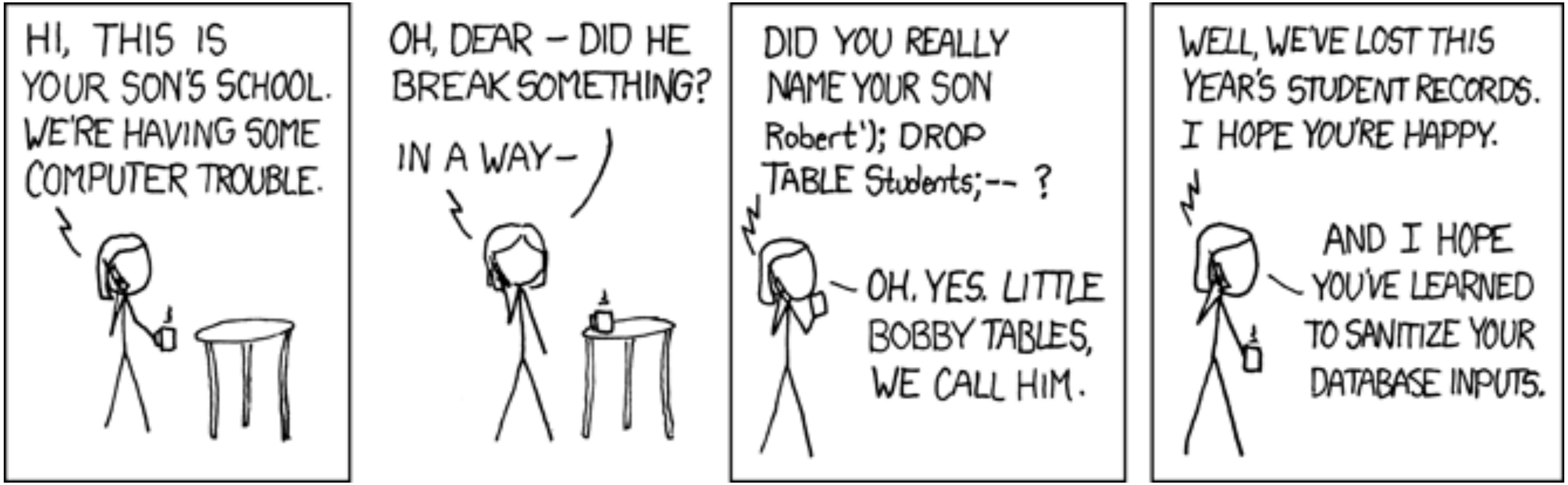
Week 2 – Advanced SQLi

Intro to Offensive Security

Recap

- SQLi: injecting our own query into the running query to modify the results


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.

Exfiltrating Data

- First, let's orient ourselves
  - Where are we? What are the table schemas?
  - `DATABASE()`
- What databases/tables/columns can we access?
  - "magic" information_schema database
  - information_schema.SCHEMATA
    - `SELECT SCHEMA_NAME`
  - information_schema.TABLES
    - `SELECT TABLE_NAME WHERE TABLE_SCHEMA = '…'`
  - information_schema.COLUMNS
    - `SELECT COLUMN_NAME WHERE TABLE_SCHEMA = '…' AND TABLE_NAME = '…'`

Exfiltrating Data

- Odds are good there are >1 DB, table, column
- How can we iterate over them?
- `LIMIT 1 OFFSET n`

- `SELECT TABLE_NAME WHERE TABLE_SCHEMA = '…' LIMIT 1 OFFSET 0`
- `SELECT TABLE_NAME WHERE TABLE_SCHEMA = '…' LIMIT 1 OFFSET 1`
- …

Exfiltrating Data

- Optimization: concatenate into 1 string
- `GROUP_CONCAT(TABLE_NAME SEPARATOR ',')`
- Returns a string with all names concatenated with `','`

Demo

Blind SQLi

- Blind = we don't get any data back
  - No immediate errors/return data, no UPDATE/INSERT injection, etc.
- But we do get some metadata back…
  - Timing
  - Error code
- How can we test for injection?
  - SLEEP()
  - Return bad data causing a 500

Time-based Blind SQLi

- Brute-force character by character
- IF(expr, val_if_true, val_if_false)
  - Evaluate expr, and return the 2nd arg if true, 3rd if false
- SUBSTR(str, start, len)

- SELECT IF(SUBSTR(name, 1, 1) = 'A', SLEEP(1), 0);

Demo

Time-based Blind SQLi Cont'd

- Optimization: binary search on each character we want to extract
- ASCII(char)
  - Get the equivalent ASCII character code for char
  - Basically the same as Python's ord()

- IF(ASCII(SUBSTR(name, 1, 1)) < 0x40, SLEEP(1), 0)

Second-order SQLi

- "First layer" properly escapes or parameterizes
- "Second layer" gets that data, then doesn't escape/parameterize
- Example scenario: online shopping
  - First layer: ordering
    - `INSERT INTO orders …`
  - Second layer: nightly batch processing
    - `SELECT address FROM orders...`
    - `INSERT INTO shipping_labels VALUES ('$address', …)`