CISC 327 - Software Quality Assurance

Lecture 29b
Web Application Security
Outline

• Web Application Security
  – SQL Injection
  – Parameter Manipulation
Security on the Web

• In an ideal world...
  – You develop a simple and robust web application
  – Users discover your web application and use it as you anticipate they would
  – The interface is clear enough to communicate proper usage of your application
  – There is no need to defend against malicious attacks against your application
Security on the Web

• In the real world...
  – Most users are wonderful
    • Even inexperienced users don't get too far out of line
  – Users often use applications in ways that you never could have anticipated
    • "That's how you use it?"
    • A significant amount of Google queries should have been entered in the Location bar
  – Malicious users like to hack
Security on the Web

• Don't trust user input
  – Users will not always submit data that your application will expect
  – As a general principle, do not trust user input by default
    • URL parameters, form data, cookies, etc.
  – Problems will often be unintentional
    • Non-sanitized quotes, hyphens, or non-ASCII characters (see the “Piece of Crap” lecture)
SQL Injection

• Parameter injection to exploit vulnerabilities
  – SQL statements (or parts of one) are injected into a web form or URL string
  – Attacks software that does not properly filter user input
  – Arbitrary SQL commands can be authored by an attacker to dump database information or change database content
SQL Injection

• PHP Example

```php
$query = "SELECT * FROM `users` WHERE name = '' + $user_name + '';";
```

• Good case: `$user_name == "Scott Grant"`

```sql
SELECT * FROM `users` WHERE name = 'Scott Grant';
```

• Malicious case: `$user_name == '' OR '1'=1`

```sql
SELECT * FROM `users` WHERE name = '' OR '1'=1;
```
**SQL Injection**

- Valid SQL statements can be constructed

```javascript
$query = "SELECT * FROM `users` WHERE name = '' + $user_name + '';"
```

- `$user_name == ''; DROP TABLE `users`; --"`
Preventing SQL Injection

• Ensure characters are escaped
  – The problem in the earlier query was caused by non-escaped quotes
    
    SELECT * FROM `users` WHERE name = ''' OR '1'='1';

    – If the input string contains characters that need to be quoted in SQL strings, we must ensure that those characters are actually quoted

    SELECT * FROM `users` WHERE name = '\'' OR '\1\'='\1';
Preventing SQL Injection

• Proper type checking
  – If a parameter is supposed to be a number, we must ensure that a number is used

```php
$quer\textcolor{red}{y} = "SE\textcolor{red}{L}ECT * FROM `users` WHERE id = " + \\
\textcolor{red}{$user\_id} + ";";
```

  – Malicious: $\textcolor{red}{user\_id} = "1; DROP TABLE `users`"

```sql
SELECT * FROM `users` WHERE id = 1;
DROP TABLE `users``
SQL Injection

• https://xkcd.com/327/
Parameter Manipulation

• Manual modification of parameters
  – Information is usually stored in cookies, hidden form fields, or URL query strings
  – If cost is a part of the parameters, it can be changed by a malicious user
...walks into a bar...

- [https://www.sempf.net/post/On-Testing1](https://www.sempf.net/post/On-Testing1)


  “You Used To Be Able To Order A Negative Quantity Of Books On Amazon And Get Paid Real Money”
Parameter Manipulation

- Manual modification of parameters

Hackers breached Citibank security using simple URL manipulation (June 15, 2011)

The theft of approximately 200,000 Citibank customer accounts may have achieved by means of a simple manipulation of the Citibank URL. Security experts told the New York Times that the hackers were able to impersonate actual account holders by using a simple trick.

After logging into a valid account, the URL to the Citi Account Online system contains a string of numbers which represents the customer's account. By changing this string, the criminals were able to easily switch between multiple accounts and obtain private customer information. Using a script to automate this process allowed them to do so hundreds of thousands of times.
Parameter Manipulation

• Don't trust the user!
  – If you have the cost of a book in a database, query the database instead of the parameter string
  – If you have a user logging in, track their account number using almost anything other than the parameters
  – In general, assume that a request is malicious, even if the majority of users are friendly
Parameter Manipulation

• Consider the effect of each parameter
  – If every parameter can be modified on the fly, what does this mean for your program?
  – Spending the time early in development to prevent unauthorized access will save time repairing issues later
  – Remember, form data and cookies are just other forms of user input, and they must be treated with caution
Summary

• SQL Injection attacks are a real threat
  – Compromise sensitive user data
  – Alter or damage critical data
  – Provide unwanted access to the database

• Validate and sanitize data early
  – Best to sanitize input at entry point into the code

• But respect your users!
  – Please don’t “sanitize” François into Francois or Franois, O’Brien into OBrien, or Håkan into Hkan
Next Week:

- Review lecture Tuesday
- Last Mini-Exam next Wednesday
  - I drop the lowest, so if you wrote the first three and you’re happy with your marks, you could skip it
- Assignment 6 due next Friday