

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.



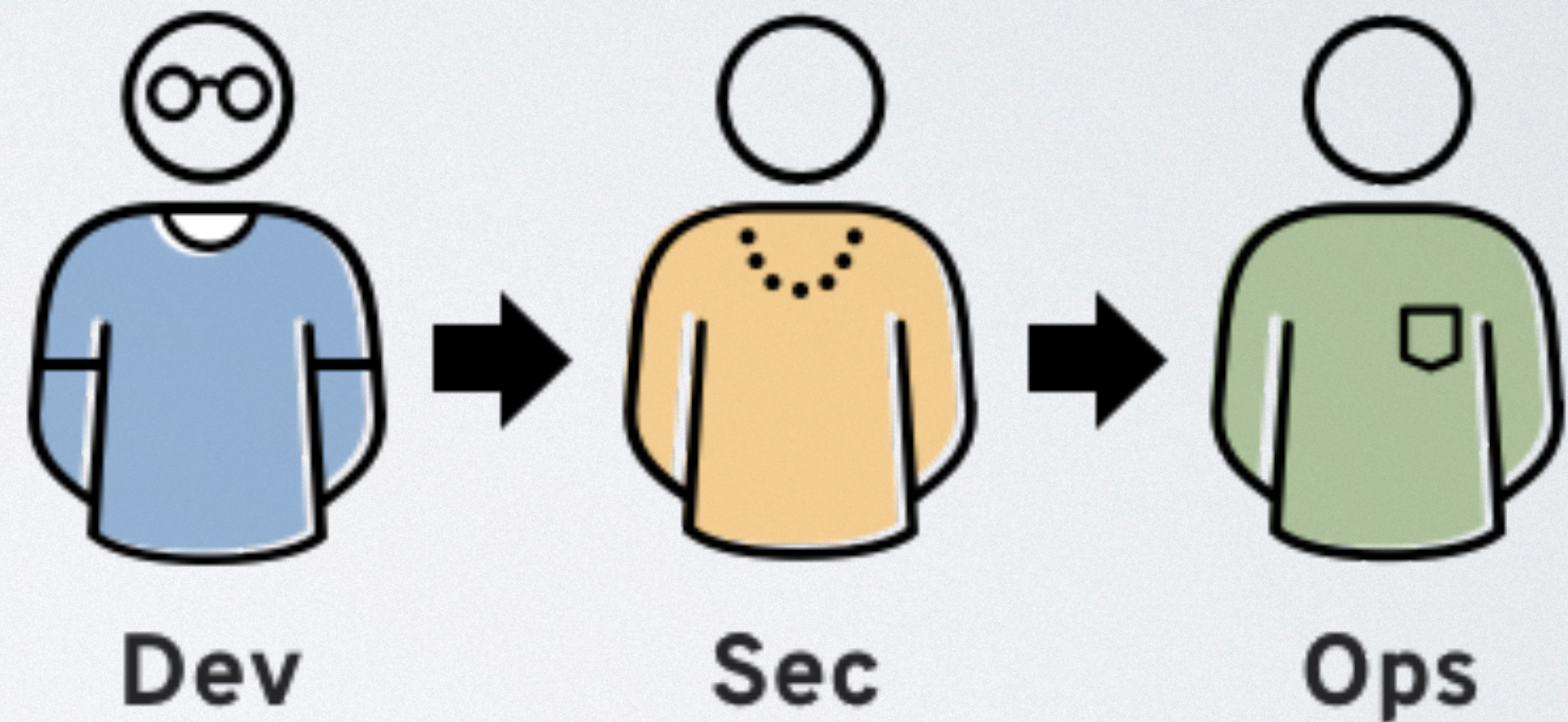
AND I HOPE
YOU'VE LEARNED
TO SANITIZE YOUR
DATABASE INPUTS.

Security and DevSecOps

Integrating Security into the Software Development Process

The Old Way

- First, write the code
- Then, have the security people do their thing
- Then, let the operations people host it
- But doing security too late is bad...



