | 47 | American | Viral Infection | 7 | 1485* | ≥ 40 | * | Viral Infection |
| 8 | 14850 | 49 | American | Viral Infection | 8 | 1485* | ≥ 40 | * | Viral Infection |
| 9 | 13053 | 31 | American | Cancer | 9 | 130** | 3* | * | Cancer |
| 10 | 13053 | 37 | Indian | Cancer | 10 | 130** | 3* | * | Cancer |
| 11 | 13068 | 36 | Japanese | Cancer | 11 | 130** | 3* | * | Cancer |
| 12 | 13068 | 35 | American | Cancer | 12 | 130** | 3* | * | Cancer |

Original data 4-anonymous data

Susceptible to homogeneity and background knowledge attacks

[Machanavajjhala, Gehrke, Kifer. ICDE 06]
Differential privacy

- **Intuition:**
  - Statistics released are unchanged whether or not an individual is in the database.
Differential privacy

A randomized algorithm $A$ provides \textbf{ε-differential privacy}, if for all neighboring DBs $D$ and $D'$ and any set of outputs $S$:

$$\Pr[A(D) \in S] \leq e^{\epsilon} \Pr[A(D') \in S]$$

\textbf{lower } $\epsilon$ = stronger privacy