Internet Security

All the PIRACY, none of the SCURVY.

PL Fencing Day

- Wed Apr 27 (unless it rains) @ 3:30pm
- Darden Courtyard; no experience necessary
- I provide gear
- I provide lesson

Final Projects Due Mon May 7

- The "automated testing" is a favor, not a necessity
  - PS5/7 is still due even if it is down (email me)

One-Slide Summary

- **Physical security** and **operating system security** are of critical importance and must be understood.

- Key issues in internet security, including **buffer overruns**, **virus detection**, **spam filtering**, **SQL code-injection attacks**, and **cross-site scripting** can all be understood in terms of *lexing* and *parsing*.

High-Level Lecture Today!

Select the correct answer.

The IDS monitors and collects network system information and analyzes it to detect attacks or intrusions.

○ True

○ I don't know

Lecture Outline

- Physical Security
- Operating System Security
  - Privileges
- Viruses and Scanning
- Side Channel and Non-Control Data Attacks
- Spam and Filtering
- SQL Injection Attacks

Physical Security

- It is generally accepted that anyone with physical access to a machine (i.e., anyone who can open the case) can *compromise that entire machine*.

- Given physical access …
  - How would I read your personal files?
  - How would I leave a backdoor (rootkit) for myself?
  - How would I log in as you?

- Ignore networked/encrypted filesystems for now …

- **Them:** Important user, NT box, lost admin password, sad, sad, sad.
- **Me:** No problem, change password with magic linux disk, offline NT password editor.
- **Them:** No, no, no. Never work. NT secure. Get real.
- **Me:** Watch. (reboot)
- **Them:** Gasp! This floppy is dangerous! Where did you get it?
- **Me:** Internet. Been around forever.
- **Them:** How do we keep students from using this?
- **Me:** Can't. Migrate. Linux. Mac.
- **Them:** No, no, no. Just make NT safe.
- **Me:** Can't. NT inherently unsafe.
- **Them:** Must be safe. NT good. We have never seen problems.
- **Me:** You just saw one now.
- **Them:** No, no, no. NT good. Win2k better.
- **Me:** Win2k is NT. Same thing. Should I give this floppy to a student?
- **Them:** No, no, no. Give here.
- **Me:** Whatever. What do you want me to do?
- **Them:** Change admin password.
- **Me:** Fine. To what?
- **Them:** "p-a-s-s-w-o-r-d"
- **Me:** No, no, no.

A Fairy Tale? Not Quite.

Hey You! Get Off Of My Lawn!

- Must keep people out of the server room …
- Heavy-weight physical security measures are often skipped entirely
- They are "not worth it" to the people involved
- *Social engineering*
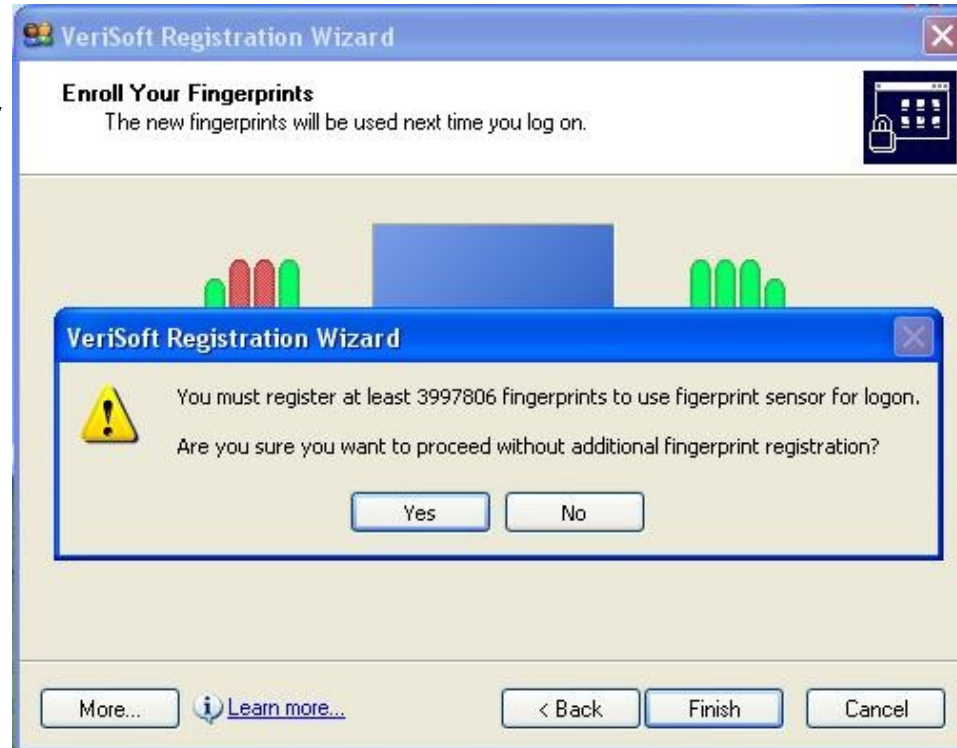
Corporate Espionage

