

Database Security

CS 445

Fall 2008

Gerome Miklau, UMass Amherst

Outline

- **Security basics**
- Access control in Databases
- Beyond access control
- Privacy in data publishing

Security properties

- Confidentiality
- Authenticity
- Availability
- Privacy

Confidentiality

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

- Threats to confidentiality
 - direct release, approximate disclosure, leaks from inference & outside knowledge
- Providing confidentiality
 - access controls, inference controls, encryption

Authenticity

Also called: *data integrity*

A guarantee that data has not been modified from its original state by an unauthorized party.

- Aspects of authenticity:
 - data comes from original source
 - not modified
 - freshness: current, not re-used
- Threats to authenticity:
 - forging, tampering, replay
- Providing authenticity
 - access control, digital signatures, hashing

Confidentiality or Authenticity ?

- Which security properties matter for:
 - Student grades for this course stored in database.
 - Resume or CV posted on webpage.
 - Medical records stored in database.

- Security basics
- **Access control in Databases**
- Beyond access control -- SQL Injection
- Privacy in data publishing

Access control

- Regulates direct access to resources
 - Subjects (i.e. registered users)
 - Objects (files, directories, tables)
 - Privileges (read, write, insert, delete, etc.)
- **Discretionary** access control
 - Users can grant access at their discretion.
- **Mandatory** access control
 - All subjects and objects classified by an authority and global rules determine privileges.

SQL Security

- Core security features present in nearly all database systems:
 - User authentication
 - Discretionary access control:
 - Subjects (database users)
 - Privileges (select, insert, delete, update)
 - Objects (tables, columns, **views**)
 - In SQL: **GRANT** / **REVOKE**

System R authorization model [Griffith and Wade'76], [Fagin'78]

Discretionary AC in SQL

GRANT privileges ON object TO users
[WITH GRANT OPTIONS]

privileges = SELECT |
 INSERT(column-name) |
 UPDATE(column-name) |
 DELETE |
 REFERENCES(column-name)

object = table | view

Examples

GRANT INSERT, DELETE ON Customers TO **Yuppy**
WITH GRANT OPTIONS

Queries allowed to
Yuppy:

```
INSERT INTO Customers(cid, name, address)  
VALUES(32940, 'Joe Blow', 'Seattle')
```

```
DELETE Customers WHERE  
LastPurchaseDate < 1995
```

Queries denied to
Yuppy:

```
SELECT Customer.address  
FROM Customer  
WHERE name = 'Joe Blow'
```

Examples

GRANT SELECT ON Customers TO **Michael**

Now **Michael** can SELECT,
but not INSERT or DELETE

Examples

```
GRANT SELECT ON Customers TO Michael  
WITH GRANT OPTIONS
```

Michael can say this:

```
GRANT SELECT ON Customers TO Yuppi
```

Now **Yuppi** can SELECT on Customers

Examples

GRANT UPDATE (price) ON Product TO Leah

Leah can update, but only Product.price,
but not Product.name

Examples

Customer(cid, name, address, balance)

Orders(oid, cid, amount) cid= foreign key

Bill has INSERT/UPDATE rights to Orders.
BUT HE CAN'T INSERT ! (why ?)

GRANT REFERENCES (cid) ON
Customer TO **Bill**

Now **Bill** can INSERT tuples into Orders

Views and Security

David owns

Customers:

Name	Address	Balance
Mary	Huston	450.99
Sue	Seattle	-240
Joan	Seattle	333.25
Ann	Portland	-520

Fred is not
allowed to
see this

```
CREATE VIEW PublicCustomers  
  SELECT Name, Address  
  FROM Customers;
```

```
GRANT SELECT ON PublicCustomers TO Fred
```

David says

David owns

Views and Security

Customers:

Name	Address	Balance
Mary	Huston	450.99
Sue	Seattle	-240
Joan	Seattle	333.25
Ann	Portland	-520