Security Has Architectural Implications

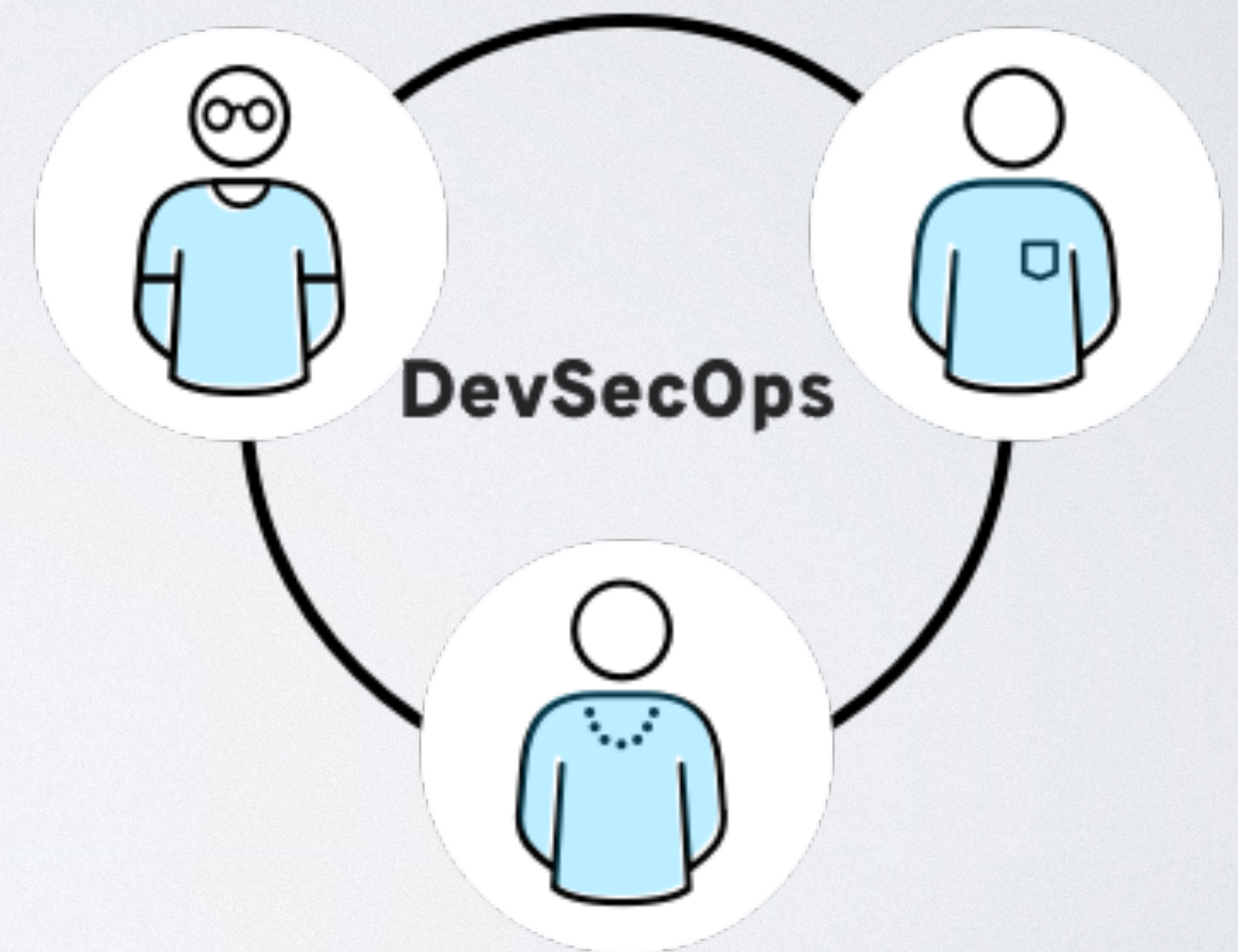
- Where is access control?
- Where is authentication?
- How are credentials passed?
- What are the attack vectors?

