Data Mining (DM)

- Data mining (DM) uses statistics, machine learning, mathematical models, and other techniques to discover patterns and relations on large datasets automatically
  - Tools: association, sequences, classification, clustering and prediction
  - Can support many application domains, including security (detecting cyber-attacks)
- The size and value of the datasets lead to high security & privacy risks
  - Private personal data
  - Confidential intellectual property

Challenges: Correctness and Integrity

- Error in one data element can impact many results of DM
  - DM has no backlink from result to source
  - E.g., what if your name is wrongly linked to a terrorist and processed by DM?
- Inconsistent data semantics across sources
  - E.g., one DB has income by dollar and another has income in euros
- False positives and false negatives draw error conclusions

Challenges: Privacy

- Big data enables new data collecting and correlating capabilities, causing new privacy issues
  - E.g., automatically inferring your preferences and showing relevant ads
- How can we protect our privacy? How can we contain data collectors' unlimited capabilities?
  - Data anonymization
  - Privacy-preserving analytics
  - More to cover in "Privacy" lectures

SQL Injection (or SQLI)

- Background
- Examples of SQLI
- Prevention

SQL Injection Background

- Web server treats user supplied “data” as “code”
- Execute the SQL query with malicious data (code)
- Compromise back-end database
- Comparison with XSS
  - XSS executes on client
  - SQLI executes on server
  - But the root-cause for both are the confusion between code and data

Hack that targeted Arizona voter database was easy to prevent, expert says

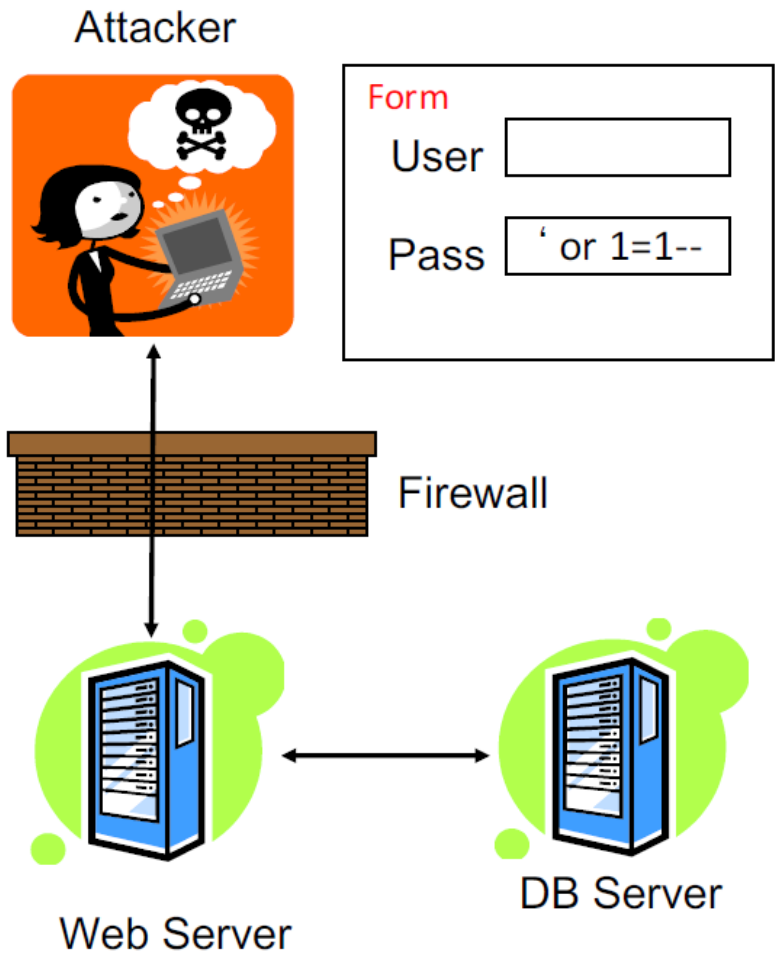
Published: Thursday, September 1st 2016, 1:58 am EDT
Updated: Thursday, September 1st 2016, 3:49 pm EDT

