INF1343, Winter 2012

# Data Modeling and Database Design

Yuri Takhteyev
Faculty of Information
University of Toronto

# Week 11

# Database Security

# Final Project

## Implementation

plan for unexpected problems
the database is what matters most
backup your stuff
test!

## Documentation

update the ER in the end
include all SQL (even if it doesn't work)
explain what is not obvious design
explain non-obvious testing steps

## Questions?

# Security

Confidentiality / secrecy:

Preventing unwanted disclosure

Integrity

Preventing unwanted modification

Availability

Preventing disruption or loss

(Consider the interrelation between the three.)

# Threats

Attacks

- Profit
- Revenge/terrorism
- Lulz

Mistakes / accidents / oversights

Outsiders vs insiders

- Insiders are armed with internal knowledge.
- Consider “social engineering”.

# Cryptography

**Encryption**

makes text unreadable without a key

**Digital Signature**

a “stamp of approval” for text

**Hash / digest**

identifies text without revealing it’s content

# Encryption

## Symmetric Cyphers

same key for encryption and decryption
e.g.: AES

## Public Key / Private Key

a public key for encryption,
a private key for decryption

# AES in MySQL

```
create table user (
username varchar(100),
ccnumber varbinary(32)
);
```

---

```
insert into user values ("bob",
aes_encrypt("1111222233334444",
"May the Force be with you!"));
```

---

```
select aes_decrypt(ccnumber,
"May the Force be with you!")
from user;
```

# Access Control

Define who can do/see what

1. Identifying users
   - Passwords + other mechanisms
   - Users can be assigned to groups/roles

2. Determining rights
   - Grant/deny rights to specific data/operations
   - Could be done at the level of group/role

# SQL Privileges

```
grant «privilege» on «object»
to «user»;
```

```
grant select on secretdb.*
to okenobi;
```

```
grant select on secretdb.*
to okenobi with grant option;
```

# “Flushing” Privileges

```
flush privileges;
```

# SQL Privileges

```
revoke «privilege» on «object»
from «user»;
```

---

```
revoke select on secretdb.*
from okenobi;
```

Also (not in MySQL):

```
revoke select on secretdb.*
from okenobi cascade;
```

# SQL Privileges

Variations:

```
grant select on secretdb.alpha
to okenobi;
```

The user can see table “alpha” but not the other tables.

# Roles

More flexible:

```
create role student;
grant select on secretdb.*
to student;
grant student to okenobi;
```

(Not available in MySQL.)

# Using Views

```
create table true_omega
(boring TEXT, scary TEXT);

create view omega as
select boring from true_omega;

grant all on secretdb.omega
to okenobi;
```

The user can now see and modify the values in “boring” but not in “scary”.

# Using Views

```
create view omega as
select boring from true_omega
where scary is null;
```

```
grant all on secretdb.omega
to okenobi;
```

The user can now see and modify the values in “boring” but only for those roles for where “scary” is not set.

# Limitations

## Somewhat inflexible

- 1000 employees = 1000 views?
- Solution: “fine-grained access control”

## Discretionary

- Can be bypassed by users with access:
  - Obvious: “with grant option”
  - Less obvious: users with clearance can copy sensitive data into tables accessible to users without clearance
- Solution: “mandatory access control”
  - i.e., a centrally enforced row-level policy

# Creating Users

```
create user alice;
set password for alice
= password('bob');
```

You can now login as 'alice' with passworc 'bob':

```
mysql -p -u alice
```

BTW, to change your own password:

```
set password = password('newpass');
```

# Storing User Data

```
use mysql;

select User, Password
from user where User="alice";
```

You don't have access to database “mysql,” but here is what I get to see:

```
+-------+-------------------------------------------+
| User  | Password                                  |
+-------+-------------------------------------------+
| alice | *61584B76F6ECE8FB9A328E7CF198094B2FAC55C7 |
+-------+-------------------------------------------+
1 row in set (0.00 sec)
```

# Hashing Passwords

```
set password for alice
= password('bob');
```

Compare with:

```
select password('bob');
Select sha1('bob');
```

# Hashing Passwords

**A “one-way” function**

bob → 61584B76F6ECE8FB9A328E7CF198094B2FAC55C7

Easily calculated.

61584B76F6ECE8FB9A328E7CF198094B2FAC55C7 → bob

Very hard to calculate (in theory).

How hard is “very hard”?

| | |
|---|---|
| MD5: | a few seconds |
| SHA1/2: | 1 hour + $2 |
| SHA3: | hopefully centuries |

# Password Attacks

## A brute-force attack

- Just try all possible combinations
  - 5 lower case letters: 12 million combinations
  - 5 letters / numbers: ~ 1 billion
  - 8 letters / numbers: > 200 trillion

## A dictionary attack

- Check passwords based on words

## Matching passwords to accounts

- Who uses “123456” as their password?

