CSE 484 / CSE M 584 (Spring 2012)

Web Security + User Authentication

Tadayoshi Kohno

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

Goals for Today

- Web Security
- User Authentication

- Lab 2 out today, due next Friday

Web Security...

- **...**
- *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

- (Notation approximate)

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

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']." ' ");
```

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

In normal execution, this becomes

SQL query setting password in the DB

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

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

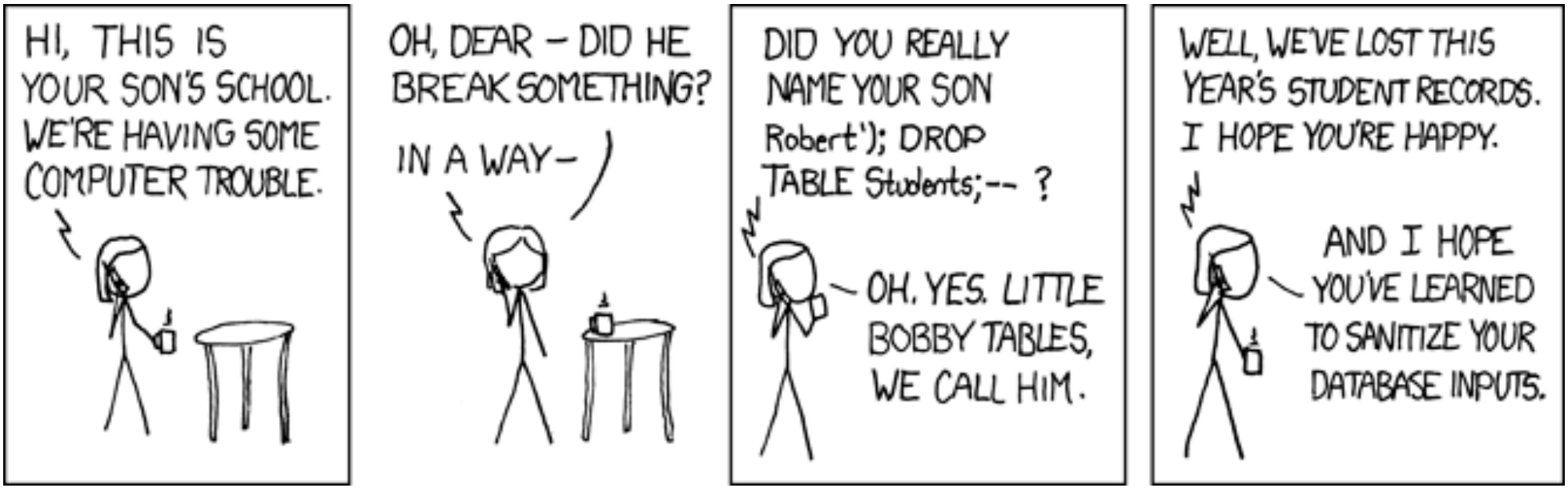
UPDATE users SET user_password=md5('badPwd')

user_level='103', user_aim=('???????')

WHERE user_id='userid'

... with superuser privileges

User's password is set to 'badPwd'



http://xkcd.com/327/

Also: 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
- Malware also distributed through emails and ads

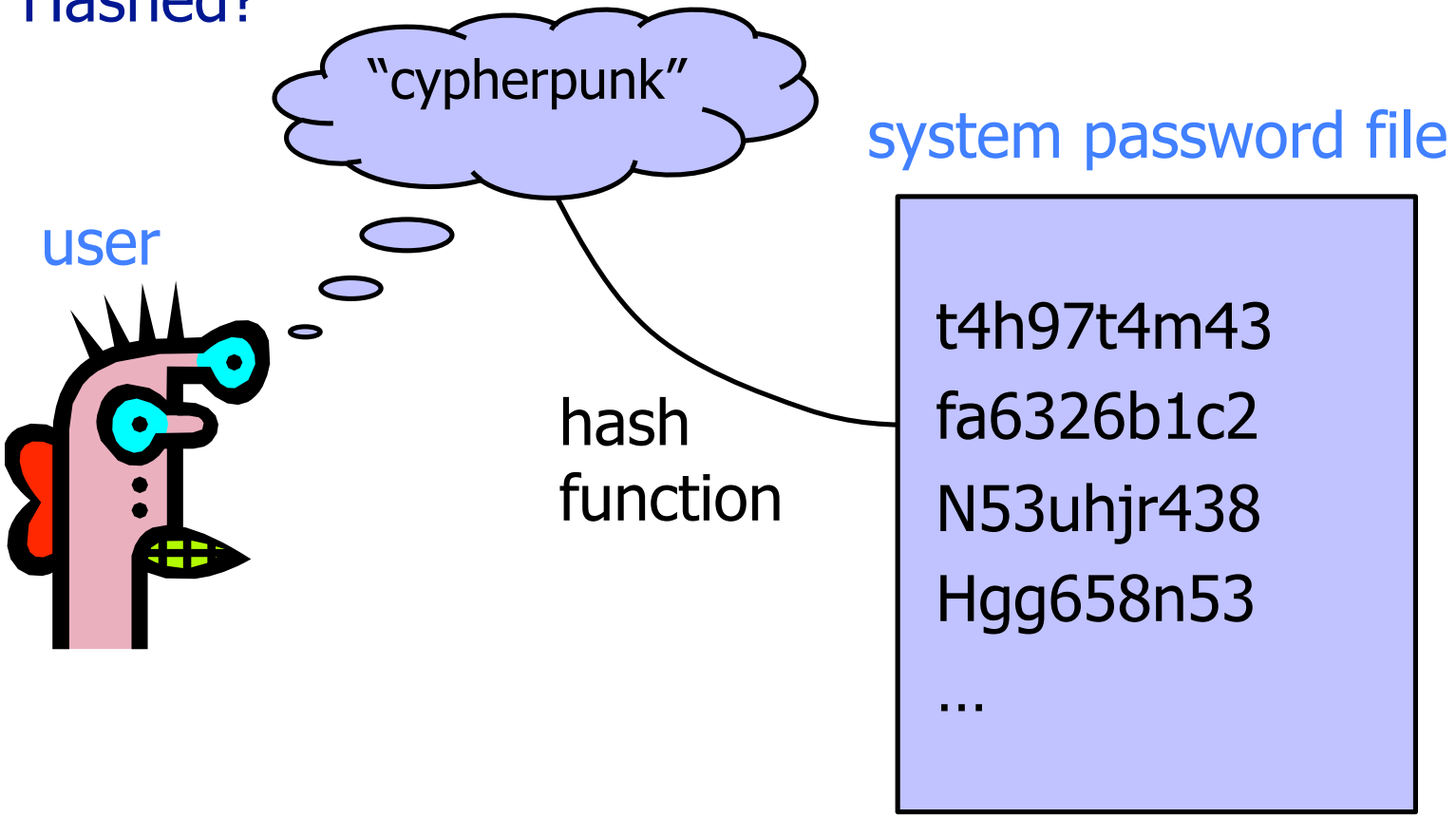
Server Authentication

- Q: How should we store passwords on a server?

- Q2: What threats are you worried about?

UNIX-Style Passwords

- How should we store passwords on a server?
  - In cleartext?
  - Encrypted?
  - Hashed?



Password Hashing

- Instead of user password, store H(password)
- When user enters password, compute its hash and compare with entry in password file
  - System does not store actual passwords!
  - System itself can't easily go from hash to password
    - Which would be possible if the passwords were encrypted
- Hash function H must have some properties
  - One-way: given H(password), hard to find password
    - No known algorithm better than trial and error
    - The attacker doesn't need to find **the** password, just **a** password that hashes to the stored value
  - "Slow" to compute

(Early) UNIX Password System

- Uses DES encryption as if it were a hash function
  - Encrypt NULL string using password as the key
    - Truncates passwords to 8 characters!