- In 1999, Fortune 1000 companies lost more than $45 billion to corporate espionage

- Office card keys ("no drafting") and dumpster-diving prevention are two Top Five ways to defeat espionage

- *Social engineering awareness* is much more important, however!

NEW DOOR LOCK FITTED FOR ENTRY TYPE IN CODE: 4.5.8.0.X ON KEYPAD

Death By Heat Lamps?

- Sophisticated physical attacks are possible
  - S. Govindavajhala and A. Appel: Using Memory Errors to Attack a Virtual Machine. *IEEE Symposium on Security and Privacy*, 2003
- They write a Java program that can break out of the Java Virtual Machine if a single bit error occurs in memory …
  - Shine lamp on memory!
- For the rest of this talk I'll assume physical security.
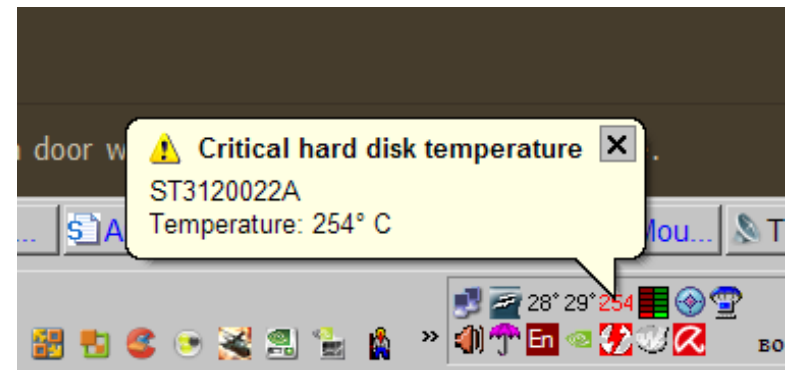
Is Unix Any Better?

- No; if you have physical access to a unix machine you can get root access.
  - Linux example: reboot, wait for GRUB/LILO, ask for the bootloader prompt, and type:

  `linux init=/bin/bash`

- One solution: store important files on encrypted (sub-)filesystem
  - Either requires frequent password entry or stores password in memory
  - This is only secure if no malicious programs run
  - Thus: we still need operating system security!

Unix Security Model

- All files in Unix filesystems have *permissions*

  - **-rwxr-xr-x** 1 root root 735004 2008-01-15 09:29 /bin/bash

- Three levels: user, group, others

- Exception: a special *root* user can change the permissions on any file (and thus do anything)

- Passwords must be stored for login to work

- Password file stores *hashes*:

  - smt6k:SASsHTBDJKdsa4:510:511:Sean Talts:/home/smt6k:/bin/bash

  - eas2h:p3612PxZBAx37ne:511:513:Elizabeth Soechting:/home/eas2h:/bin/bash

  - dsn9m:aw73sXHa3I3dn348:512:514:David Noble:/home/dsn9m:/bin/bash

Trojan Horses

- root is convenient … but also dangerous!
- Suppose you are running out of disk space and are hunting around for files to remove
  - Evil user makes evil files called "ls" and "dir"
  - These trojan horses email your password to Microsoft *and then* list the files
  - You may never know you've been tricked!
- **This single concept accounts for the vast majority of windows vulnerabilities**
  - Pre-Vista you were always "root", so if I could get you to click on some evil program I send over the network, I could take over your computer.

Detecting Malicious Programs

- So we need to detect viruses/trojans/worms
- This is done by lexing (no, really)
- A virus or trojan typically leaves most of the program unchanged (to avoid suspicion) and tacks on a special *payload* for dirty work
- Make one regular expression for each payload
  – Called the virus *signature*
- Scan (lex) programs with union of regexps
  – A virus database file is basically just a .lex file and each new version has some new "tokens"