By Derek Staahl CONNECT

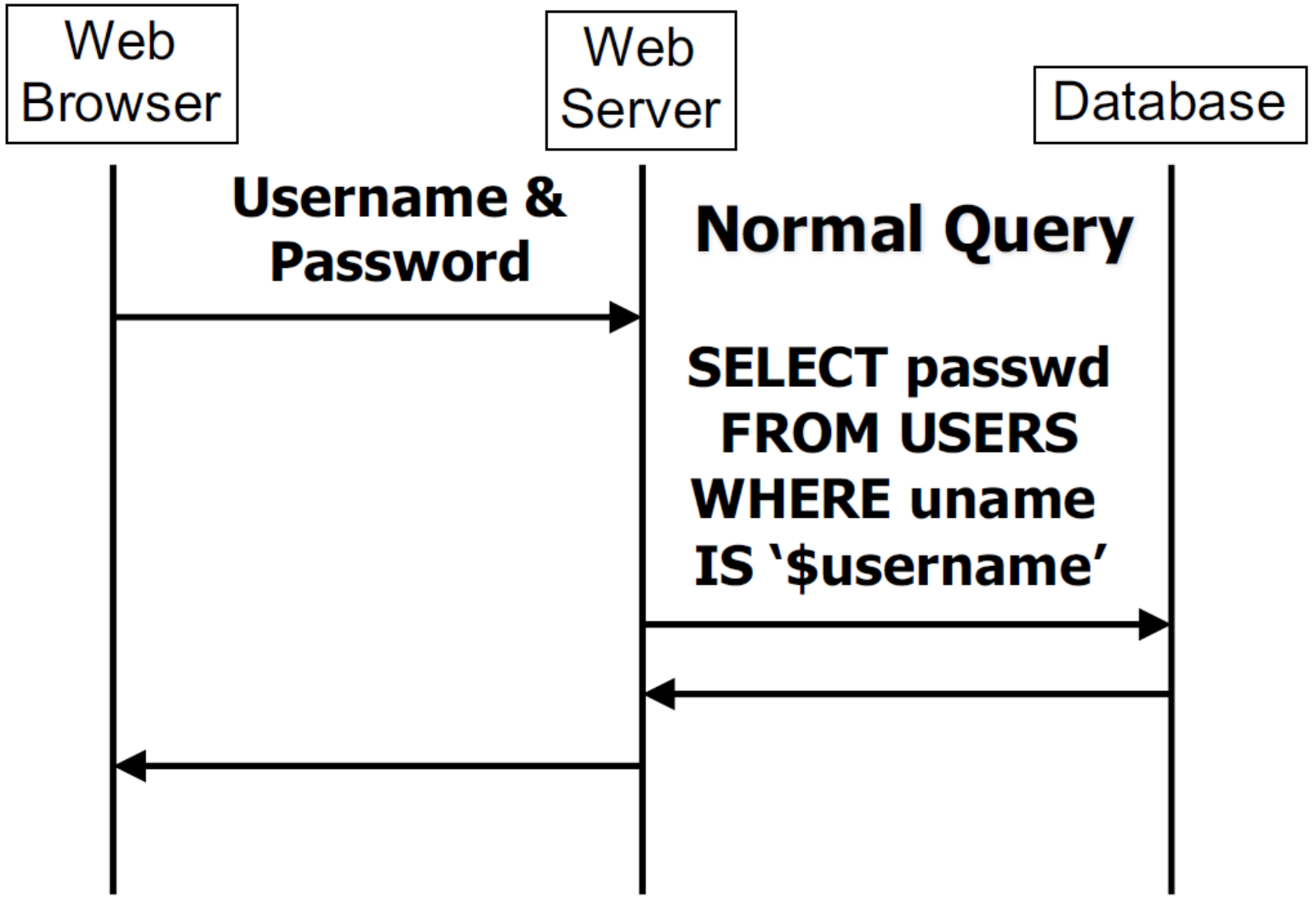
It’s a big real-world threat

Procedure of SQL Injection

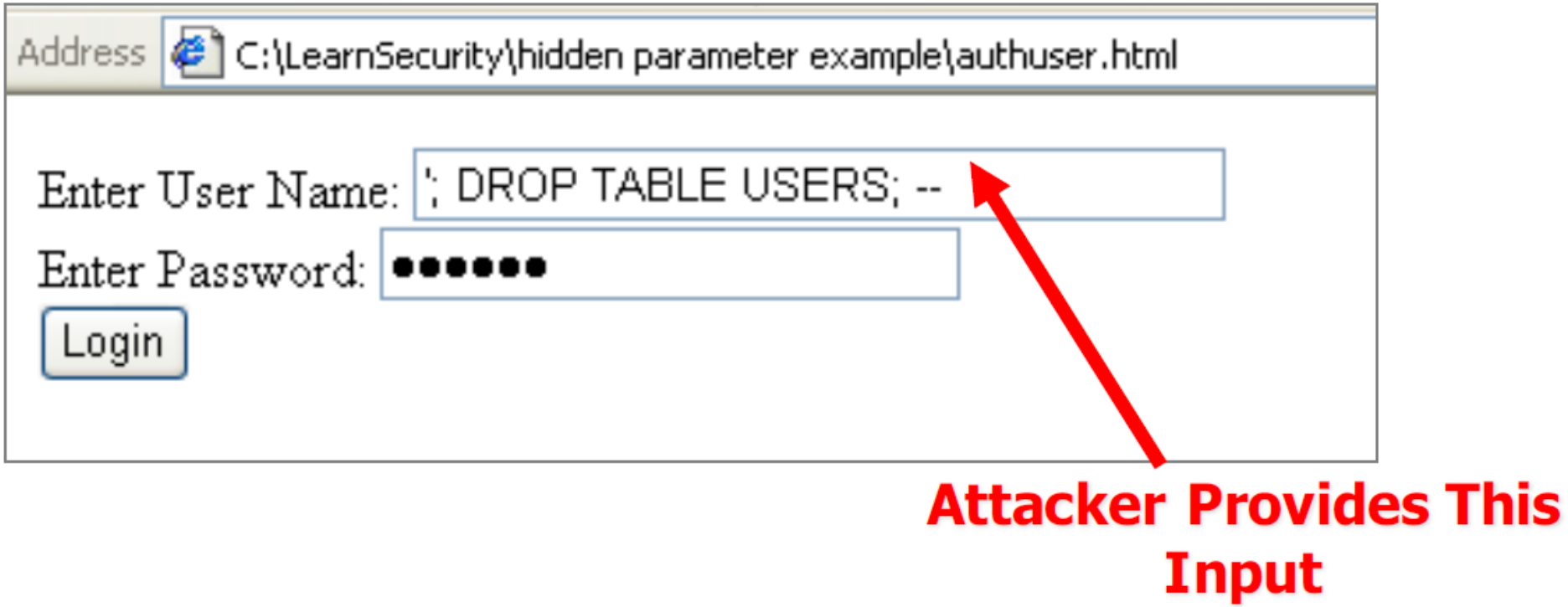
- 1. A website has a form, e.g., login
- 2. Attacker submits form with SQL exploit data
- 3. Server builds string with exploit data
- 4. Server sends SQL query to DB
- 5. DB executes query, including exploit, sends data back
- 6. Server returns data to user