John is
allowed to
see only <0
balances

David says

```
CREATE VIEW BadCreditCustomers
SELECT *
FROM Customers
WHERE Balance < 0;
GRANT SELECT ON BadCreditCustomers TO John
```

Views and Security

David says

- Each customer should see only her/his records

Name	Address	Balance
Mary	Huston	450.99
Sue	Seattle	-240
Joan	Seattle	333.25
Ann	Portland	-520

```
CREATE VIEW CustomerMary
```

```
SELECT * FROM Customers  
WHERE name = 'Mary'
```

```
GRANT SELECT  
ON CustomerMary TO Mary
```

```
CREATE VIEW CustomerSue
```

```
SELECT * FROM Customers  
WHERE name = 'Sue'
```

```
GRANT SELECT  
ON CustomerSue TO Sue
```

Doesn't scale.

Need *row-level* access
control !

Summary of SQL Security

Limitations:

- No row level access control
- Table creator owns the data

Access control = great success story of the DB community...

... or spectacular failure:

- Only 30% assign privileges to users/roles
 - And then to protect entire tables, not columns

- Security basics
- Access control in Databases
- **Beyond access control**
 - SQL Injection
- Privacy in data publishing

SQL Injection

- Popular attack on databases accessed through web interfaces.
- Attacker is able to insert SQL statements into a query by manipulating application input data.
- Ranked as a top 10 security vulnerability
 - SANS Institute
 - Open Web Application Security Project (OWASP)

Top Vulnerabilities in Web Applications		
A1	Unvalidated Input	Information from web requests is not validated before being used by a web application. Attackers can use these flaws to attack backend components through a web application.
A2	Broken Access Control	Restrictions on what authenticated users are allowed to do are not properly enforced. Attackers can exploit these flaws to access other users' accounts, view sensitive files, or use unauthorized functions.

SQL injection: example

Treatment table

patient	doctor	date	diagnosis
fred	Dr. Lee	9/1/2005	cancer
mary	Dr. Lee	5/2/2004	flu
fred	Dr. Ash	1/18/2005	diabetes
joe	Dr. Boul	6/4/2005	flu

SQL Injection

Your health insurance company has a web site for claims:

First login:

User:	<input type="text" value="fred"/>
Password:	<input type="password" value="*****"/>

Then search:

Search claims:	<input type="text" value="Dr. Lee"/>
----------------	--------------------------------------

SELECT...FROM...WHERE doctor='Dr. Lee' and
patient='fred'

SQL Injection

Now try this:

Search claims: `Dr. Lee' OR patientID = 'mary'; --`

.....WHERE doctor='Dr. Lee' OR patientID='mary'; --' and patientID='fred'

Even better:

Search claims: `Dr. Lee' OR 1 = 1; --`

SQL Injection

- Those attacks threaten **confidentiality**
- There are also **authenticity** attacks
 - these often require knowledge of schema
 - can be discovered using error messages!

Extreme case:

Search claims: `Dr. Lee'; DROP TABLE Patients; --`

Determining schema

Illegal GROUP BY can reveal column names

**Microsoft OLE DB Provider for ODBC Drivers error '80040e14'
[Microsoft][ODBC SQL Server Driver][SQL Server]Column
'users.username' is invalid in the select list because it is not
contained in either an aggregate function or the GROUP BY clause.**