Escalation

- One key problem with this approach is that you must constantly update your database of virus signatures in response to new virus inventions
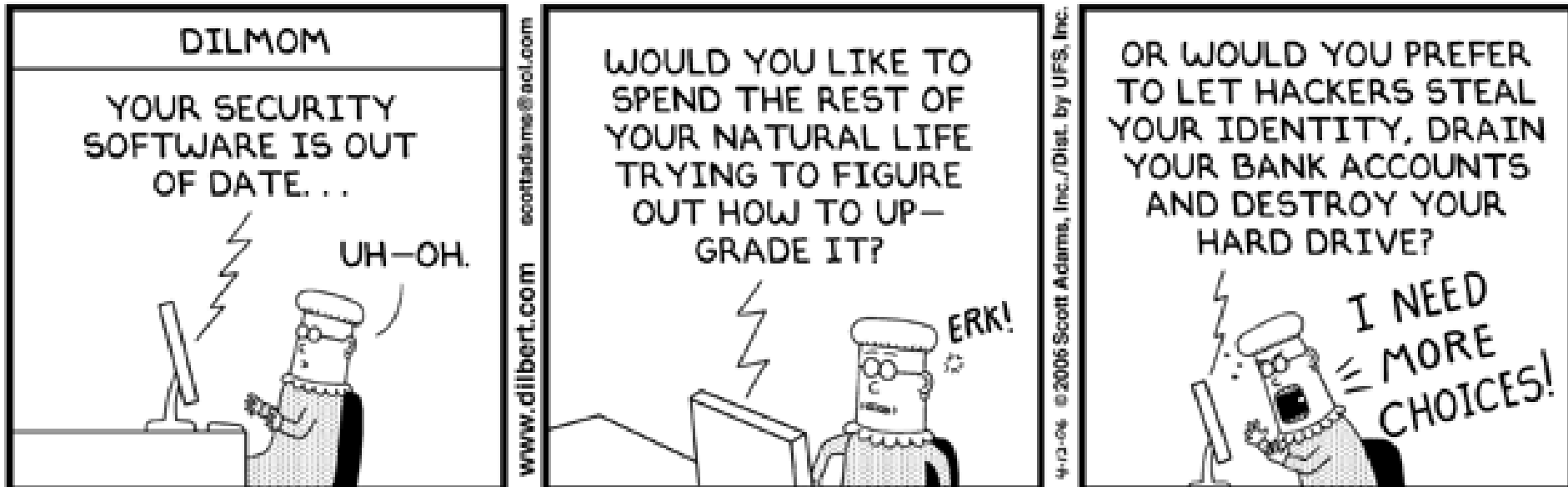
Does This Work?

- Assume we've solved the update problem.
- What could go wrong with searching for exact code sequences?

Stealth

- Any change to the virus defeats the signature
- Beware: self-modifying virus!
- *Encryption* with a new key per file
  - payload = decrypt module + encrypted virus code
- *Polymorphic* Virus: new decrypt per file
  - payload = unique decrypt + encrypted virus code
- *Metamorphic* Virus: rewrite each time
  - Basically: insert no-ops, "optimize" virus, etc.
  - Win32/Smile is >14000 lines of ASM, 90% of which is metamorphic engine … and was out in 2002

Virus Scanners In Practice

- Offline: unix servers scan win32 attachments
  - Basically just like PA2
- Online: scan every file before it is executed
  - Requires **OS support**: register a callback whenever a program or DLL is loaded *(why require OS support?)*
  - Or whenever a file is opened in general
  - This is very slow (cf. games)
- Viruses need privileges (e.g., read and write other files), so one defense is to not have those privileges …

My Secret Identity

- If you know another user's password, you can become that user (i.e., substitute its userid for yours --- like logging in as that person)

- The *su* and *sudo* programs implements this

Using a root account is rather like being Superman; an administrator's regular user is more like Clark Kent. Clark Kent becomes Superman for only as long as necessary, in order to save people. He then reverts to his "disguise". Root access should be used in the same fashion. The Clark Kent disguise doesn't really restrict him though, as he is still able to use his super powers. This is analogous to using the sudo program.
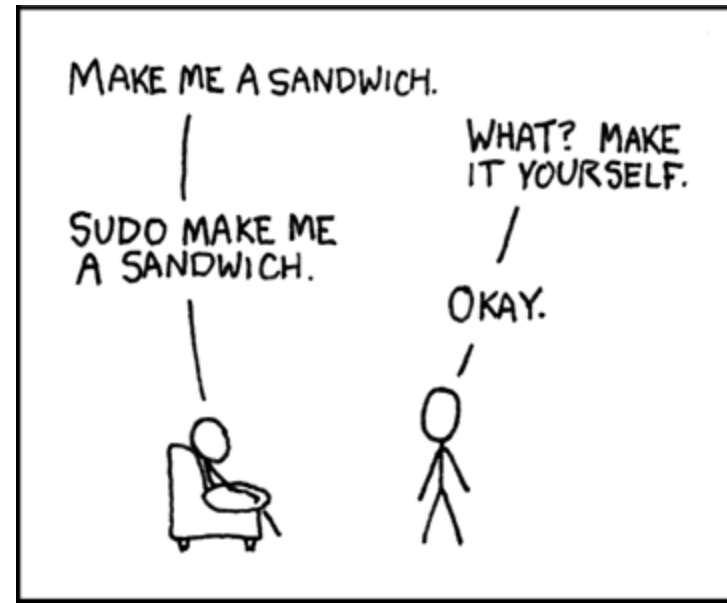
A Sendmail Dilemma