More Design Implications

- Tooling: you aren't going to use C/C++, are you?
- Testing processes
 - Penetration tests?
- How will you mitigate social engineering attacks?

DevSecOps

- Integrate security into the development process
- The rest of today: how to include security concerns



Kinds of Security Challenges

Challenge	Approach
Undefined behavior	Don't use unsafe languages (when possible)
Incorrect security-related code	Review, test, control changes
Higher-level design mistakes	Architectural review, penetration testing
Users (e.g., social engineering attacks)	HCI techniques; training; compromise procedures

Microsoft DevSecOps Advice

- Train
 - Perform threat modeling
- Define security requirements
 - Use tools and automation
- Define metrics and compliance reporting
 - Keep credentials safe
- Use Software Composition Analysis and Governance
 - Use continuous learning and monitoring

Train

- Glad you're here.

Define Security Requirements

- Legal and industry requirements
- Internal standards and coding practices
- Review of previous incidents, and known threats.
- Traditional requirements analysis, with security focus

Define Metrics and Compliance Reporting

- How will you know whether you've succeeded?
- Does one breach mean you've failed?
 - Better to focus on progress than success/failure

Threat Modeling

- Goal: enumerate all possible threats
- STRIDE model helps you remember possible threats:
 - **S**poofing identify
 - **T**ampering with data
 - **R**epudiation
 - **I**nformation disclosure
 - **D**enial of service
 - **E**levation of privilege

Exercise

- In groups: enumerate possible threats for your project
- In a real meeting: spend 2 hours, identify 20-40 issues.

Use Software Composition Analysis and Governance

- Vulnerabilities can come via third-party tools and components

Use Tools and Automation

- Tools must be integrated into the CI/CD pipeline.
- Tools must not require security expertise.
- Tools must avoid a high false-positive rate of reporting issues.
- Static analysis
- Dynamic analysis

Keep Credentials Safe

- Scan for keys in source code

Use Continuous Learning and Monitoring

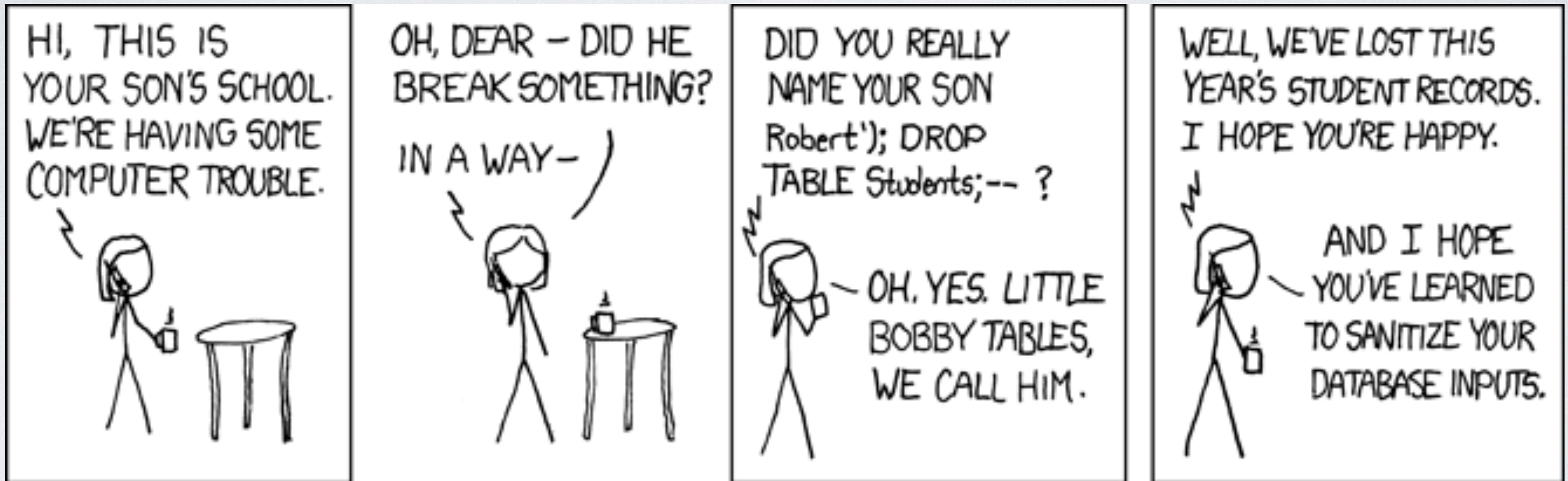
- Continuous integration / continuous delivery
 - Should run analyses automatically
- Mean time to identify (MTTI)
- Mean time to contain (MTTC)

