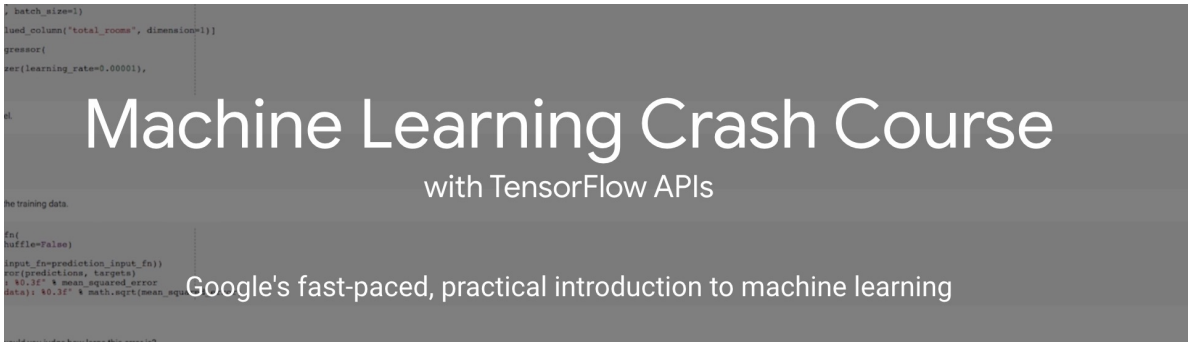
Lecture 14
Security and Privacy

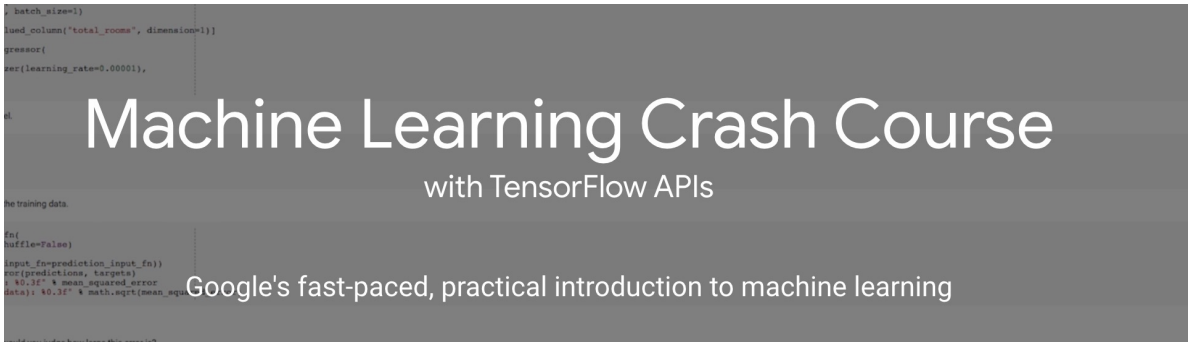
Lampros Flokas



If you had to prioritize improving one of the areas below in your machine learning project, which would have the most impact?

- A more clever loss function
- A deeper network
- The quality and size of your data
- Using the latest optimization algorithm

https://developers.google.com/machine-learning/data-prep



If you had to prioritize improving one of the areas below in your machine learning project, which would have the most impact?

A more clever loss function

A deeper network

The quality and size of your data ✓

Data trumps all. It's true that updating your learning algorithm or model architecture will let you learn different types of patterns, but if your data is bad, you will end up building functions that fit the wrong thing. The quality and size of the data set matters much more than which shiny algorithm you use.

Correct answer.

