CSE 484 / CSE M 584 (Autumn 2011)

Web Security (cont.)

Daniel Halperin
Tadayoshi Kohno

Thanks to Dan Boneh, Dieter Gollmann, John Manferdelli, John Mitchell, Vitaly Shmatikov, Bennet Yee, and many others for sample slides and materials ...

Today, 10/31

- Finish web security
- Start user authentication
- Reminder: Catalyst posts are due this Friday, 11/4
- HW 2: office hours after class in CSE 210

Web Security so Far

- Need to secure both sides: User and Server
- HTTP(S), Forms, Cookies, JavaScript
- Servers shouldn't trust users
  - Validate/clean input
  - Check integrity of data, even in, e.g., hidden fields or cookies
- Servers shouldn't trust each other
  - e.g., XSS and CSRF attacks
- ***Lower parts of the stack*** need securing, e.g., DNS

Cheating the Same Origin Policy

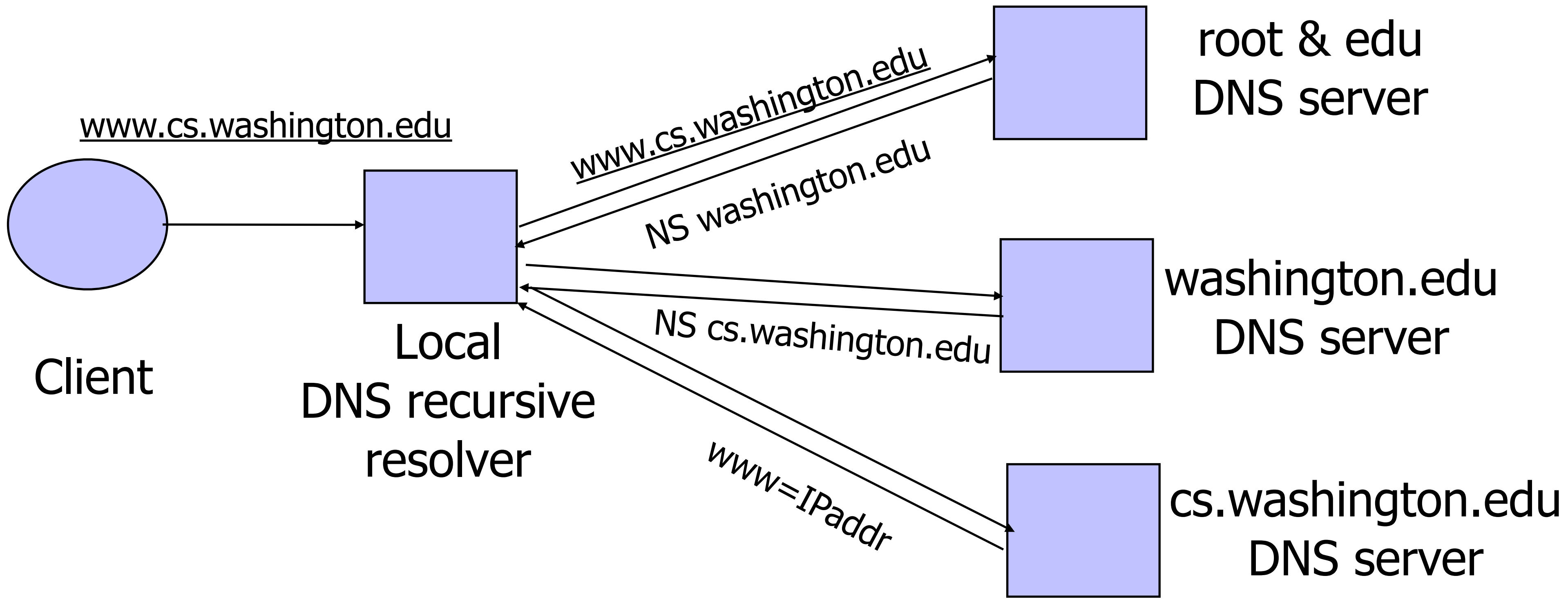
- JavaScript same-origin policy
  - Can only read properties of documents and windows from the same server, protocol, and port

- But can an attacker change the server?
  - Yes! If an attacker can control DNS (Domain Name Service)

DNS: Domain Name Service

DNS maps symbolic names to numeric IP addresses

(for example, www.cs.washington.edu ↔ 128.208.3.88)