# Password Trilemma

- Pick a simple one
  - Might be guessed or brute-forced
- Write it down
  - Might be found
- Forget it later
  - Need a procedure for recovery

# Password Theft

- Stored passwords
  - Paper, email accounts, files
- Insecure transmission
  - “Sniffing”, “key-loggers”
- Get a user to reveal the password
  - Phishing, social engineering

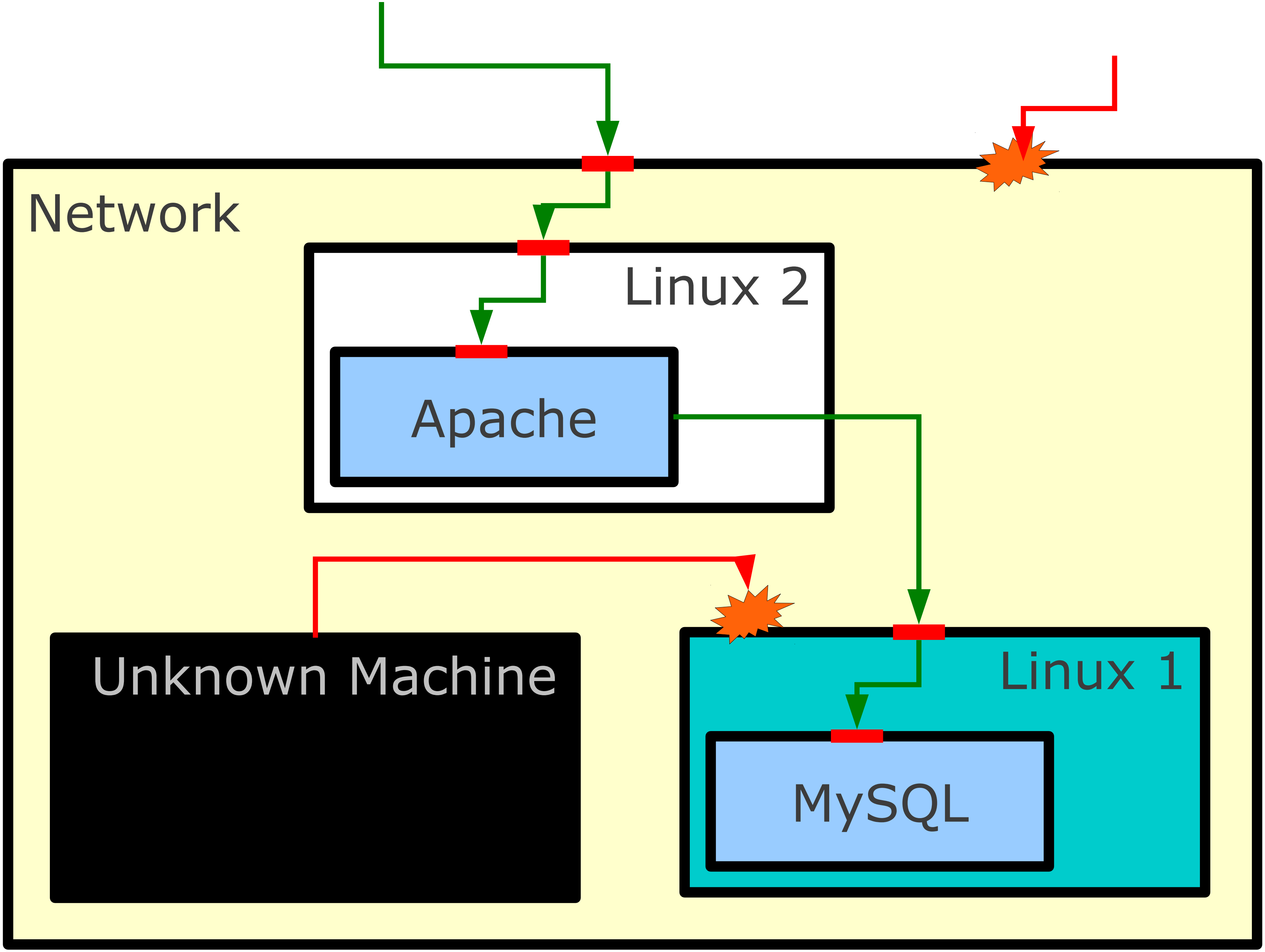
# Password Alternatives

## Public key cryptography

- Authentication *without* revealing any secrets
- A pair of keys (long numbers):
  - A public key for encoding
  - A private key for decoding

## RSA SecurID:

image source: http://en.wikipedia.org/wiki/File:SecureID_token_new.JPG


Network
Linux 2
Apache
Unknown Machine
Linux 1
MySQL

# Web App Accounts

Create a table for user accounts:

- usernames
- hashed passwords
  - never store actual passwords!