  - Artificial slowdown: run DES 25 times
    - Why 25 times? Slowdowns like these are important in practice!
  - ("Don't use DES like this at home.")
  - Can instruct modern UNIXes to use cryptographic hash function

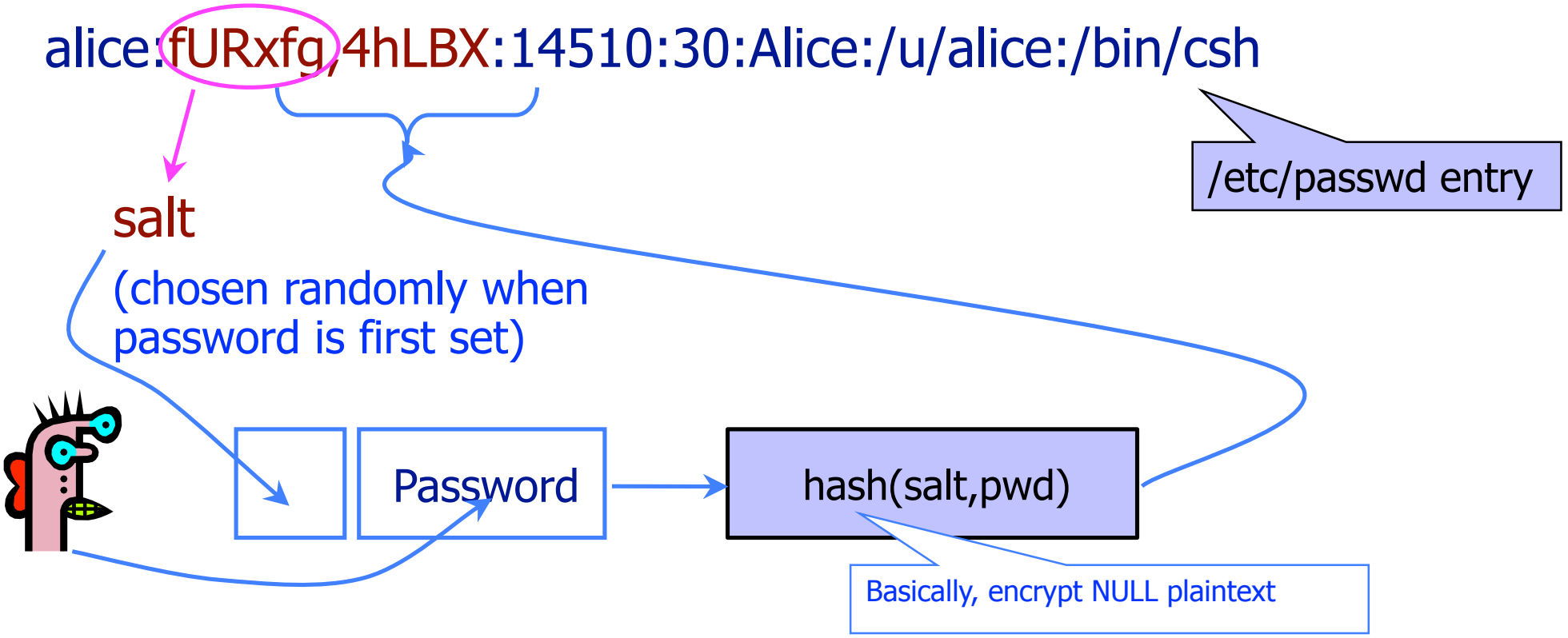
(Early) UNIX Password System

- Uses DES encryption as if it were a hash function
  - Encrypt NULL string using password as the key
    - Truncates passwords to 8 characters!
  - Artificial slowdown: run DES 25 times
    - Why 25 times? Slowdowns like these are important in practice!
  - ("Don't use DES like this at home.")
  - Modern systems use a cryptographic hash function
- Problem: passwords are not truly random
  - With 52 upper- and lower-case letters, 10 digits and 32 punctuation symbols, there are $94^8 \approx$ 6 quadrillion possible 8-character passwords (around 2^{52})
  - Humans like to use dictionary words, human and pet names $\approx$ 1 million common passwords

Dictionary Attack

- Password file /etc/passwd is world-readable
  - Contains user IDs and group IDs which are used by many system programs
- Dictionary attack is possible because many passwords come from a small dictionary
  - Attacker can compute H(word) for every word in the dictionary and see if the result is in the password file
  - With 1,000,000-word dictionary and assuming 10 guesses per second, brute-force online attack takes 50,000 seconds (14 hours) on average
    - This is very conservative. Offline attack is much faster!
  - As described (H(word)), could just create dictionary of "word to H(word)" mapping once -- for all users!!

Salt



- Users with the same password have <u>different</u> entries in the password file
- Online dictionary attack is still possible! (Precomputed dictionaries possible too -- but significantly more expensive.)

Advantages of Salting

- Without salt, attacker can pre-compute hashes of all dictionary words once for all password entries
  - Same hash function on all UNIX machines
  - Identical passwords hash to identical values; one table of hash values can be used for all password files