Using the latest optimization algorithm

Security

SQL Injection

Privacy vs Security

Access Controls

Encryption

SQL Injection

Pass *sanitized* values to the database

```
args = ('Dr Seuss', '40')
conn1.execute(
    "INSERT INTO users(name, age) VALUES(%s, %s)",
    args)
```

Pass in a tuple of query arguments

DBAPI library will *properly escape* input values

Most libraries support this

Never construct raw SQL strings

SQL Injection

Why pass values using query parameters?

```
name = “eugene”

conn1.execute(
  “SELECT * FROM users WHERE name=%s”, name)
```

```
conn1.execute(
  “SELECT * FROM users WHERE name=‘{name}’”.format(name=name))

   SELECT * FROM users WHERE name=‘eugene’
```

SQL Injection

Why pass values using query parameters?

```
name = "eugene';\nDELETE * FROM users;--"

conn1.execute(
  "SELECT * FROM users WHERE name=%s", name)
```

```
conn1.execute(
  "SELECT * FROM users WHERE name='{name}'".format(name=name))
```

```
SELECT * FROM users WHERE name='eugene';
DELETE * FROM users;
--'
```

FEATURES | By Joseph Cox | Nov 20 2015, 8:00am

The History of SQL Injection, the Hack That Will Never Go Away

Over 15 years after it was first publicly disclosed, SQL injection is still the number one threat to websites.

https://motherboard.vice.com/en_us/article/aekzez/the-history-of-sql-injection-the-hack-that-will-never-go-away

sql injection

FILTER

Running an SQL Injection Attack - Computerphile

Computerphile 1.6M views • 2 years ago

Just how bad is it if your site is vulnerable to an **SQL Injection**? Dr Mike Pound shows us how they work. Cookie Stealing: ...

SQL Injection Attack Tutorial (2019)

HackHappy • 76K views • 8 months ago

SQL Injection attacks are still as common today as they were ten years ago. Today I'll discuss what are SQLi and how you can ...

CC

SQL Injection Basics Demonstration

Imperva • 360K views • 9 years ago

Imperva presents an educational video series on Application and Database Attacks in High Definition (HD)

Hacking Websites with SQL Injection - Computerphile

Computerphile 1.4M views • 5 years ago

Websites can still be hacked using **SQL injection** - Tom explains how sites written in PHP (and other languages too) can be ...

CC

DEFCON 17: Advanced SQL Injection

Christiaan008 • 220K views • 8 years ago

Speaker: Joseph McCray Founder of Learn Security Online **SQL Injection** is a vulnerability that is often missed by web application ...

