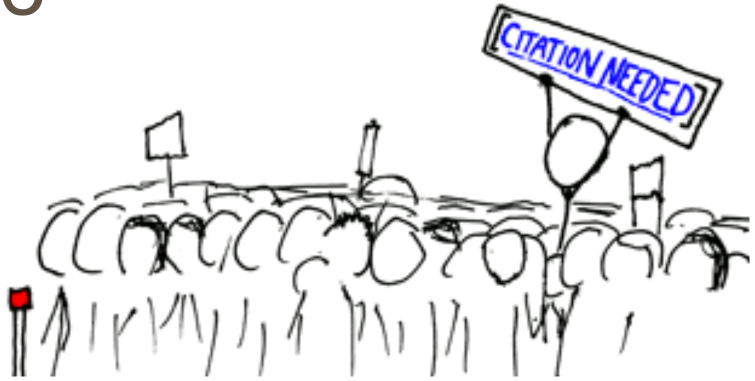
Security
[help from XKCD, Christo Wilson]

Lecture 16

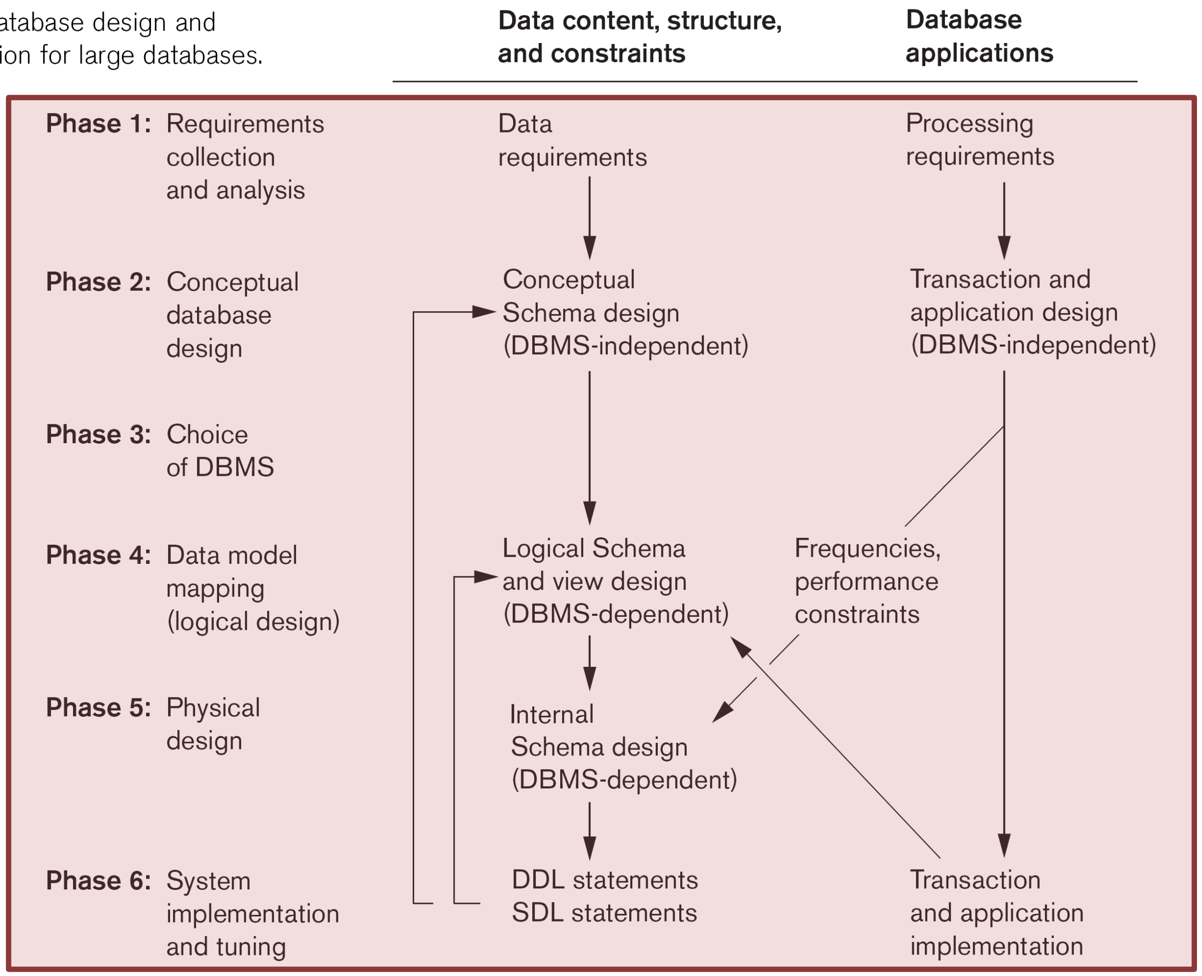


Outline

- Context
- Access Control
  - Strong password policies, 2FA
  - Discretionary, Mandatory
  - Least Privilege, Separate Privileges
- Attacks
  - SQL Injection
  - DoS (limit password length!)
  - Brute force password attempts (iCloud)
  - Internal vs. External (80% internal via Oracle)
  - Separate server, updates, audit logs
- Inference Control
- Encryption
  - Symmetric, Asymmetric, Hashing – tricky to get right!
  - Whole Database (and backups!), Communication
  - Sensitive Data, Password Storage

Database Design and Implementation Process

Figure 10.1
Phases of database design and implementation for large databases.


Data content, structure, and constraints
Database applications
Phase 1: Requirements collection and analysis
Data requirements
Processing requirements