DNS Vulnerabilities

- DNS host-address mappings are <u>not</u> authenticated
- DNS implementations have vulnerabilities
  - Reverse query buffer overrun in old releases of BIND
    - Gain root access, abort DNS service…
  - MS DNS for NT 4.0 crashes on chargen stream
    - telnet ntbox 19 | telnet ntbox 53
- Denial of service is a risk
  - If can't use DNS ... can't use the "Internet"

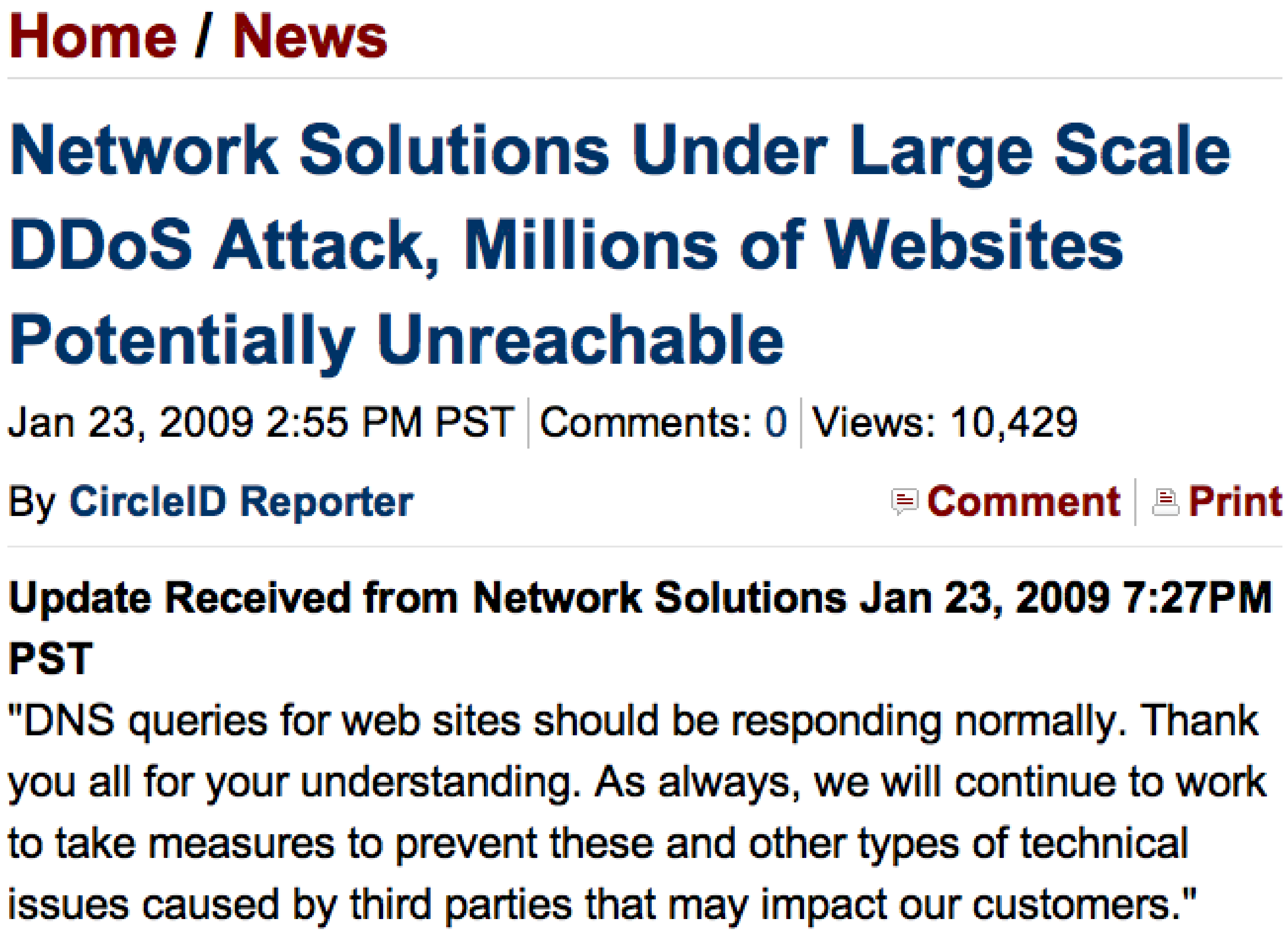
- Just recently (summer 2010) DNSSEC starting to be deployed
  - http://www.commerce.gov/news/press-releases/2010/07/16/commerce-department-icann-and-verisign-deploy-new-technology-enhance-