Authenticate the user with a query

```
select * from users where
username="%s" AND password="%s";
```

Proceed if successful

- Give the user a token for the future, e.g. as a “cookie”
  - cookies can be stolen (unless encrypted w/ SSL)

# The Accounts

MySQL accounts

to access the database
(from an authorized machine)

Machine accounts

to access a machine
(from within the network)

Application accounts

your users, e.g. bank clients

# Gawker Media

## Hacked by “Gnosis” in 2010

1.25 million accounts stolen.

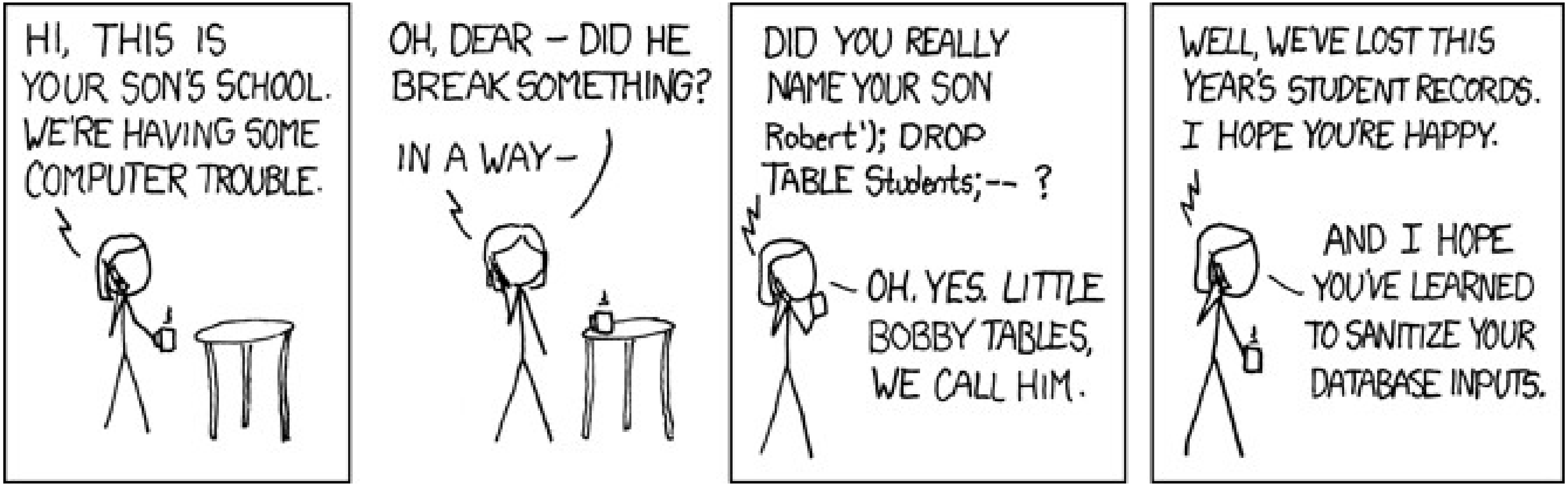
300,000 weakly hashed passwords released.

(Most common ones: “12345”, “password”)

## The scenario:

hacked email → access to chat logs chat →
→ sysadmin passwords → access to servers →
→ root access (unpatched OS) →
→ MySQL access →
→ MySQL data, including passwords →
→ Twitter logins, etc. → Twitter spam