Illegal UNION can reveal data types

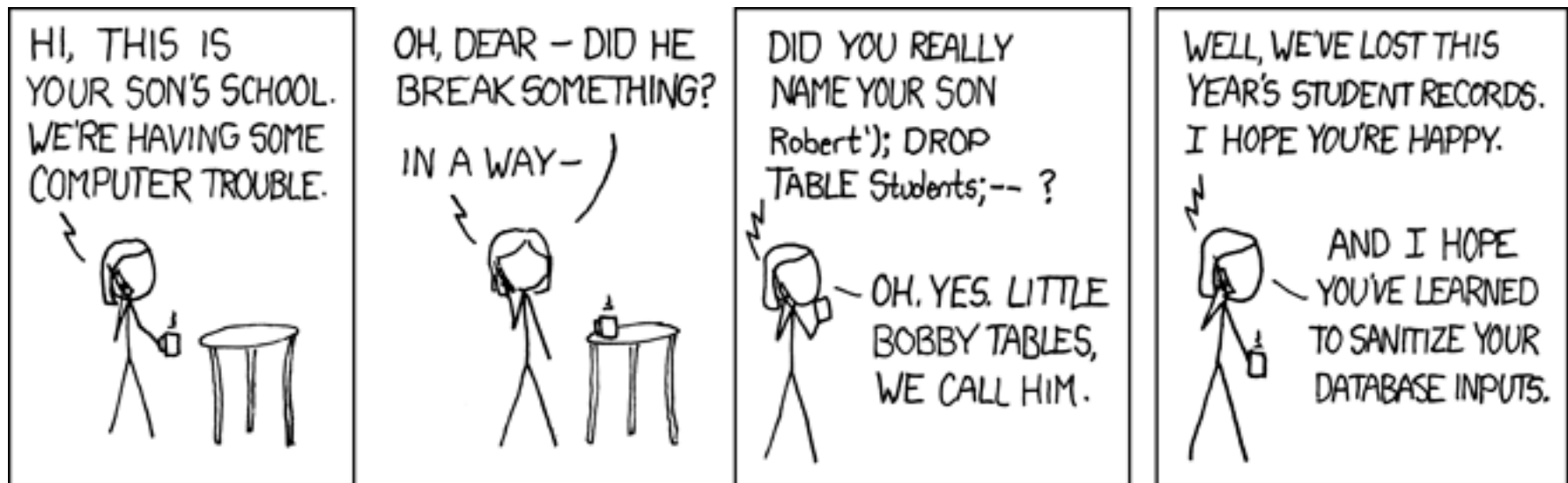
Username: ' union select sum(username) from users--

**Microsoft OLE DB Provider for ODBC Drivers error '80040e07'
[Microsoft][ODBC SQL Server Driver][SQL Server]The sum or
average aggregate operation cannot take a varchar data type as
an argument.**

Solutions

- Input validation
 - check content, length, format
 - generally a pain, hard to check, may accidentally reject good input
- Stored procedures with parameters
- Deeper answer:
 - Move policy implementation from apps to DB

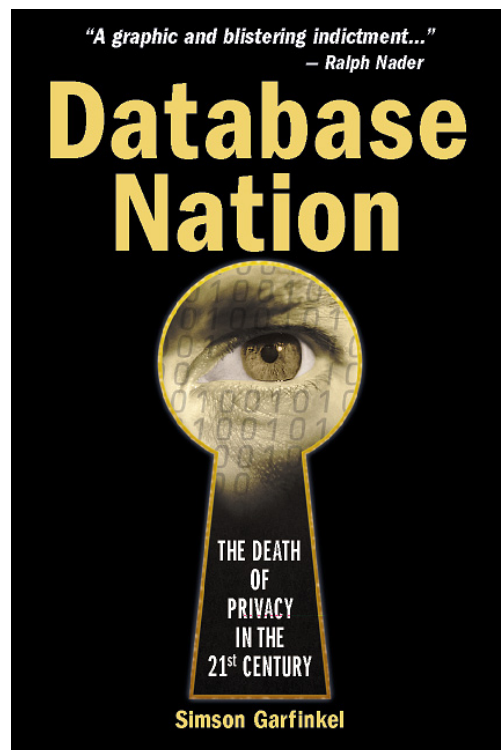
Database humor



Next

- Security basics
- Access control in Databases
- Beyond access control
- **Privacy in data publishing**

Definitions of privacy



Informational privacy

- Westin, 1967:
 - *the ability to determine for ourselves when, how, and to what extent information about us is communicated to others.*
- Hughes, 1993:
 - *the power to selectively reveal oneself to the world.*

Privacy, **For** and Against

- A fundamental human right
- Aspect of personal freedom, liberty
- Requirement for democracy
- Prerequisite to developing sense of self.