- Some programs, such as **sendmail**, must run as root to do useful work

  – Mail programs must be able to append incoming mail to the end of a given user's mailbox file

- These programs also do less-critical work

  – Mail programs may run a user-specified "vacation" program that responds to mail with "I'm away for two weeks"-style messages

- Any possible problems?

Dropping Privileges

- Important system tasks that must run as root try to drop those privileges as quickly as possible
    - Sendmail appends incoming mail to your inbox, then throws away its super powers, then runs your vacation program
- However, if you have a buffer overrun (or somesuch) I may be able to trick you into doing something before you drop privileges

Setuid Demystified

- Dropping privileges correctly is tricky, but that's another story ... [Chen, Wagner, Dean. *Usenix '02*]
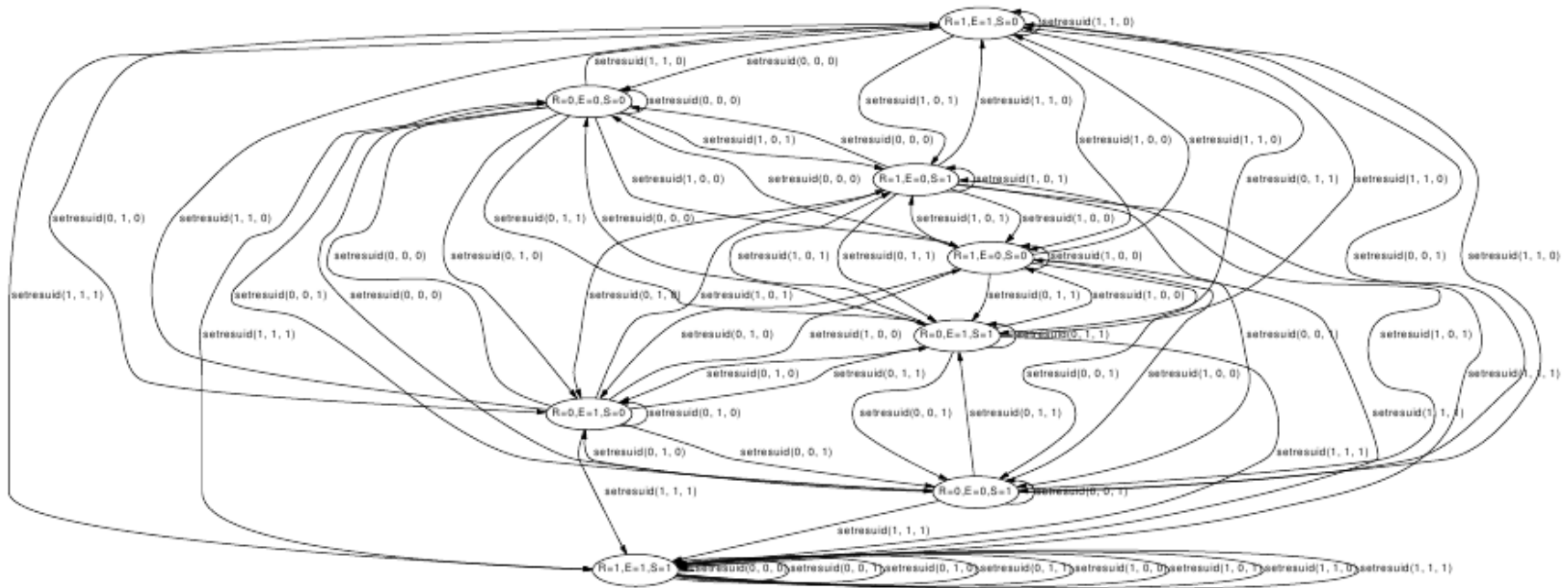
(c) An FSA describing *setresuid* in Linux

Figure 5: Three finite state automata describing the *seteuid*, *setreuid*, *setresuid* system calls in Linux respectively. Ellipses represent states of the FSA, where a notation like "R=1,E=0,S=1" indicates that *euid* = 0 and *ruid* = *suid* ≠ 0. Each transition is labelled with the system call it corresponds to.