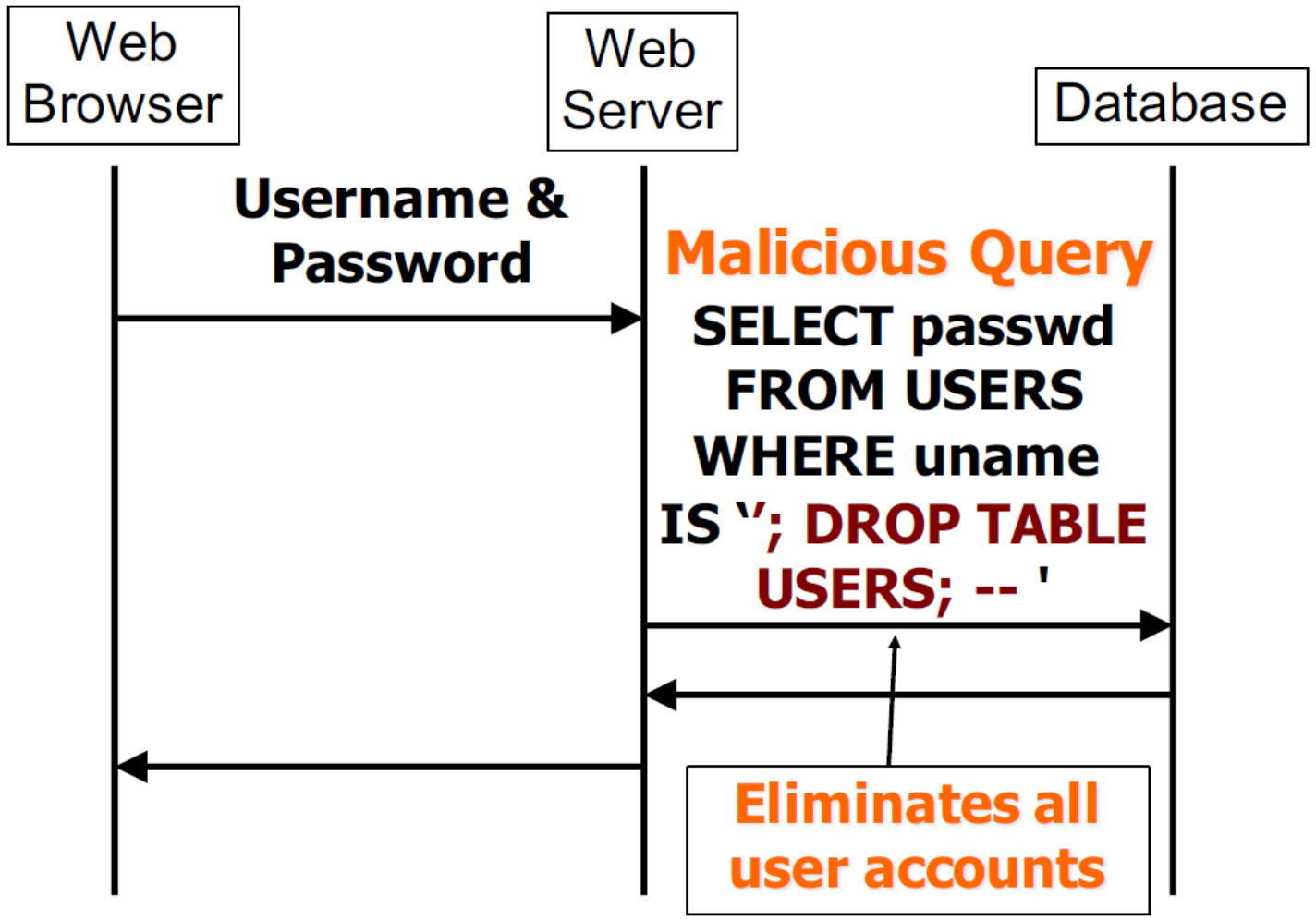
SQL Injection Example

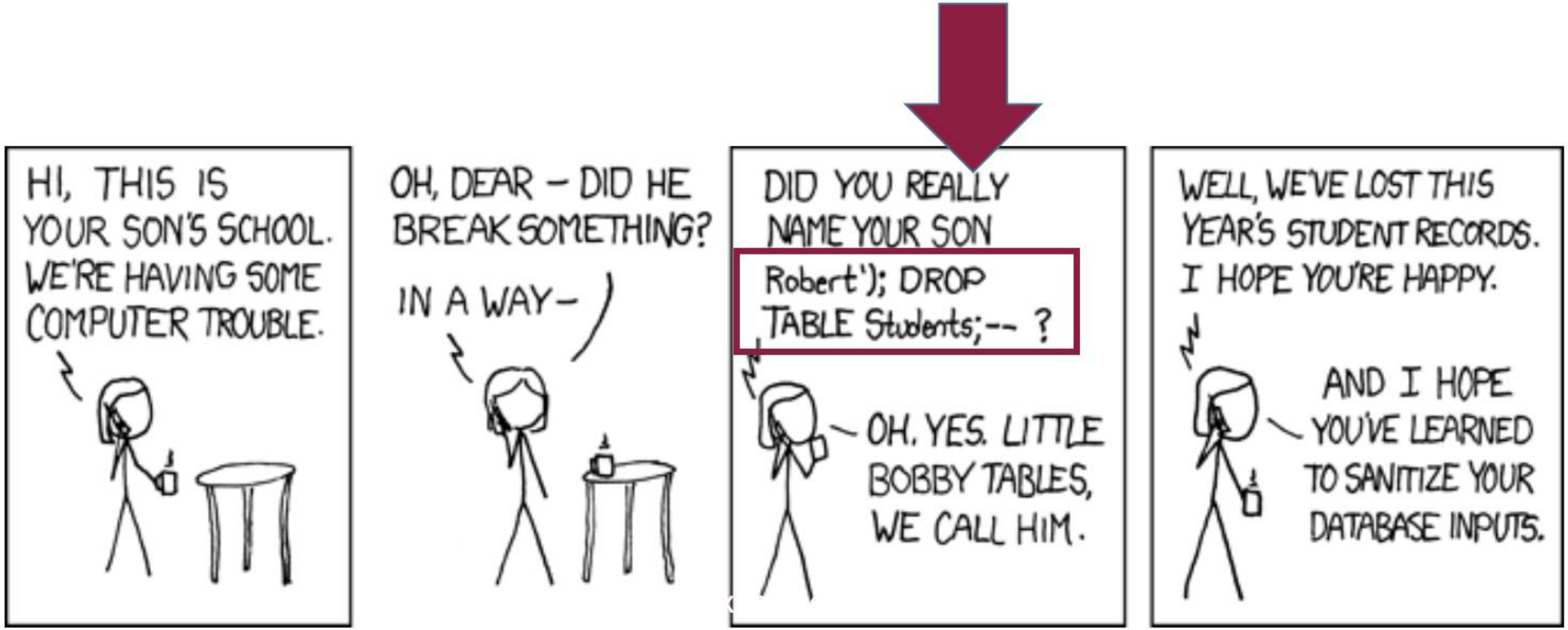


SQL Injection Example (cond.)

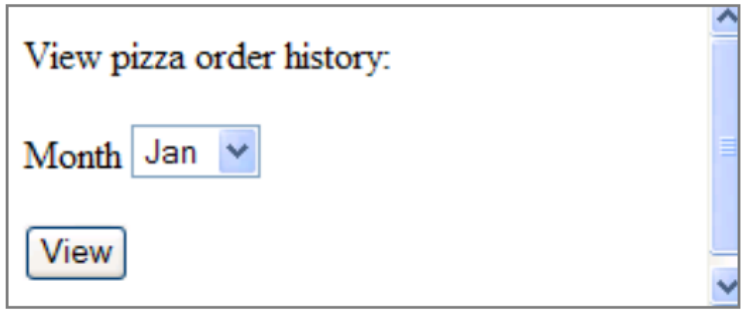


SQL Injection Example (cond.)




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.

SQL Injection Example 2




```
View pizza order history:<br>
<form method="post" action="...">
Month
<select>
<option name="month" value="1">
Jan</option>
...
<option name="month" value="12">
Dec</option>
</select>
<p>
<input type=submit name=submit
                    value=View>
</form>
```

SQL Injection Example 2 (cond.)

Normal SQL Query

```
SELECT pizza, toppings, quantity,
       order_day
FROM orders
WHERE userid=4123
AND order_month=10
```

Type 2 Attack

For `order_month` parameter, attacker could input

```
<option name="month" value="0 OR 1=1">
Dec</option>
```

Malicious Query

```
WHERE userid=4123
AND order_month=0  OR 1=1
```

SQL Injection Example 2 (cond.)

