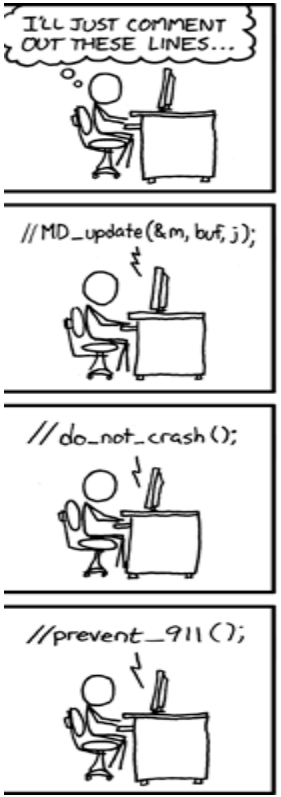
Turn on your video if possible.

CS1320 Creating Modern Web and Mobile Applications

Lecture 28

Security I

Security is a big problem



IN THE RUSH TO CLEAN UP THE DEBIAN-OPENSSL FIASCO, A NUMBER OF OTHER MAJOR SECURITY HOLES HAVE BEEN UNCOVERED:

AFFECTED SYSTEM	SECURITY PROBLEM
FEDORA CORE	VULNERABLE TO CERTAIN DECODER RINGS
XANDROS (EEE PC)	GIVES ROOT ACCESS IF ASKED IN STERN VOICE
GENTOO	VULNERABLE TO FLATTERY
OLPC OS	VULNERABLE TO JEFF GOLDBLUM'S POWERBOOK
SLACKWARE	GIVES ROOT ACCESS IF USER SAYS ELVISH WORD FOR "FRIEND"
UBUNTU	TURNS OUT DISTRO IS ACTUALLY JUST WINDOWS VISTA WITH A FEW CUSTOM THEMES

```
int getRandomNumber()
{
    return 4;  // chosen by fair dice roll.
               // guaranteed to be random.
}
```

Security & Privacy Problems

- Security Week
  - securityweek.com
- SC Magazine
  - scmagazine.com
- CNET on Security
  - www.cnet.com/topics/security
- ThreatPost on Web Security
  - https://threatpost.com/category/web-security

Security and Privacy

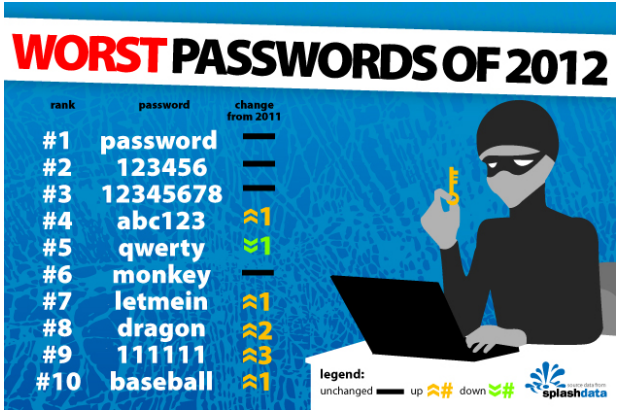
- Many web sites are developed initially without taking these into account
  - What are the consequences?
  - We hear about security/privacy issues daily
  - Most exploits use well-understood techniques
  - Most exploits could be avoided with a little care
- Optimizations
- Need to think about security and privacy
  - **From the start**
  - Design the web site with this in mind
  - Design the code with this in mind
  - Change the code with this in mind

Security is Fun

- Creative thinking – outside the box
- Think of all the ways of breaking software
  - Other peoples -- not yours
- Think of ways of preventing such breakage