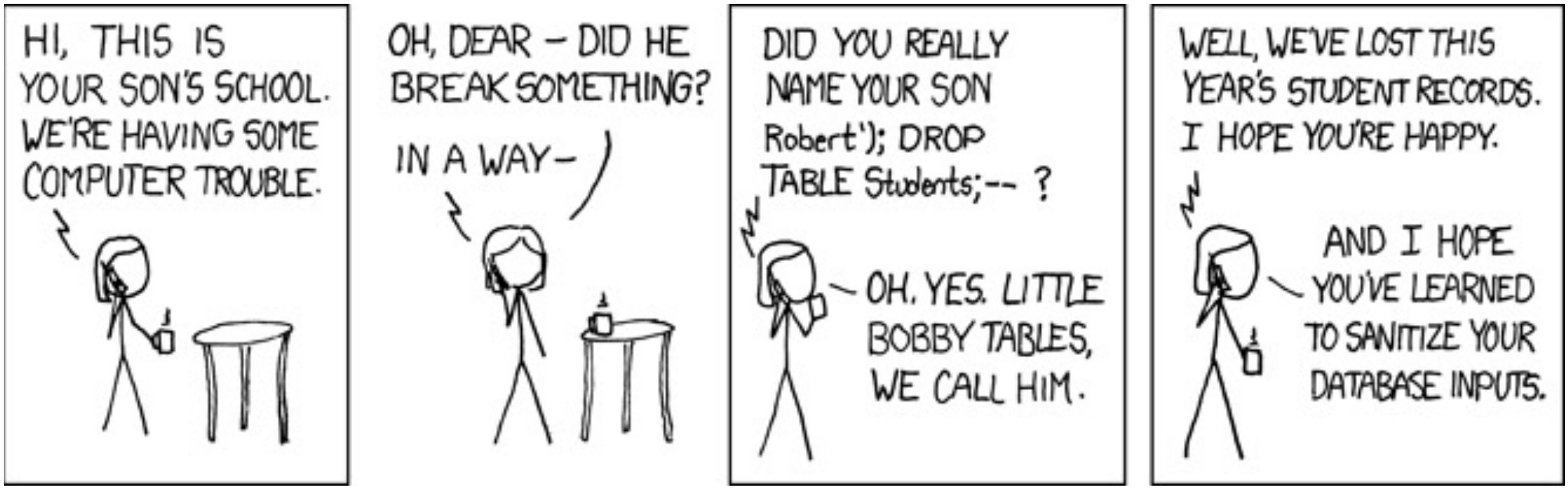
SQL Injection

http://w4111.github.io/inject

code on github:
`w4111/w4111.github.io/src/injection/`

Use query parameters by passing form values as arguments to `execute()`
Library sanitizes inputs automatically (and correctly!)

SQL Injection



Project: You'll need to protect against SQL injections

More examples: https://corenumb.wordpress.com/2016/05/14/mr-robot-blind-sql-injection-vulnerability/

Privacy vs Security

Privacy:

- person's right to control their personal information and how it's used
- Should not release data that can be used to *infer* user's personal information
- hard to define

Security:

- prevent unauthorized access to data
- access controls
- encryption prevents reading data even w/ access

Privacy or Security Breach?

BuzzFeed News

REPORTING TO YOU

SIGN IN

WORLD

The Location Data From Just Two Of Your Apps Is Enough To Identify You

A new report from researchers at Columbia University and Google has found that geotagged posts on just two social media apps are enough to draw a line back to a specific user.

Sheera Frenkel
BuzzFeed News World Correspondent

Posted on April 13, 2016 at 9:10 pm

https://www.buzzfeednews.com/article/sheerafrenkel/the-location-data-from-just-two-of-your-apps-is-enough-to-id#.ni3MMrkVa

Privacy or Security Breach?

The New York Times

Equifax Says Cyberattack May Have Affected 143 Million in the U.S.

Give this article 1K

By Tara Siegel Bernard, Tiffany Hsu, Nicole Perlroth and Ron Lieber

Sept. 7, 2017

Equifax, one of the three major consumer credit reporting agencies, said on Thursday that hackers had gained access to company data that potentially compromised sensitive information for 143 million American consumers, including Social Security numbers and driver's license numbers.

https://www.nytimes.com/2017/09/07/business/equifax-cyberattack.html

Privacy or Security Breach?

https://www.wsj.com/articles/facebook-scandal-what-to-know-about-cambridge-analytica-and-your-data-1521806400

Security Mechanisms

Access Control and GRANT

Different user accounts in w4111 DB

- staff, student, ew2493

Each user only has access privileges to read/modify subset of DB:

- Privileges: read, insert, update, delete
- Objects: Databases, Schemas, Tables, Views, Attributes
- Who? users, roles

https://www.postgresql.org/docs/current/sql-grant.html

Access Control and GRANT

```
GRANT <privileges>
    ON <objects>
    TO <users/roles>
    [ WITH GRANT OPTION ]
```

```
CREATE USER ew2493;
CREATE ROLE admin;

GRANT SELECT, INSERT ON users TO admin;     // users relation
GRANT CREATE, CONNECT ON DATABASE test TO admin;

GRANT admin to ew2493;
```

Access Control and REVOKE

```
REVOKE  <privileges>
   ON   <objects>
   FROM <users/roles>
   [ CASCADE ]
```

```
CREATE USER ew2493;
CREATE ROLE admin;

REVOKE SELECT, INSERT ON users FROM admin;  // users relation
REVOKE CREATE, CONNECT ON DATABASE test FROM admin;

REVOKE admin FROM ew2493;
```

Access Control and GRANT

How to restrict user's access to subsets of a relation or only aggregated statistics?

Combine GRANT and Views!

```
CREATE VIEW stats AS
    SELECT department, avg(salary) as avg_sal
      FROM costs
  GROUP BY department

GRANT SELECT ON stats TO appuser
```

Views and Updates?

Granting privileges over views is great for reading data. Does it support updates, inserts?

```
CREATE VIEW stats AS
    SELECT department, avg(salary) as avg_sal
      FROM costs
  GROUP BY department
```

Would this work?

```
UPDATE stats SET avg_sal = 500
```

Views and Updates?