Leaking Information

- Consider this version of login: what's wrong?

```
let name = recv_from_network () in
let pword = recv_from_network () in
let file = open_in ("/etc/passwd") in
while not end_of_file(file) do
  let name', hpword' = read_from (file) …
  if name = name' then
     return (hash(pword) = hpword')
done ;
return FALSE
```

Side-Channel Attacks

- Imagine it takes t microseconds to read in the entire password file
  - Then it takes t microseconds to return false for a made-up username
  - But $t/2$ microseconds (on average) to return false for a real username with a bad password
- A *side-channel attack* is any attack based on information gained from the implementation of a cryptosystem, *not* from a theoretical weakness
  - Examples: timing info, power consumption, electromagnetic leaks (TEMPEST), …
  - http://en.wikipedia.org/wiki/TEMPEST

Non-Control Data Attacks

```
remote_cmd(socket) {
  bool auth = false;
  char name[1024], pword[1024], cmd[1024];
  recv(socket, name);
  recv(socket, pword);
  if (matches(name,pword)) auth = true;
  if (!auth) then return false;
  recv(socket, cmd);
  if (auth) exec(cmd);
} // Buffer Overrun 2 (Electric Buffaloo) – why?
```

Q: Theatre (011 / 842)

- Complete the song verse from Rodgers and Hammerstein's 1949 musical **South Pacific** that begins *"I'm gonna wash that …"*.

- In addition to its title character, this 1966 song, recorded with an eight-person string section, describes the sock-darning Father McKenzie.

- Which of the following mythical creatures cannot turn people to stone?
  - Basilisk
  - Cockatrice
  - Golem
  - Gorgon

spam bacon sausage …

- Not everyone is running a server that I can exploit ... how can I get a payload to you?

- *Spamming* is abusing an electronic messaging system (i.e., email) to send unsolicited bulk messages.

- Started in the mid-1990s, spam now accounts for 80-85% of all email in the world (conservative) to as much as 95% of all world email.

  - European Union Internal Market Commission: €10 billion per year worldwide in '01

  - CA Legislature: $13 billion alone to US companies in '07

- Today most spam is sent from *zombie* networks of virus-infected machines

Why does spam work?

- Based on physical-world direct mail, bulk mail, targeted marketing, etc.
  - Like those advertising circulars you get with grocery store coupons in them
  - Those work because you can get huge amounts of statistical information just from the zip code
  - … and because people go to nearby supermarkets
- Example: in 2005, a random house in 22904
  - AGI of $67,125 and was headed by someone under 30 (36%), 30-44 (25%), 45-60 (22%) or 61+ (15%)
- Bulk physical mail is *not* a shot in the dark
  - Benefit (medium) exceeds cost (low)

SPAM

- Spam also works because of a cost-benefit analysis
  - Benefit (mico)
  - Cost (none) *(why?)*
- Ultimately, *some people click on spam*.
  - Not just *phishing* spam either!

Harvesting

- How do I get a list of email addresses?
- *Dictionary Spamming*
  - Guess by using a dictionary of plausible names as prefixes to known (registered) domain names
- *Spambot Web Crawling*
  - Gather from web sites, newsgroups, special-interest group postings, chat-room conversations
  - Basically, regular expressions! (cf. early HW)
  - Wow, it's lexing again!
- Selling email lists is a big business ...

Stopping Spam

- *Blacklisting* – do not accept messages from domain X?
  - Defeated by zombie botnets, remailers, …
- How to find domain X?
  - Wait for users to report it …
  - *List poisoning*: subscribe fake "honeypot" email addresses to mailing lists, post them on web: any email that gets to them is spam
- Other, more technical approaches (e.g., greylisting), but mostly …

Filtering

- *Filtering* – examine the contents of an email message and try to predict mechanically if it is spam or not
  - Simplest approach: block words (e.g., viagra)
  - Easily thwarted: (v1agra)
  - More complex: bayesian network filtering …

The PC Weenies®

©2005 Krishna M. Sadasivam

Kan't Spel?
No probblim!
Mak munee one the NETT!
Joyn ar mas-markit maling teem TU-DAE!