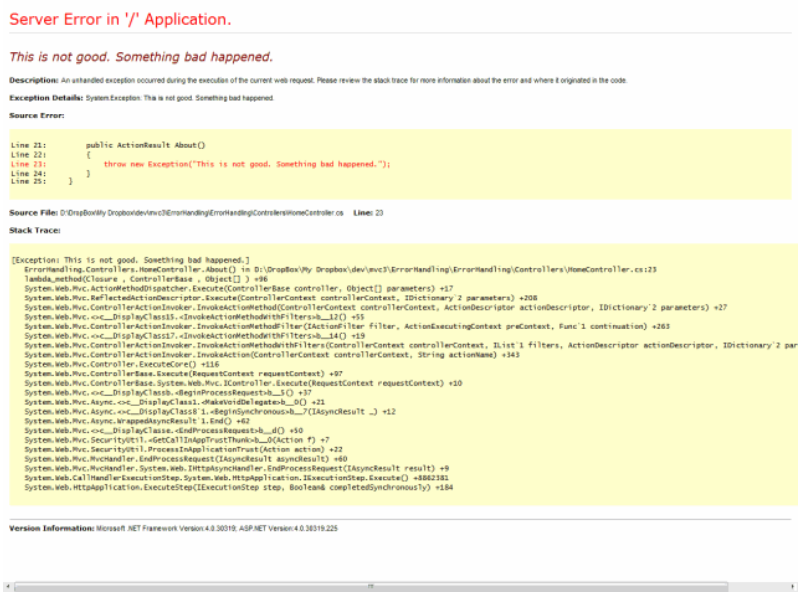
Obvious Problems

- Not considering security in your application
  - Not requiring user authentication
  - Allowing weak authentication
  - Not encrypting sensitive communication
  - Sessions that don't time out
  - Session ids that are guessable
- Not putting in the resources needed
  - Yahoo



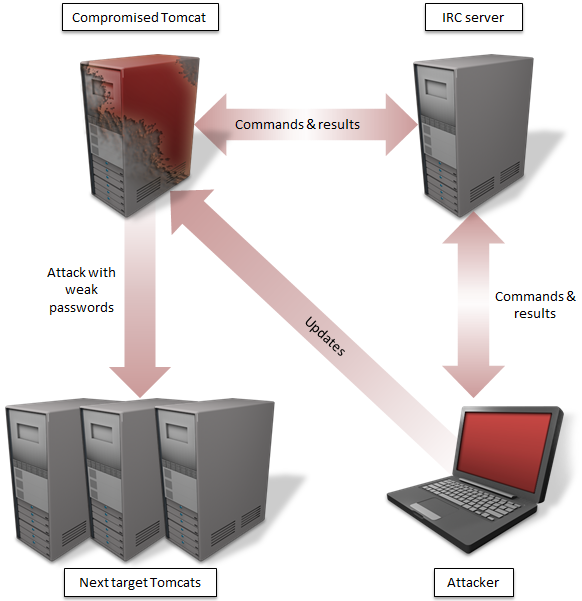
Obvious Problems

- Having a vulnerable server
  - Not being at the latest patch level
- Having bugs in the software
  - Disclosing information inadvertently
  - Exposing SQL and other errors
- Using vulnerable libraries
  - NPM and GitHub now warn you
  - But that is not enough
- Sending private information publically



Semi-Obvious Problems

- Applications sharing a common back end
  - PHP, Tomcat, … have common code
  - Applications can interfere with each other
- Phishing attacks
  - Gain access through authorized user
  - Gain access to authorized machine
- Loss of hardware
  - Laptops get stolen or lost

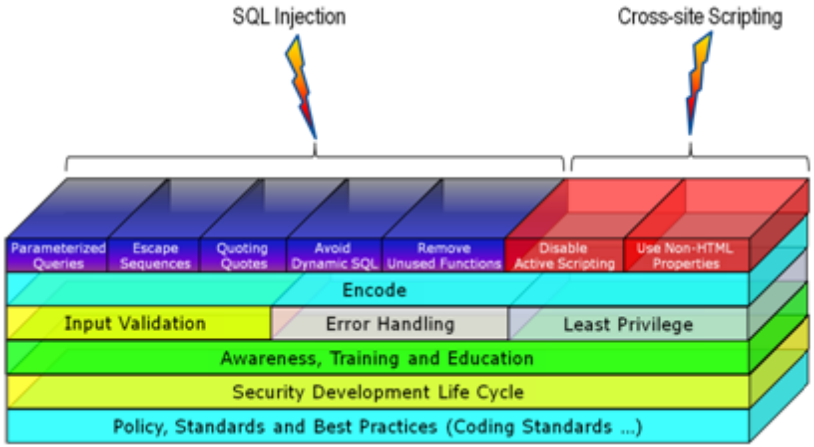


Non-Obvious Problems

- Assume we have secured our server and libraries, validated our code, used best practices for data encoding, etc.
  - Are there still things that can go wrong?
- No system is ever totally secure
  - You only hope is it more expensive to break
  - Than the value gained from breaking it