- With salt, attacker must compute hashes of all dictionary words once for each password entry
  - With 12-bit random salt, same password can hash to 2^{12} different hash values
  - Attacker must try all dictionary words for each salt value in the password file
- Pepper: Secret salt (not stored in password file)

Other Password Issues

- Keystroke loggers
  - Hardware
  - Software / Spyware
- Shoulder surfing
- Online vs offline attacks
  - Online: slower, easier to respond
- Multi-site authentication
  - Share passwords?


I AM MORDAC, THE PREVENTER OF INFORMATION SERVICES. I BRING NEW GUIDELINES FOR PASSWORDS.
S. Adams E-mail: SCOTTADAMS@AOL.COM
"ALL PASSWORDS MUST BE AT LEAST SIX CHARACTERS LONG... INCLUDE NUMBERS AND LETTERS... INCLUDE A MIX OF UPPER AND LOWER CASE..."
© 1998 United Feature Syndicate, Inc.
"USE DIFFERENT PASSWORDS FOR EACH SYSTEM. CHANGE ONCE A MONTH. DO NOT WRITE ANYTHING DOWN."
SQUEAL LIKE A PIG!!!

Recovery Passwords

- http://www.wired.com/threatlevel/2008/09/palin-e-mail-ha/

Palin E-Mail Hacker Says It Was Easy

By Kim Zetter ✉ September 18, 2008 | 10:05 am | Categories: Elections, Hacks and Cracks

A person claiming to be the hacker who obtaine
private
suppos
revealir
took to
Republi

> after the password recovery was reenabled, it took seriously 45 mins on wikipedia and google to find the info, Birthday? 15 seconds on wikipedia, zip code? well she had always been from wasilla, and it only has 2 zip codes (thanks online postal service!)
>
> the second was somewhat harder, the question was "where did you meet your spouse?" did some research, and apparently she had eloped with mister palin after college, if youll look on some of the screenshits that I took and other fellow anon have so graciously put on photobucket you will see the google search for "palin eloped" or some such in one of the tabs.
>
> I found out later though more research that they met at high school, so I did variations of that, high, high school, eventually hit on "Wasilla high" I promptly changed the password to popcorn and took a cold shower…

Password Reuse