Phase 2: Conceptual database design
Conceptual Schema design (DBMS-independent)
Transaction and application design (DBMS-independent)
Phase 3: Choice of DBMS
Phase 4: Data model mapping (logical design)
Logical Schema and view design (DBMS-dependent)
Frequencies, performance constraints
Phase 5: Physical design
Internal Schema design (DBMS-dependent)
Phase 6: System implementation and tuning
DDL statements
SDL statements
Transaction and application implementation

Guidelines

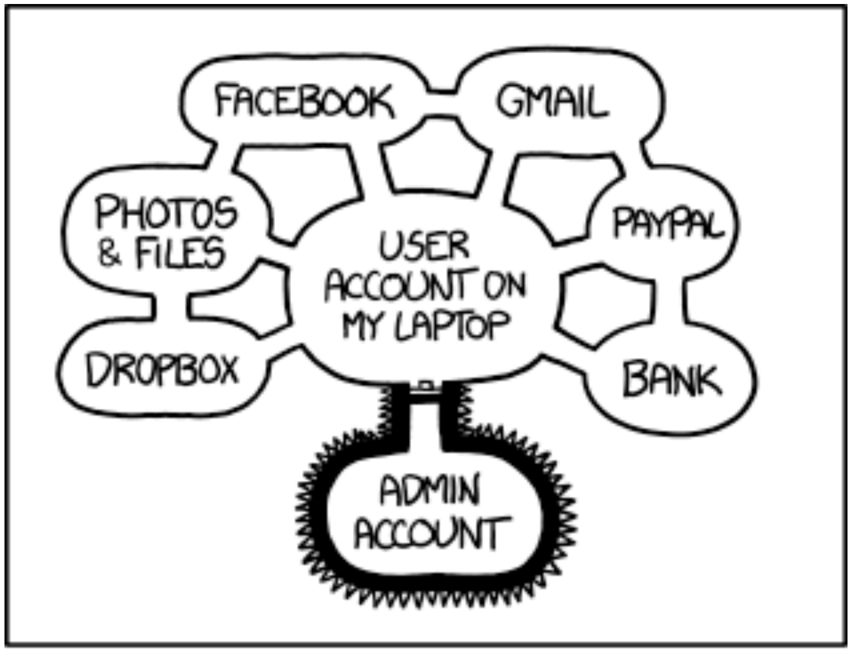
- Security as first-class citizen
  - *Early on security was an add-on, now it is everything*
- Security via depth
  - *Don’t assume a firewall will save you*
- Design for failure
  - *What happens after a breach occurs?*
- Secure the weakest link
  - *Anything but the crypto!*
- Obscurity is not security
  - *Keys in binary stand out like sore thumbs*
  - *Stored procedures are not a cure for access control*