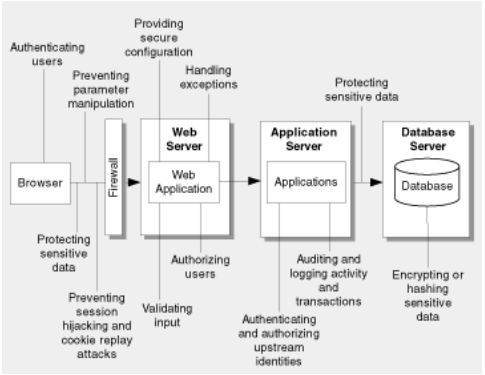
Web Security Issues

- Application security attacks
  - SQL injection attacks
  - Cross-site scripting attacks
  - Cross-site request forgery
  - Code insertion attacks
  - File name attacks
  - Buffer overflow attacks
  - Timing attacks …
- Keeping information secret
  - Passwords, Credit Cards, …
  - From whom and when?



Security and Web Applications

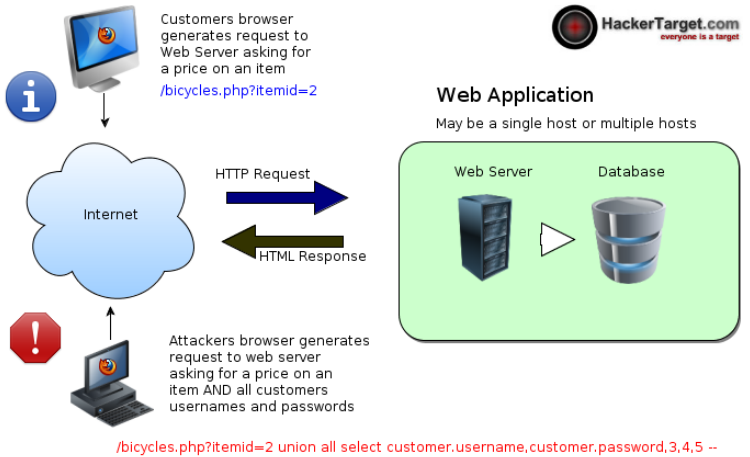
- "All we can teach people about server-side programming in a few hours is how to create security holes, even if we use modern frameworks."
- Think about security throughout the design
- Apply the principle of minimum access
  - Holes in the software shouldn't
    - Be able to harm anyone but the user invoking them
    - Give the user privileges they don't need
  - Restrict people's permissions



SQL Injection Attacks

- Let user = user_id from the web page
  - Authentication or just for information
- Code

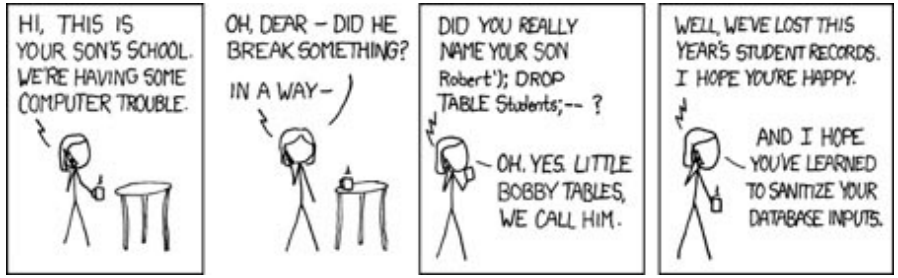
```
var q = "SELECT firstname
            FROM users
            WHERE name = '" + user + "'";
var rslt = db.query(q);
var msg = "Hello " + rslt[0].firstname;
```