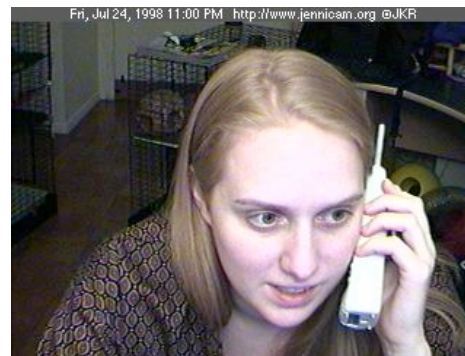
Privacy, For and Against

- Surveillance has benefits
- Accountability requires sacrificing privacy
- Posner (1981) - Economic critique
- MacKinnon (1989) - Feminist critique
- Brin (1999) Increased flow of information can benefit all, if access is free and equal.

Attitudes

- Westin's categories, through surveys
 - Privacy fundamentalist (25%)
 - Feel they've lost privacy, resistant to further erosion
 - Privacy pragmatist (55%)
 - Concerned about privacy, but willing to share info given choice and notice
 - Privacy unconcerned (20%)

Jennicam, 1996

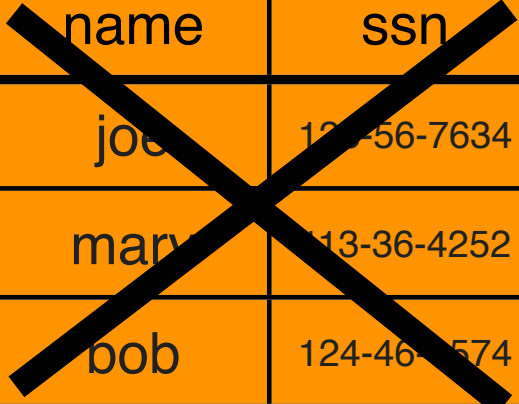


Behavior != Attitude

- Behavior is not always consistent with stated attitudes
 - economic model of behavior - rational economic agents protecting or divulging their personal info
 - price of privacy
- Individuals are not rational actors

“Anonymized” data publishing

- Mass. Group Insurance Commission (GIC) is responsible for purchasing health insurance for state employees
- GIC collects data, and publishes it:



name	ssn	gender	dob	zip	diagnosis
joe	123-56-7634	male	1/4/64	1045	cancer
mary	413-36-4252	female	3/24/45	1312	flu
bob	124-46-574	male	5/4/55	1452	HIV

↑
Identifier

↑
Sensitive Attr.

Additional data source

- Sweeney paid \$20 and bought the public voter registration list for Cambridge Mass.:

VOTER

name	party	gender	dob	zip
joe	Dem	male	1/4/64	1045
mary	Rep	female	3/24/45	1312
bob	Dem	male	5/4/55	1452

Re-identification

INSURED(zip, dob, sex, diagnosis, procedure, ...)
VOTER(name, party, ..., zip, dob, sex)

- William Weld (former governor) lives in Cambridge, hence is in VOTER
- 6 people in INSURED share his dob
- only 3 of them were male (same gender)
- Weld was the only one in that zip
- Sweeney learned Weld's medical records !