Top 10 Threats (OWASP)

- Broken access control
- Cryptographic failures
- Injection
- Insecure design
- Security misconfiguration
- Vulnerable and outdated components
- Identification and authentication failures
- Software and data integrity failures
- Security logging and monitoring failures
- Server-side request forgery

Mitigating Key Threats

Threat 1: Untrusted Data



Avoiding Injection Attacks

- Validate input
- Avoid `eval()`
- Sanitize input when constructing SQL queries

Cross-Site Scripting (XSS) Attacks

1. Untrusted data enters web app

2. Data is included in content sent to a user (victim)

Example source: <https://owasp.org/www-community/attacks/xss/>

XSS Example

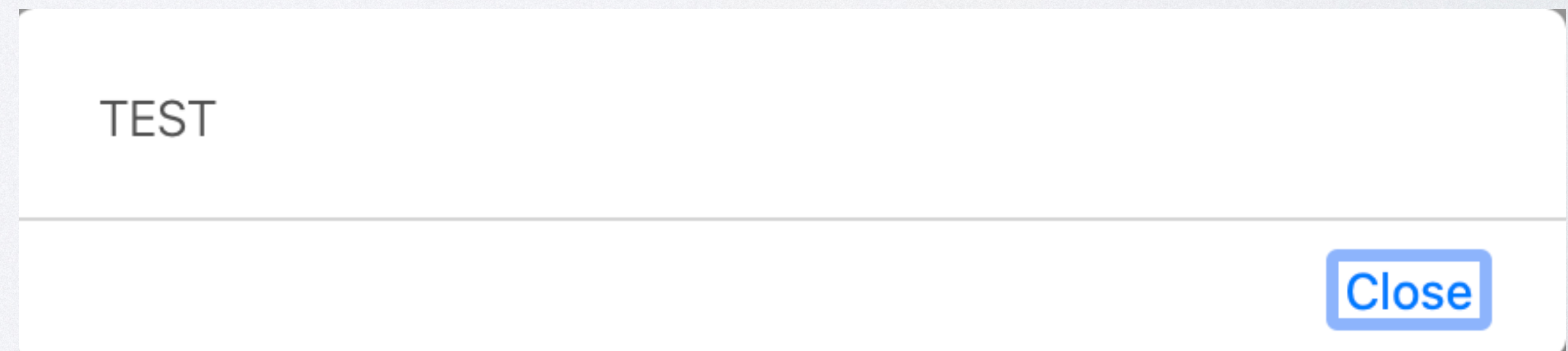
Vulnerable web page (PHP):

```
<html>
<body>
<?php
print "Not found: " . urldecode($_SERVER["REQUEST_URI"]);
?>

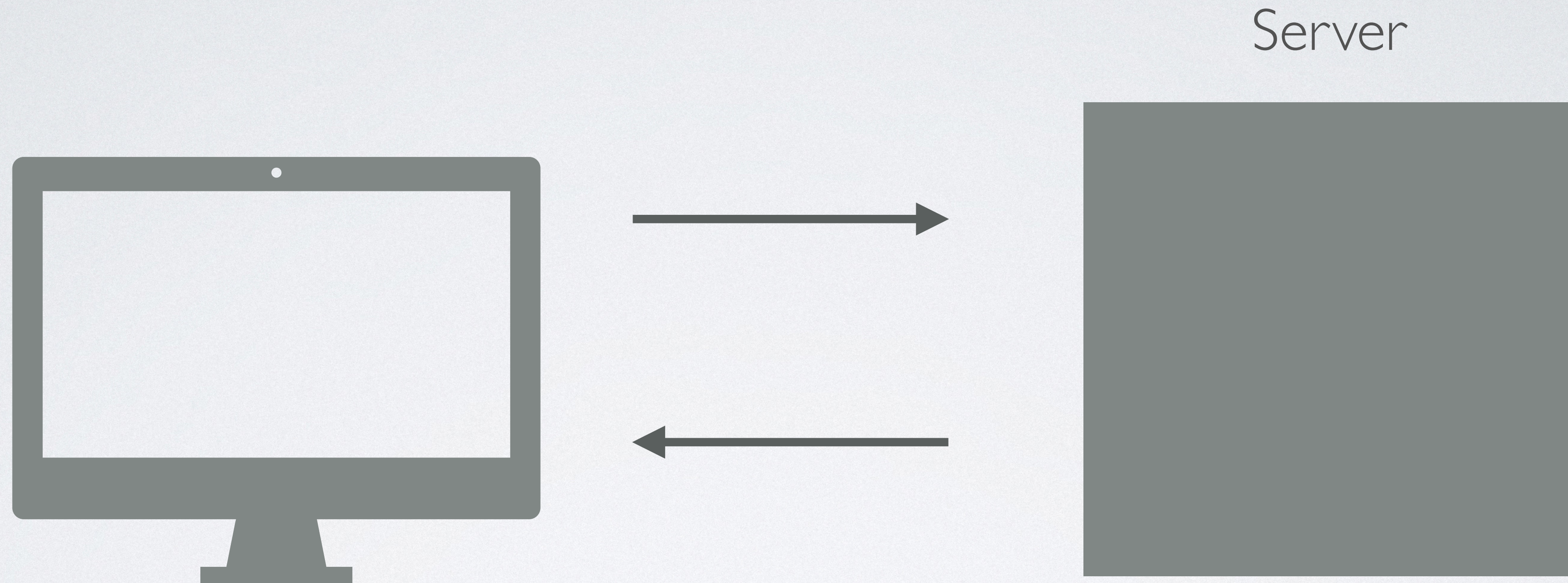
</body>
</html>
```