Reverse DNS Spoofing

- Trusted access is often based on host names
  - E.g., permit access to website from all .cs.washington.edu IPs
- Network requests such as Web or ssh arrive from numeric source addresses
  - System performs reverse DNS lookup to determine requester's host name and checks if it's in .htaccess
- If attacker can spoof the answer to reverse DNS query, he can fool target machine into thinking that request comes from an authorized host
  - No authentication for DNS responses and typically no double-checking (numeric → symbolic → numeric)

Other DNS Risks

- DNS cache poisoning
  - False IP with a high time-to-live will stay in the cache of the DNS server for a long time
  - Basis of pharming
- Spoofed ICANN registration and domain hijacking
  - Authentication of domain transfers based on email addr
  - Aug ’04: teenager hijacks eBay’s German site
  - Jan ’05: hijacking of panix.com (oldest ISP in NYC)
    - "The ownership of panix.com was moved to a company in Australia, the actual DNS records were moved to a company in the United Kingdom, and Panix.com's mail has been redirected to yet another company in Canada."
- Misconfiguration and human error



Home / News

Network Solutions Under Large Scale DDoS Attack, Millions of Websites Potentially Unreachable

Jan 23, 2009 2:55 PM PST | Comments: 0 | Views: 10,429

By CircleID Reporter — Comment | Print

Update Received from Network Solutions Jan 23, 2009 7:27PM PST

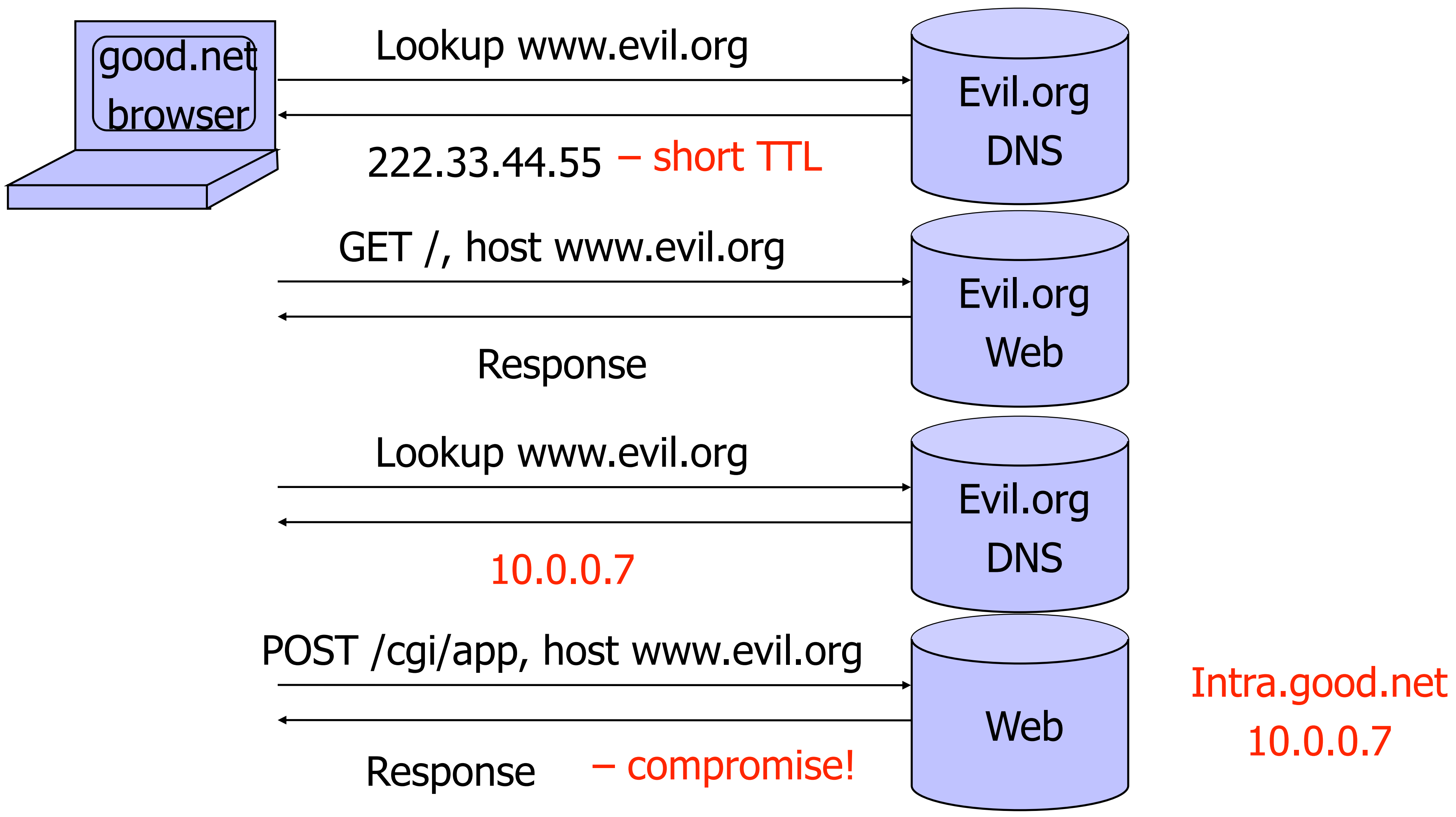
"DNS queries for web sites should be responding normally. Thank you all for your understanding. As always, we will continue to work to take measures to prevent these and other types of technical issues caused by third parties that may impact our customers."