All User Data Compromised

Your Pizza Orders:

Pizza	Toppings	Quantity	Order Day
Diavola	Tomato, Mozarella, Pepperoni, ...	2	12
Napoli	Tomato, Mozarella, Anchovies, ...	1	17
Margherita	Tomato, Mozarella, Chicken, ...	3	5
Marinara	Oregano, Anchovies, Garlic, ...	1	24
Capricciosa	Mushrooms, Artichokes, Olives, ...	2	15
Veronese	Mushrooms, Prosciutto, Peas, ...	1	21
Godfather	Corleone Chicken, Mozarella, ...	5	13

SQL Injection Example 2 (cond.)

- A more damaging breach of user privacy:

```
0 AND 1=0
UNION SELECT cardholder, number,
             exp_month, exp_year
      FROM   creditcards
```

- Attacker is able to
  - Combine the results of two queries
  - Empty table from first query with the sensitive credit card info of all users from second query

SQL Injection Example 2 (cond.)

Credit Card Info Compromised

Pizza	Toppings	Quantity	Order Day
Neil Daswani	1234 1234 9999 1111	11	2007
Christoph Kern	1234 4321 3333 2222	4	2008
Anita Kesavan	2354 7777 1111 1234	3	2007
...			

Different Types of SQL Injections

- SQL injection can modify any type of query
- **SELECT statements**
  - SELECT * FROM accounts WHERE user='${u}' AND pass='${p}'
- **INSERT statements**
  - INSERT INTO accounts (user, pass) VALUES ('${u}', '${p}')
  - Note that in this case one has to figure out how many values to insert
- **UPDATE statements**
  - UPDATE accounts SET pass='${np}' WHERE user= '${u}' AND pass='${p}'
- **DELETE statements**
  - DELETE * FROM accounts WHERE user='${u}'

Determining Number/Types of Parameters

- Determine the number of columns in a query
  - Send progressively longer NULL columns
  - Until the correct query is returned
    - UNION SELECT NULL
    - UNION SELECT NULL, NULL
    - UNION SELECT NULL, NULL, NULL
- Determine type of columns
  - E.g., to determine if a column that has a string type
    - UNION SELECT ‘foo’, NULL, NULL
    - UNION SELECT NULL, ‘foo’, NULL
    - UNION SELECT NULL, NULL, ‘foo’

Preventing SQL Injection

- Whitelisting
  - Why? Blacklisting chars doesn't work:
    - Forget to filter out some characters
    - Could prevent valid input (e.g. username O'Brien)
  - Allow well-defined set of safe values: [A-Za-z0-9]* [0-1][0-9]
  - Valid input set defined through regular expressions
  - Can be implemented in a web application firewall
- Escaping
  - For valid string inputs like username o'connor, use escape characters. Ex: escape(o'connor) = o''connor (only works for string inputs)

Preventing SQL Injection (Cond.)

- Developers must never allow client-supplied data to modify SQL statements

- Stored procedures
  - Isolate applications from SQL
  - All SQL statements required by the application are stored procedures on the database server

- Prepared statements
  - Statements are compiled into SQL statements before user input is added

SQL Injection – Prevention

- Prepared statements
  - Specify structure of query then provide arguments
- Prepared statements – example

```
$stmt = $db->prepare("select * from `users` where `username` =
:name and `password` = SHA1( CONCAT(:pass, `salt`)) limit 1;");
$stmt->bindParam(':name', $name);
$stmt->bindParam(':pass', $pass);
```

- Sanitize inputs
- Limit the output of debugging information, which can be exploited to learn DB schema

Summary

- Database security requirements include:
  - Physical integrity, Logical integrity, Element integrity, Auditability, Access control, User authentication, Availability
- There are many subtle ways for sensitive data to be inadvertently disclosed, and there is no single answer for prevention
- Data mining and big data have numerous open security and privacy challenges
- SQL injection breaks DB integrity and the consequences can propagate to all users

Slides credit

- Security in computing 5th edition, Textbook Slides
- SQL Injection, Gang Wang