User clicks a link: `http://www.ucsd.edu/<script>alert("TEST");</script>`

User is surprised to see an alert:



Threat 2: Bad Authentication



"Am I talking to a legitimate server?"

Use TLS to check server's certificate

"Is the client who they say they are?"

Check user credentials

Authentication vs. Authorization

- Authentication: are you who you say you are?
- Authorization: Given who you are, what can you do?
 - Policies enforced with access control

Use letsencrypt.com for Free Certificates

- Without a certificate, your users can be victims of a man-in-the-middle attack

Password Cracking

- Brute force: try all strings
 - Mitigation: large space of passwords
 - Mitigation: avoid commonly-used passwords ("password")
- Rainbow table: pre-compute hashes of common passwords
 - Search hashes in stolen password table for known passwords
 - Mitigation: salts

Salts

- username: harry; password: ucsd4life
- username: bovik; password: ucsd4life
- $\text{sha256sum}(\text{"ucsd4life"}) =$
5a321b082a1e8c97f1af3314c374780d44bb7f8dce4107231660ba0a6b852d43
- Both users' passwords hash to the same value!
- An attacker who compromises harry's account and gets a copy of the database also gets access to bovik's account.

Salts

- Aside: both users picked bad passwords (too short)
- Solution: each user gets a random "salt"

username	salt	password
harry	y893r2e	sha256sum("harryy893r2e")
bovik	asdffdsjlkfs	sha256sum("bovikasdffdsjlkfs")

Passwords

- Passwords do not go in your repository!
 - Passwords go in config files (store these somewhere safe)
- Passwords do not go in your database!
 - Salted, hashed passwords go in your database

Principle of Least Privilege

- Only authorize access that is actually needed