Pseudo-identifiers

Latanya Sweeney's Finding

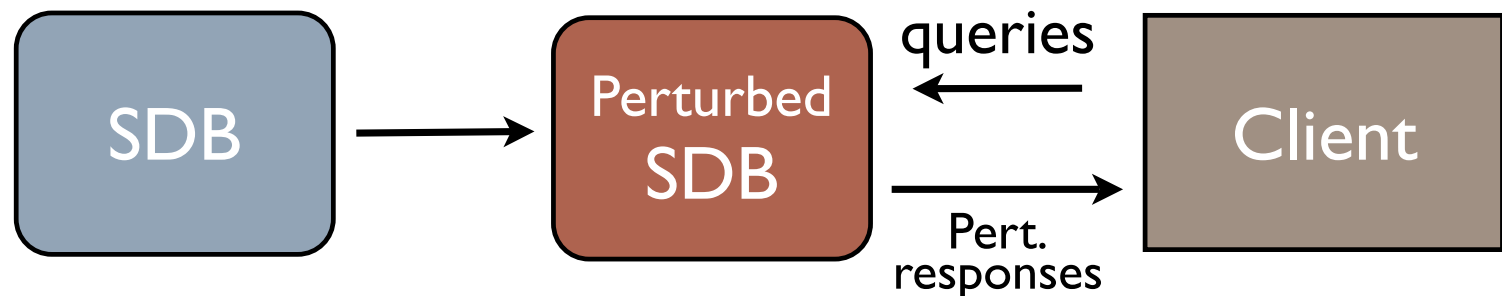
87% of the US population (216 million out of 248 million) are likely to be uniquely identified based on:

zipcode, gender, date-of-birth

K-Anonymity

- Intuition: privacy is gained by hiding individuals in groups of sufficient size
- Alter data so that:
 - At least k individuals share each pseudo-identifier occurring in the database.
 - Attribute **suppression** and **generalization**

Data perturbation

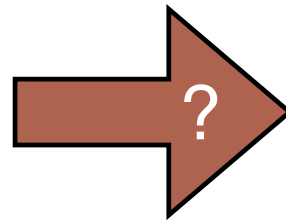


K-anonymity example

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

	Non-Sensitive			Sensitive
	Zip Code	Age	Nationality	Condition
1	130**	< 30	*	Heart Disease
2	130**	< 30	*	Heart Disease
3	130**	< 30	*	Viral Infection
4	130**	< 30	*	Viral Infection
5	1485*	≥ 40	*	Cancer
6	1485*	≥ 40	*	Heart Disease
7	1485*	≥ 40	*	Viral Infection
8	1485*	≥ 40	*	Viral Infection
9	130**	3*	*	Cancer
10	130**	3*	*	Cancer
11	130**	3*	*	Cancer
12	130**	3*	*	Cancer

Original data



4-anonymous data

Analysis

	Non-Sensitive			Sensitive
	Zip Code	Age	Nationality	Condition
1	130**	< 30	*	Heart Disease
2	130**	< 30	*	Heart Disease
3	130**	< 30	*	Viral Infection
4	130**	< 30	*	Viral Infection
5	1485*	≥ 40	*	Cancer
6	1485*	≥ 40	*	Heart Disease
7	1485*	≥ 40	*	Viral Infection
8	1485*	≥ 40	*	Viral Infection
9	130**	3*	*	Cancer
10	130**	3*	*	Cancer
11	130**	3*	*	Cancer
12	130**	3*	*	Cancer

- Higher k -- more privacy
- Fewer suppressions & generalizations -- more accuracy

Attacks on anonymized data

	Non-Sensitive			Sensitive
	Zip Code	Age	Nationality	Condition
1	130**	< 30	*	Heart Disease
2	130**	< 30	*	Heart Disease
3	130**	< 30	*	Viral Infection
4	130**	< 30	*	Viral Infection
5	1485*	≥ 40	*	Cancer
6	1485*	≥ 40	*	Heart Disease
7	1485*	≥ 40	*	Viral Infection
8	1485*	≥ 40	*	Viral Infection
9	130**	3*	*	Cancer
10	130**	3*	*	Cancer
11	130**	3*	*	Cancer
12	130**	3*	*	Cancer

- Homogeneity attack
 - Alice knows Bob is 31, living in zip 13053
- Background knowledge attack
 - Alice has japanese friend who is 21 and living in 13068.