SQL Injection Attacks

- Input: user_id = spr
  - SELECT firstname FROM users WHERE name = ‘spr’
  - Put the result on the result page (Hello Steven)
- Input: user_id = x’ or ‘1’ = ‘1
  - SELECT firstname
  - FROM users WHERE name = ‘x’ or ‘1’ = ‘1’

SQL Injection Attacks



- What if the user passes in

  *x'; DROP TABLE 'users'; SELECT * from passwords WHERE '1' = '1*

  SELECT firstname FROM users where name = 'x';

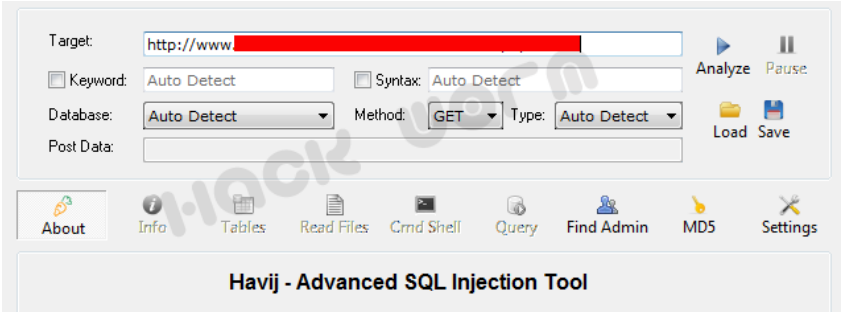
  DROP TABLE 'users';

  SELECT * FROM passwords WHERE '1' = '1'

  - This needs to be explicitly enabled (don't)
- Pass in *x' -- comment*
- Pass in a query that takes a long time

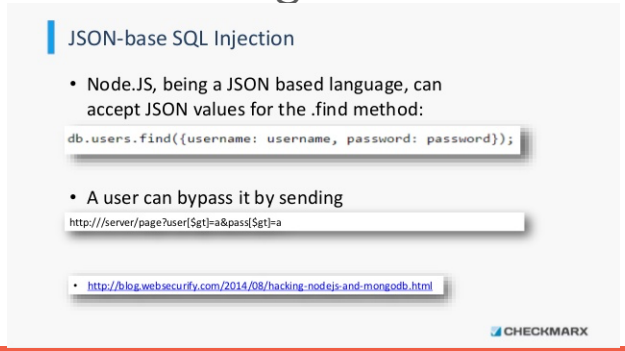
SQL Injection Attacks

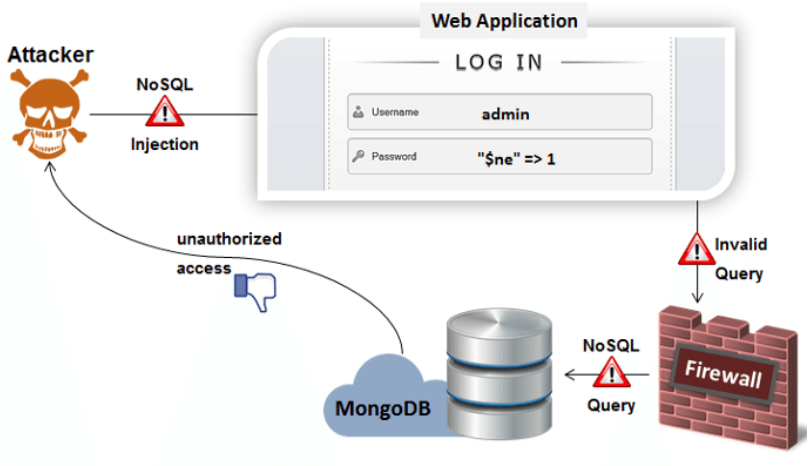
- The attacker needs to know
  - What the queries look like (the code)
  - The structure of the underlying database
- Can you determine this?
  - Yes – might take some time, but easy to do
  - Recipes are on the web
  - Tools are available



SQL Injection Attacks

- Aren't limited to SQL
- Can be used with mongo and other databases
  - Any time a query is constructed dynamically using input text
  - And the text isn't sanitized
  - Use with Node might be safe