www.pcweenies.org

KMS 9-10-05
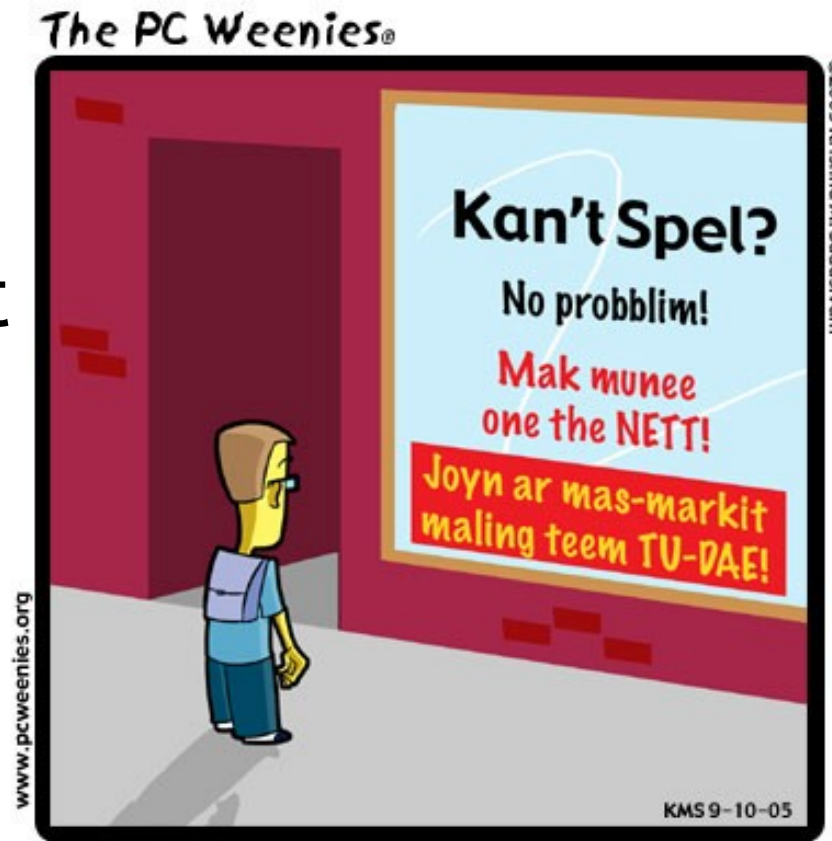
RECRUITMENT ADS FOR POTENTIAL SPAMMERS

SPAM Solutions

- Ultimate problem is that sending email is free
  - The Tragedy of the Commons *(read on Wikipedia)*
- SMTP, the current mail protocol, is an entrenched legacy problem
- Thus only incremental solutions are viable
- Training models to discriminate between spam and valid email is an open area of research!
- Crackpot solutions are a dime a dozen, as we can see by this idea rejection simple chart …

Your post advocates a

( ) technical ( ) legislative ( ) market-based ( ) vigilante

approach to fighting spam. Your idea will not work because:

( ) Spammers can easily use it to harvest email addresses

( ) Mailing lists and other legitimate email uses would be affected

( ) No one will be able to find the guy or collect the money

( ) It is defenseless against brute force attacks

( ) It will stop spam for two weeks and then we'll be stuck with it

( ) Users of email will not put up with it

( ) Microsoft will not put up with it

( ) The police will not put up with it

( ) Requires too much cooperation from spammers

( ) Requires immediate total cooperation from everybody at once

( ) Many email users cannot afford to lose business or alienate potential employers

( ) Spammers don't care about invalid addresses in their lists

( ) Anyone could anonymously destroy anyone else's career or business

Specifically, your plan fails to account for:

( ) Laws expressly prohibiting it

( ) Lack of centrally controlling authority for email

( ) Open relays in foreign countries

( ) Ease of searching tiny alphanumeric address space of all email addresses

( ) Asshats

( ) Jurisdictional problems

( ) Unpopularity of weird new taxes

( ) Public reluctance to accept weird new forms of money

( ) Huge existing software investment in SMTP

( ) Susceptibility of protocols other than SMTP to attack

( ) Willingness of users to install OS patches received by email

( ) Armies of worm riddled broadband-connected Windows boxes

( ) Eternal arms race involved in all filtering approaches

( ) Extreme profitability of spam

( ) Joe jobs and/or identity theft

( ) Technically illiterate politicians

( ) Extreme stupidity on the part of people who do business with spammers

( ) Dishonesty on the part of spammers themselves

( ) Bandwidth costs that are unaffected by client filtering

( ) Outlook

and the following philosophical objections may also apply:

( ) Ideas similar to yours are easy to come up with, yet none have ever been shown practical

( ) Any scheme based on opt-out is unacceptable

( ) SMTP headers should not be the subject of legislation

( ) Blacklists suck

( ) Whitelists suck

( ) We should be able to talk about Viagra without being censored

( ) Countermeasures should not involve wire fraud or credit card fraud

( ) Countermeasures should not involve sabotage of public networks

( ) Countermeasures must work if phased in gradually

( ) Sending email should be free

( ) Why should we have to trust you and your servers?

