Introduction to Information Security

Web Vulnerabilities - Backend
About Passwords

- Remember long sentences > shorter randomness?
- The sentences shouldn't be coherent.
- Otherwise, Markov Chains can break them.
- Random Generator
- + Probability (Distribution)
- + Conditional Probability (Memory)

Through 20 years of effort, we've successfully trained everyone to use passwords that are hard for humans to remember, but easy for computers to guess.
Reminder

• A browser sends HTTP requests to a web server, which serves them and generates HTTP responses with HTML, CSS and JS content, which the browser then renders.

• Frontend Vulnerabilities: malicious responses (like cookie manipulations) to hack the client side

• Backend Vulnerabilities: malicious requests (like injections) to hack the server side
  • SQL Injection
SQL

- Databases with tables, with columns, with values

- `INSERT INTO` `users` (username, password) `VALUES` ('alice', 'foo')
- `UPDATE` `users` `SET` password = 'foobar' `WHERE` username = 'alice'
- `DELETE` `FROM` `users` `WHERE` username = 'alice'
- `SELECT` * `FROM` `users`
- `SELECT` username `FROM` `users` `WHERE` username = 'alice' `AND` password = 'foo'

- `SELECT` * `FROM` `users` `JOIN` `messages` ON (users.username = messages.user)
  - Rows often have a PK (primary key), and JOINs are often done on FKs (foreign keys)
Authentication

```python
def authenticate(request):
    user, pass = request.POST['username'], request.POST['password']
    rows = db.execute("SELECT * FROM users WHERE username = '%s' AND password = '%s'" % (user, pass))
    return len(rows) > 1
```

```html
<form action="/login" method="POST">
    <input type="text" name="username" />
    <input type="password" name="password" />
    <input type="submit" />
</form>
```
SQL Injection

SELECT * FROM users WHERE username = 'alice' AND password = '' OR 1 = 1 -- ' comment
More SQL Injections

- '; DROP TABLE
- UNION
Remote File Inclusion

• Essentially the same: user input is integrated into the execution and hijacks control
  • Except this time it usually involves loading and executing external / uploaded files
• In PHP, it was a mess
• Surprisingly, in Python (and Ruby, and Node.js) it's also possible
  • it all comes down to lazy programming
Python Packaging

• A python module is a file (with a .py extension) which can be imported
• A python package is a directory (with a __init__.py file) which can also be imported
  • The import result is actually the directory/__init__.py module
  • But it allows relative imports (from .sibling import x, from ..parent import y)
• Imagine this structure:
• And assume the server uses a GET parameter to device which hash
  To use: __import__(request.GET['hash']).compute()
• http://domain/resource?hash=md5
• http://domain/resource?hash=uploads
  • Of course, upload is not an importable module...
  • Or is it?
Missing Function Level Access Control

- Developers rely on the UI
  - But an attacker can easily forge requests to the "hidden" functionality!
- Examples:
  - Don't display the button
  - Please log in to continue
  - Credential validation!
Information Disclosure

• Don't leak personal information. Easy, right?

• Consider this:
  • username: foo, password: **** invalid username
  • username: dan, password: **** invalid password

• "Pipeline" structure is especially prone to information disclosure
  • Even in hardware – for example, safes