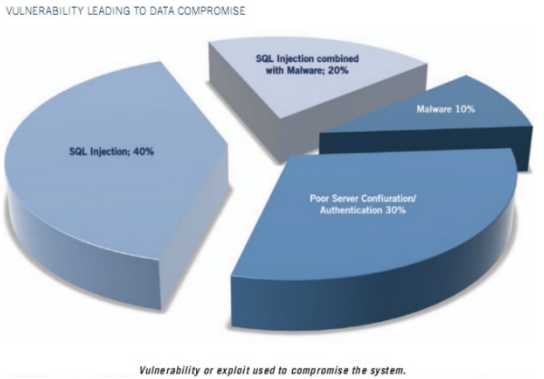


SQL Injection Attack Lesson

- Go to http://bdognom-v2.cs.brown.edu:5002
  - SQL Injection Attack Lesson
  - Try to log in
    - As any user; as administrator
  - Try it with only setting the user, not the password
    - Equivalent to password being hashed
  - Try this for 5-10 minutes
  - Raise (virtual) hand or thumbs up when done

SQL Injection Attacks

- Can do different malicious things based on query and database system
- Used to
  - email passwords to a different user
  - Get field and table names for the database
  - Find user names
  - Password guessing
  - Adding new users (super users)



Avoiding SQL Attacks

- Validate all requests before sending to server
  - Understand what data should look like
    - User ids, product ids, email, … have a given form
  - Use JavaScript in the client to do the validation
  - Is this sufficient?

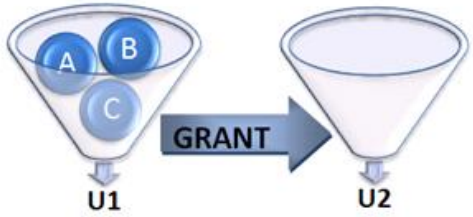
Avoiding SQL Attacks

- Validate the strings in the server
  - Most back end languages have a function for this
  - Check there are no funny characters
    - E.g. check that there are no quotes in the string
    - Is this sufficient?
      - » Value is a number
      - » Name is O'Reilly
      - » Unicode
  - Check each string is in a standard form
    - Is this sufficient?

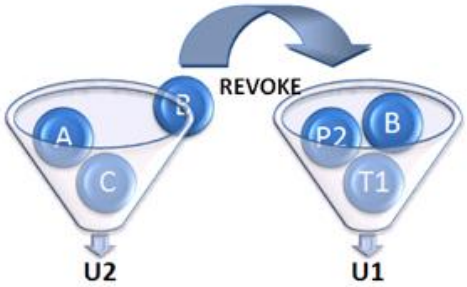
Avoiding SQL Attacks

- Don’t allow multiple statement queries
- Use prepared statements
  - SELECT firstname FROM USERS WHERE name = ?;
    - Pass in array of values separately
  - The substitution is handled by the database
    - Done correctly even with quotes, odd characters, etc.
- This is basically sufficient
  - Done automatically in RUBY, DJANGO, FLASK
  - This is what you should use

Avoiding SQL Attacks



- You will probably miss something however
- Make sure your database is relatively secure
  - Grant permissions to tables only as needed
  - Have separate database users for different roles
  - Limit access to the web application itself
  - Encrypt the database
  - **Principle of least access**



Next Time

- Security II

Homework

- What news stories did you find
  - Did they tell you what went wrong?

Question

A SQL injection attack involves

A. A malicious user inputting text that is used in a prepared SQL query to do malicious things.

B. A malicious user inputting text that is concatenated to form an unexpected SQL operation.

C. A malicious user adding JavaScript code to the web page to create new SQL operations in the back end.

D. A malicious user generating a XMLHttpRequest that cause the back end to add information to the database.

E. A malicious user adding code to the web server to execute their own SQL commands.

Question

A cross-site scripting attack (XSS) can involve:

A. One web site using its cookies to infect another.
B. A malicious user inserting text into a blog causing another user to run arbitrary JavaScript.
C. An IFRAME from one page accessing data from another page being displayed in the browser.
D. A malicious user redirecting traffic from one site to a look-alike,
E. None of the above