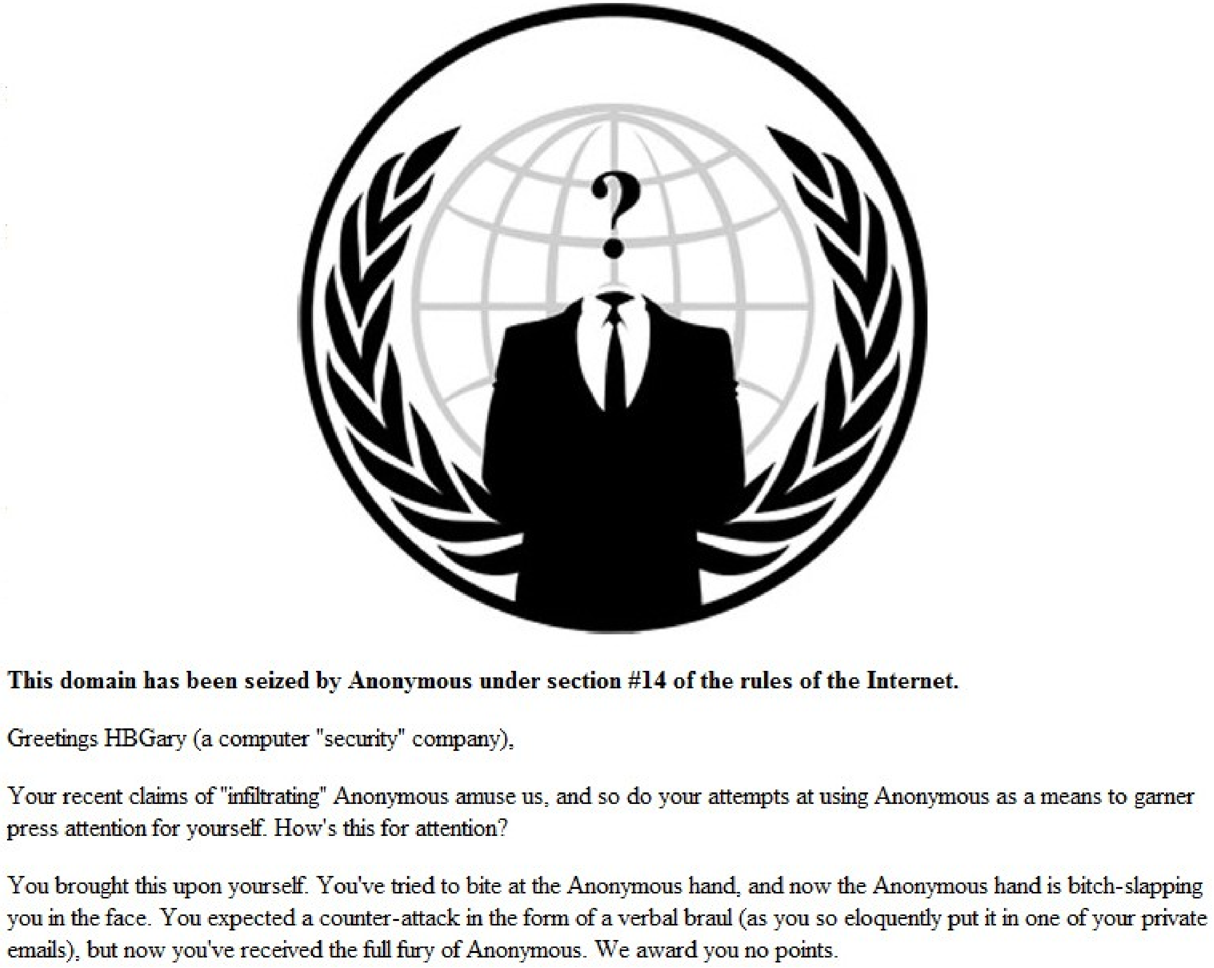
# SQL Injection


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.

Did you really name your son
*Robert'); DROP TABLE Students;--?*

Image source: XKCD, http://xkcd.com/327/

# Anonymous vs HBGary



**This domain has been seized by Anonymous under section #14 of the rules of the Internet.**

Greetings HBGary (a computer "security" company),

Your recent claims of "infiltrating" Anonymous amuse us, and so do your attempts at using Anonymous as a means to garner press attention for yourself. How's this for attention?

You brought this upon yourself. You've tried to bite at the Anonymous hand, and now the Anonymous hand is bitch-slapping you in the face. You expected a counter-attack in the form of a verbal braul (as you so eloquently put it in one of your private emails), but now you've received the full fury of Anonymous. We award you no points.

(An SQL injection.)

# Cablegate

260,000 “cables”

> “I would come in with music on a CD-RW labelled with something like ‘Lady Gaga’… erase the music … then write a compressed split file. No one suspected a thing … [I] listened and lip-synched to Lady Gaga’s Telephone while exfiltrating possibly the largest data spillage in American history.”

Solutions?

# Physical Attack

http://bit.ly/server_theft

(Also: accidents, natural disasters, hardware failure, file corruption.)

# Backup

mysqldump

```
mysqldump -u okenobi -p starwars > starwars.sql
mysql < starwars.sql
```

(Add “create database starwars; use starwars;”)

hot backup

replication

off-site copies

# RAID

combining disks for reliability
and/or performance

# Cloud Storage

E.g., Amazon S3
(http://aws.amazon.com/s3)

# Questions?