Granting privileges over views is great for reading data. Does it support updates, inserts?

```
CREATE VIEW cs_professor_salaries AS
    SELECT name, salary, department
      FROM costs NATURAL JOIN professors
    WHERE department = “CS”
```

We could make this work, but is tricky

```
UPDATE cs_professor_salaries SET salary = 900000
WHERE name = “Eugene Wu”
```

Row Level Access Control

Granting row level security privileges allows fine grained control for update/insert queries

```
CREATE TABLE accounts (manager text, company
text, contact_email text);


ALTER TABLE accounts ENABLE ROW LEVEL SECURITY;


CREATE POLICY account_managers
ON accounts TO managers
USING (manager = current_user);
```

Hashing and Encryption

Mistakes happen and hackers may access our database. Is there something we can do?

Data should be stored in a way such that

- It can be used for our application needs
- While being unintelligible to the attackers

Two approaches : Hashing and Encryption

Encryption

Encryption: 2 way function, is reversible

- only users with key can read
- key needs to be kept safe!

```
Data → encrypt(key) → encdata → decrypt(key) → Data
```

Hackers get our key => game over. Can we do something about it?

https://www.postgresql.org/docs/current/encryption-options.html

Hashing

Hashing: store a hash instead of original data

- Hard to solve hash(x) = y for a given y
  Commonly referred to as one-way function

- Despite losing data, we can support equality checks

```
Data → hash() → hasheddata
hash(password) == hash(input)
```

- Collisions technically exist, but very unlikely

https://www.postgresql.org/docs/current/encryption-options.html

Hashing vs Encryption

Encryption

- Attackers work hard to break unless they have the key
- Can support arbitrary queries