. . .

JavaScript/DNS Intranet attack (I)

- Consider a Web server **intra.good.net**
  - IP: 10.0.0.7, inaccessible outside **good.net** network
  - Hosts sensitive CGI applications
- Attacker at **evil.org** gets **good.net** user to browse **www.evil.org**
- Places Javascript on **www.evil.org** that accesses sensitive application on **intra.good.net**
  - This doesn't work because Javascript is subject to "same-origin" policy
  - … but the attacker controls evil.org DNS

JavaScript/DNS Intranet attack (II)



Web Security so Far

- ...
- *Lower parts* of the stack need securing, e.g., DNS
- **Higher parts** of the stack need securing, e.g., SQL
- ...

User Data in SQL Queries

- set UserFound=execute(

  “SELECT * FROM UserTable WHERE ”

  “username=’ ” & form(“user”) & “ ’ AND ”

  “password=’ ” & form(“pwd”) & “ ’ ”);

  - User supplies username and password, this SQL query checks if user/password combination is in the database

- If not UserFound.EOF

  Authentication correct

  else Fail

  Only true if the result of SQL query is not empty, i.e., user/pwd is in the database

- (Notation approximate, to focus on key issues)

SQL Injection

Always true!

- User gives username ’ OR 1=1 --
- Web server executes query

```
set UserFound=execute(
        SELECT * FROM UserTable WHERE
        username=’ ’ OR 1=1 -- … );
```

Everything after -- is ignored!

- This returns the entire database!
- UserFound.EOF is always false; authentication is always “correct”

It Gets Better (or Worse?)

- User gives username

```
’ exec cmdshell  ’net user badguy badpwd’ / ADD --
```

- Web server executes query

```
set UserFound=execute(
      SELECT * FROM UserTable WHERE
      username=’ ’ exec … -- … );
```

- Creates an account for badguy on DB server

Uninitialized Inputs

```
/* php-files/lostpassword.php */
for ($i=0; $i<=7; $i++)
     $new_pass .= chr(rand(97,122))
…
$result = dbquery("UPDATE ".$db_prefix."users
    SET user_password=md5('$new_pass')
    WHERE user_id='".$data['user_id']." ' ");
```

In normal execution, this becomes

```
UPDATE users SET user_password=md5('???????')
WHERE user_id='userid'
```

Uninitialized Inputs

```
/* php-files/lostpassword.php */
for ($i=0; $i<=7; $i++)
     $new_pass .= chr(rand(97,122))
…
$result = dbquery("UPDATE ".$db_prefix."users
    SET user_password=md5('$new_pass')
    WHERE user_id='".$data['user_id']." ' ");
```

Creates a password with 7 random characters, assuming $new_pass is set to NULL

In normal execution, this becomes

```
UPDATE users SET user_password=md5('???????')
WHERE user_id='userid'
```

Uninitialized Inputs

```
/* php-files/lostpassword.php */
for ($i=0; $i<=7; $i++)
    $new_pass .= chr(rand(97,122))
…
$result = dbquery("UPDATE ".$db_prefix."users
    SET user_password=md5('$new_pass')
    WHERE user_id='".$data['user_id']." ' ");
```

Creates a password with 7 random characters, assuming $new_pass is set to NULL

In normal execution, this becomes

```
UPDATE users SET user_password=md5('???????')
WHERE user_id='userid'
```

