14. Attacking the Web II (plus more on how it works)

Blase Ur and David Cash
February 10th, 2021
CMSC 23200 / 33250
Very Basic MySQL

• Goal: Manage a database on the server
• Create a database:
  – CREATE DATABASE cs232;
• Delete a database:
  – DROP DATABASE cs232;
• Use a database (subsequent commands apply to this database):
  – USE cs232;
Very Basic MySQL

• Create a table:
  - `CREATE TABLE potluck (id INT NOT NULL PRIMARY KEY AUTO_INCREMENT, name VARCHAR(20), food VARCHAR(30), confirmed CHAR(1), signup_date DATE);`

• See your tables:
  - `SHOW TABLES;`

• See detail about your table:
  - `DESCRIBE cs232;`
Some text here...
SQL Injection

• Goal: Change or exfiltrate info from victim.com’s database
• Main idea: Inject code through the parts of a query that you define
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.
SQL Injection

• Prerequisites:
  – Victim site uses a database
  – Some user-provided input is used as part of a database query
  – DB-specific characters aren’t (completely) stripped
SQL Injection: How?

• Enter DB logic as part of query you impact
• Back-end query
  – SELECT * FROM USERS WHERE USER=''
    AND PASS=''
• For username & password, attacker gives:
  – ' or '1'='1
• Straightforward insertion:
  – SELECT * FROM USERS WHERE USER=''
    or '1'='1' AND PASS=''
    or '1'='1'
SQL Injection: Why Does This Work?

- Database does what you ask in queries!
SQL Injection: Key Mitigations

• Sanitize / escape user input
  – Harder than you think!
  – Different encodings
  – Use libraries to do this!

• **Prepared statements** from libraries handle escaping for you!

• Use PHP’s mysqli (in place of mysql) with prepared statements
  – [https://www.w3schools.com/php/php_mysql_prepared_statements.asp](https://www.w3schools.com/php/php_mysql_prepared_statements.asp)
Sending Data to a Server

• GET request
  – Data at end of URL (following “?”)

• POST request
  – Typically used with forms
  – Data *not* in URL, but rather (in slightly encoded form) in the HTTP request body

• PUT request
  – Store an entity at a location
URL Parameters / Query String

- End of URL (GET request)
  - https://www.cs.uchicago.edu/?test=foo&test2=bar
Processing Data on the Server

• Javascript is client-side
• Server-side you find Perl (CGI), PHP, Python (Django)
• Process data on the server
• What happens if this code crashes?
Storing Data on the Server

• Run a database on the server
• MySQL, SQLite, MongoDB, Redis, etc.
• You probably don’t want to allow access from anything other than localhost
• You definitely don’t want human-memorable passwords for these
Browser Extensions

• Can access most of what the browser can
• Requires permissions system
• Malicious extensions!
What If You Get Lots of Traffic?

- CDNs (content delivery networks)
What If You Don’t Want To Code?

• CMS (content management system)
  – WordPress (PHP + MySQL), Drupal