Hashing

- Attackers always work hard to break
- Limited to equality checks

Hashing and Encryption

DBMS support varies

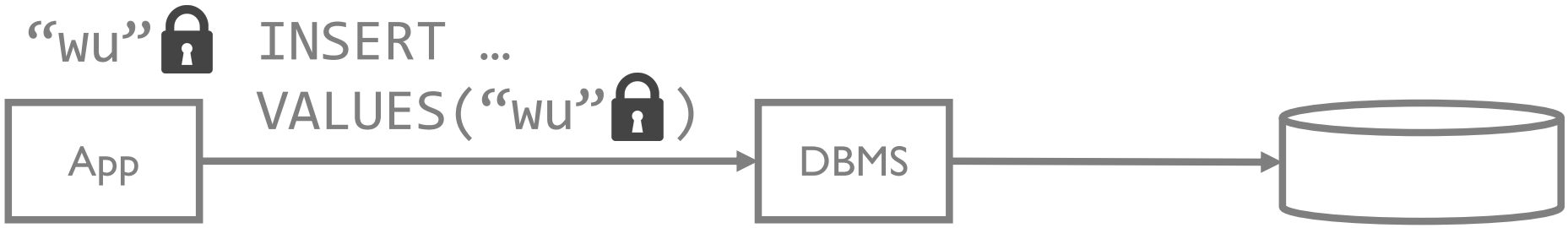
Hashing implemented as UDFs

- `INSERT INTO users VALUES(name, hash(password))`

Encryption

- encrypted hard drive, encrypted disk blocks, table, columns, …

Application encrypts `data` before issuing queries



Hashing and Encryption

DBMS support varies

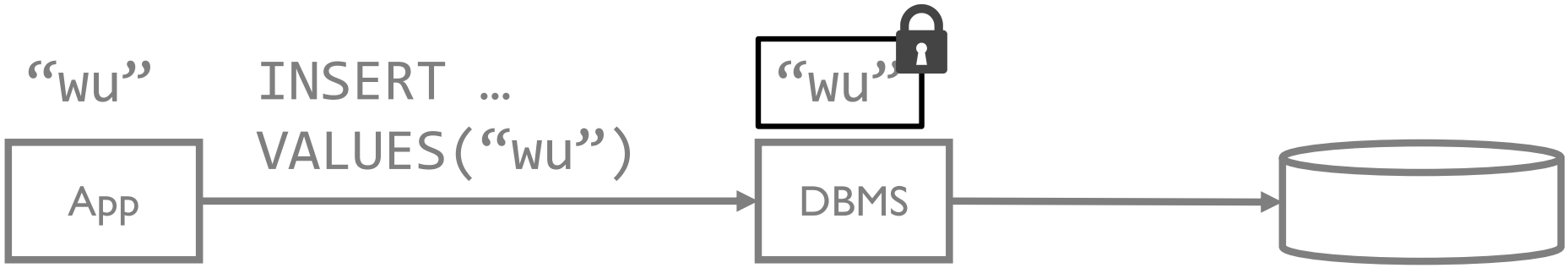
Hashing implemented as UDFs

- `INSERT INTO users VALUES(name, hash(password))`

Encryption

- encrypted hard drive, encrypted disk blocks, table, columns, …

DBMS encrypts received data (granularity of cells, rows, tables, or DB)



Hashing and Encryption

DBMS support varies

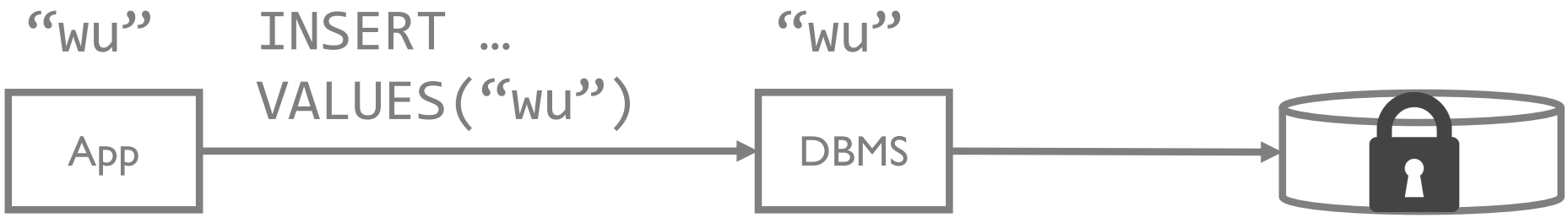
Hashing implemented as UDFs

- `INSERT INTO users VALUES(name, hash(password))`

Encryption

- encrypted hard drive, encrypted disk blocks, table, columns, …

Storage encrypts everything on e.g., hard drive



Hashing and Encryption

DBMS support varies

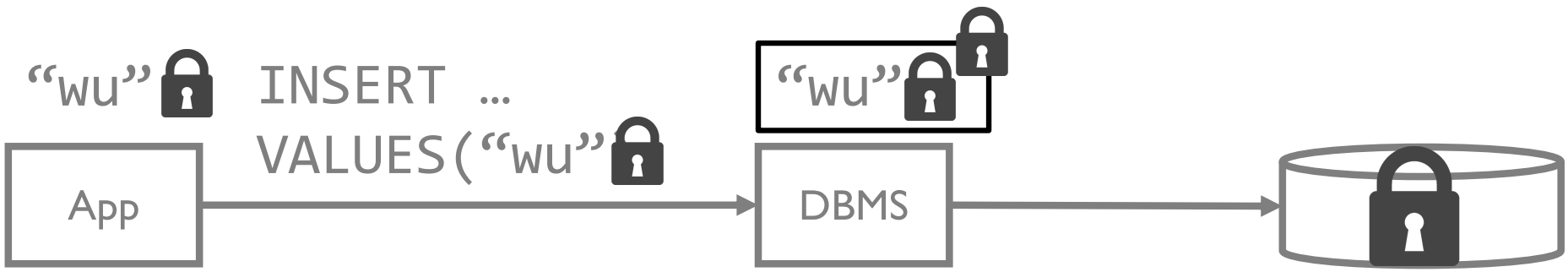
Hashing implemented as UDFs

- `INSERT INTO users VALUES(name, hash(password))`

Encryption

- encrypted hard drive, encrypted disk blocks, table, columns, …

Can mix and match



What to Understand

SQL injection and protections
Can identify privacy and security issues given a scenario
Can use access controls and views given a scenario
Properties and utility of hashing and encryption