( ) Incompatiblity with open source or open source licenses

( ) Feel-good measures do nothing to solve the problem

( ) Temporary/one-time email addresses are cumbersome

( ) I don't want the government reading my email

( ) Killing them that way is not slow and painful enough

#38

Cat and Mouse

- Suppose I have a server (e.g., Amazon.com)
- Let's imagine that I have solved …
  - Viruses: no malicious code on machine
  - Buffer overruns: no injection of evil assembly code
  - Buffer overruns: no non-control data attacks
  - Privileges: no running as root
  - Spam: as long as I'm dreaming, I'd like a pony …
- I can still convince the server to do the wrong thing with the resources it legitimately has access to …

Three-Tier Web Application

- This is how Amazon is structured
- Query is a SQL database command generated by program logic

Presentation tier

The top-most level of the application is the user interface. The main function of the interface is to translate tasks and results to something the user can understand.

> GET SALES TOTAL

> GET SALES TOTAL
4 TOTAL SALES

Logic tier

This layer coordinates the application, processes commands, makes logical decisions and evaluations, and performs calculations. It also moves and processes data between the two surrounding layers.

GET LIST OF ALL SALES MADE LAST YEAR

ADD ALL SALES TOGETHER

QUERY

SALE 1
SALE 2
SALE 3
SALE 4

Data tier

Here information is stored and retrieved from a database or file system. The information is then passed back to the logic tier for processing, and then eventually back to the user.

Database

Storage

The Problem In The Logic Tier

```
$userid = read_from_network();

if (!eregi('[0-9]+', $userid)) {
  unp_msg('You entered an invalid user ID.');
  exit;
}


$user = $DB->query("SELECT * FROM `unp_user`".
                   "WHERE userid='$userid'");


if (!DB->is_single_row($user)) {
   unp_msg('You entered an invalid user ID.');
   exit;
}
```

The Problem

```
$userid = read_from_network();

if (!eregi('[0-9]+', $userid)) {
  unp_msg('You entered an invalid user ID.');
  exit;
}


$user = $D                                    user`".
                                              id'");

if (!DB->is_single_row($user)) {
    unp_msg('You entered an invalid user ID.');
    exit;
}
```

Matches any string that contains a sequence of digits…

The Bad Place

```php
// $userid == "1'; DROP TABLE unp_user; --"

if (!eregi('[0-9]+', $userid)) {
  unp_msg('You entered an invalid user ID.');
  exit;
}

$user = $DB->query("SELECT * FROM `unp_user`".
                   "WHERE userid='$userid'");

if (!DB->is_single_row($user)) {
  unp_msg('You entered an invalid user ID.');
  exit;
}
```

The Bad Place: Destroying Data

```
// $userid == "1'; DROP TABLE unp_user; --"

if

}

$user = $DB->q             "SELECT * FROM `unp_user`".
                           "WHERE userid='$userid'");

if (!DB->is_single_row($user)) {
   unp_msg('You entered an invalid user ID.');
   exit;
}
```

```
SELECT * FROM `unp_user`
             WHERE userid='1';
DROP TABLE unp_user;
--'
```

Also A Bad Place: Viewing Data

```
// $userid == "1' OR 1 = 1 --"

if                                              ;



}


$user = $DB->q         "SELECT * FROM `unp_user` ".
                       "WHERE userid='$userid'");


if (!DB->is_single_row($user)) {
   unp_msg('You entered an invalid user ID.');
   exit;
}
```

```
SELECT * FROM `unp_user`
          WHERE userid='1'
          OR 1 = 1
   --'
```

SQL Code-Injection Vulnerabilities

- A *SQL injection* attack exploits a vulnerability in the database layer of an application whereby user input is incorrectly filtered for string literal escape characters or otherwise unexpected executed.