SQL query setting password in the DB

Exploit

User appends this to the URL:

&new_pass=badPwd%27%29%2c
user_level=%27103%27%2cuser_aim=%28%27

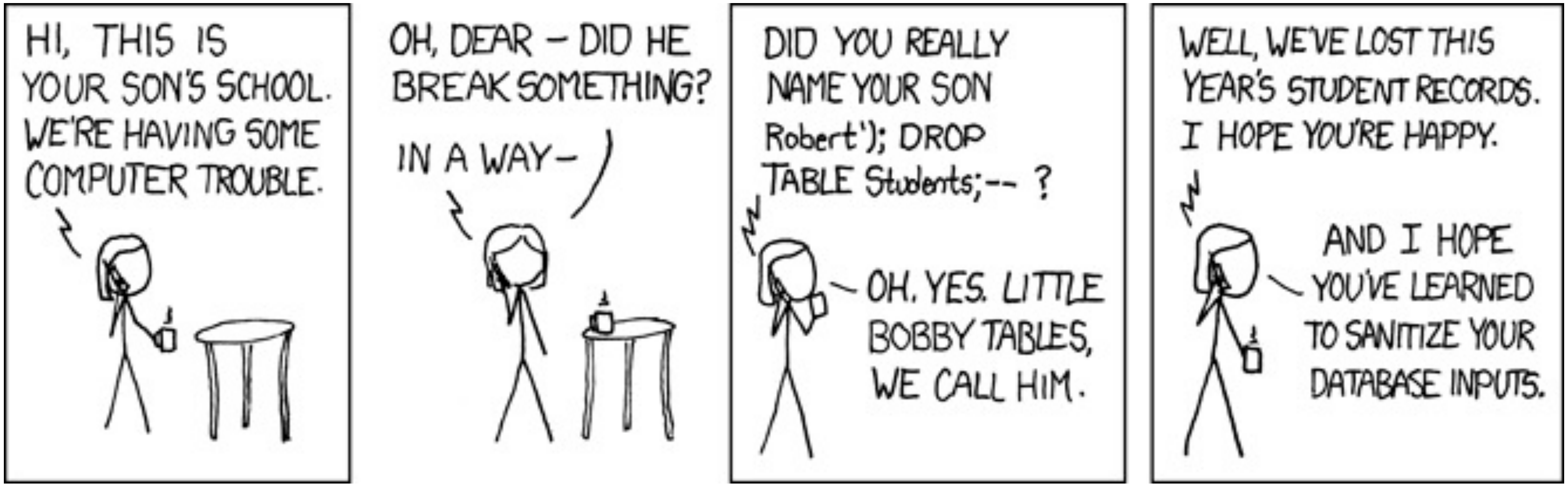
This sets $new_pass to
badPwd’), user_level=‘103’, user_aim=(‘

SQL query becomes

```
UPDATE users SET user_password=md5(‘badPwd’)
            user_level=‘103’, user_aim=(‘???????’)
WHERE user_id=‘userid’
```

… with superuser privileges

User’s password is set to ‘badPwd’


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.

http://xkcd.com/327/

Finally: don’t forget about the user side!

Dangerous Websites

- 2006 “Web patrol” study at Microsoft identified 752 unique URLs that could successfully exploit unpatched Windows XP machines
  - Many are interlinked by redirection and controlled by the same major players
- “But I never visit risky websites”
  - 11 exploit pages are among the top 10,000 most visited
  - Common trick: put up a page with popular content, get into search engines, page redirects to the exploit site
    - One of the malicious sites was providing exploits to 75 “innocuous” sites focusing on (1) celebrities, (2) song lyrics, (3) wallpapers, (4) video game cheats, and (5) wrestling
- Similar study at UW
- Now distributed through emails and ads

Browser Security Design

- "*App Isolation: Get the Security of Multiple Browsers with Just One*"
- Talk **Wednesday** @4:30 in CSE 403 by **Charlie Reis (UW CSE Ph.D. '09)**

Abstract: ***Many browser-based attacks like XSS, CSRF, and even renderer exploits can be prevented by using separate browsers for separate web sites.*** That's not practical for most users, so it would be nice to get similar protection within a single browser. In this talk, I will discuss which aspects of using multiple browsers help with security and how we can get these benefits in a single browser, with some compatibility costs. I will describe how we are starting to deploy these ideas in Chrome as an opt-in feature for apps in the Chrome Web Store.