- http://www.lightbluetouchpaper.org/2011/02/09/measuring-password-re-use-empirically/

Light Blue Touchpaper

Security Research, Computer Laboratory, University of Cambric

HOME | ABOUT | ARCHIVES | AUTHORS

Measuring password re-use empirically

February 9th, 2011 at 19:11 UTC by *Joseph Bonneau*

In the aftermath of Anonymous' revenge hacking of HBGary over the weekend, some enterprising hackers used one of the stolen credentials and some social engineering to gain root access at rootkit.com, which has been down for a few days since. There isn't much novel about the hack but the dump of rootkit.com's SQL databases provides another password dataset for research, though an order of magnitude smaller than the Gawker dataset with just 81,000 hashed passwords.

More interestingly, due to the close proximity of the hacks, we can compare the passwords associated with email addresses registered at both Gawker and rootkit.com. This gives an interesting data point on the widely known problem

"Improving" Passwords

- Add biometrics
  - For example, keystroke dynamics or voiceprint
  - Revocation is often a problem with biometrics
- Graphical passwords
  - Goal: increase the size of memorable password space
- Password managers
- Two-factor authentication
  - Leverages user's phone (or other device) for authentication

Two-Factor Authentication

Google Introduces Two-Factor Authentication Option

Users can now generate a second, one-time password for Gmail and other accounts using a mobile phone

Feb 11, 2011 | 04:09 PM | 0 Comments

By Tim Wilson
Darkreading

In an effort to help users increase the security of Gmail and other accounts, Google today introduced an option to add a second factor of authentication.

"As we announced to our Google Apps customers a few months ago, we've developed an advanced opt-in security feature called 2-step verification that makes your Google Account significantly more secure by helping to verify that you're the real owner of your account," Google says. "Now it's time to offer the same advanced protection to all of our users."