- Most common types of vulnerability in 2006:
  - **25.1% Cross-Site Scripting**
  - **14%   SQL Command Injection**
  - 7.9%   Buffer Overruns

- Attacks are easy and expose valuable data

Exploits Of A Mom

- The essence of SQL injection:
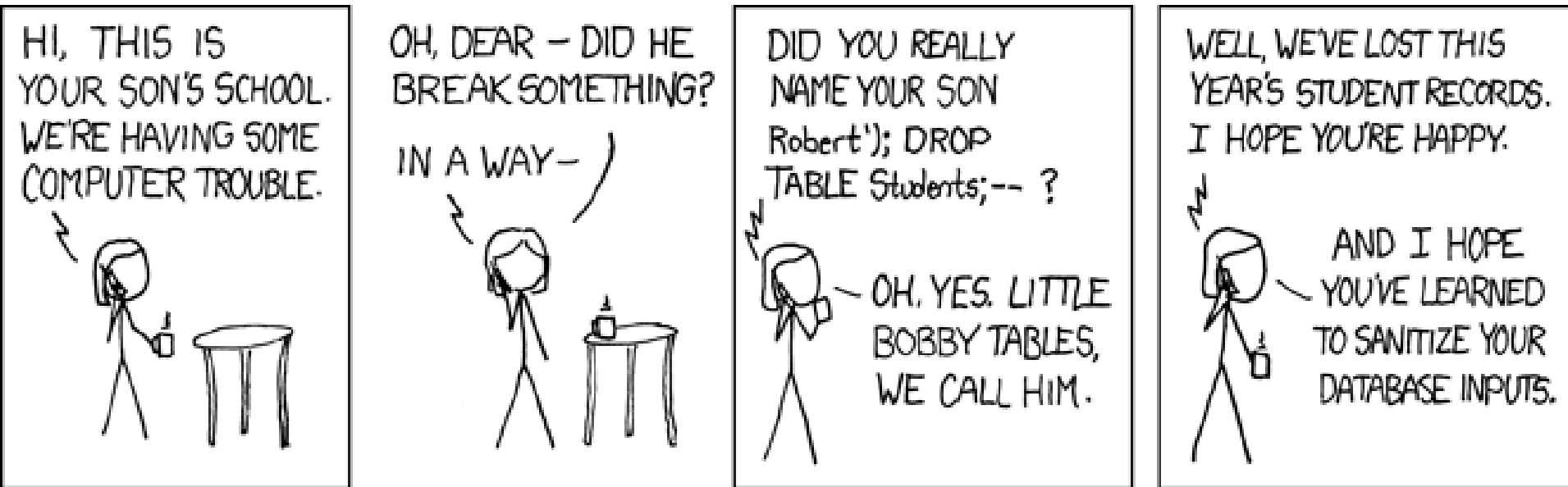
SQL Injection

- Note that it's basically a parsing problem
- We have a string constant in PHP plus a string constant from the user, and when combined they must make a valid SQL program
- One Solution: Dynamic Taint Analysis
  - Propagate a "taint" bit with every string
- One Solution: Dynamic Grammar Analysis
  - Partially parse PHP string fragment
  - If PHP string fragment + user string fragment parses to something with a different top-level structure, bail!

Parse Trees To The Rescue!

- Do the user input strings contribute to something "too high" on the parse tree?

Figure 4. Parse trees for WHERE clauses of generated queries. Substrings from user input are underlined.

Cross-Site Scripting

- *Cross-Site Scripting* (XSS): the same flavor
- Evil X posts a message with evil JavaScript in it (e.g., send passwords to me) to Blog B
  - Blog B can also be a forum, etc.
- Later, Luser browses Blog B
- Blog B sends over data, including Evil X's Message
- Luser thinks it is from Blog B (misplaced trust)
- Luser renders and interprets it

Stopping Evil Posts

- Evil network-crawling robots try to post evil JavaScript to every forum they can find

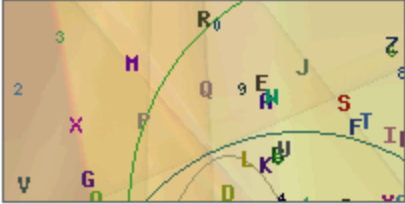- Let's require a real human when posting

- Increases cost

- *CAPTCHA*

  Complete Automated
  Public Turing test
  to tell Computers
  and Humans Apart

*City you require vehicle: Select city

*Comment/Query

Due to increased security, in order to complete your submission please copy the contents of the box OR calculate the mathematical problem into the box below the image. **Your answer is CASE SENSITIVE.**

Result from image:

Submit   Reset

Have We Won Yet?

- CAPTCHAs fail in theory and in practice
- The overarching problem is exactly the same:
  - The server takes input from an untrusted user
  - That input may be interpreted by another parser later
    - In SQL-CIVs, by the database's SQL parser
    - In XSS, by a user's JavaScript parser
  - So all of the same techniques apply for XSS
- Also, machines routinely win the Turing Test
  - http://en.wikipedia.org/wiki/Turing_test#Loebner_Prize

Homework

- Everything Due May 7th