Access Control

- **Authentication**: who are you
  - Typically username + **secret**
    - Something you know (password)
    - Something you have (smart card/phone)
    - Something you are (fingerprint, iris)

- **Authorization**: what can you do

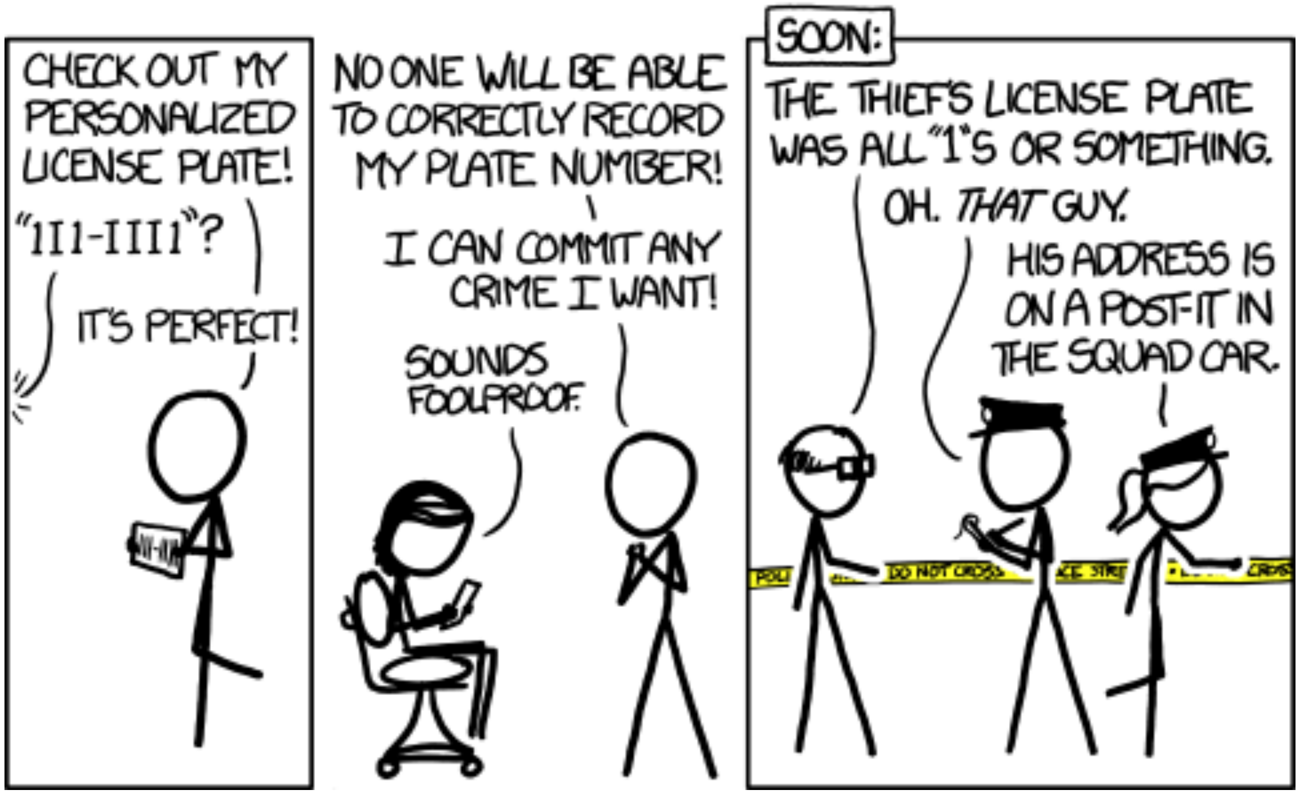
XKCD: Authorization



IF SOMEONE STEALS MY LAPTOP WHILE I'M LOGGED IN, THEY CAN READ MY EMAIL, TAKE MY MONEY, AND IMPERSONATE ME TO MY FRIENDS,

BUT AT LEAST THEY CAN'T INSTALL DRIVERS WITHOUT MY PERMISSION.

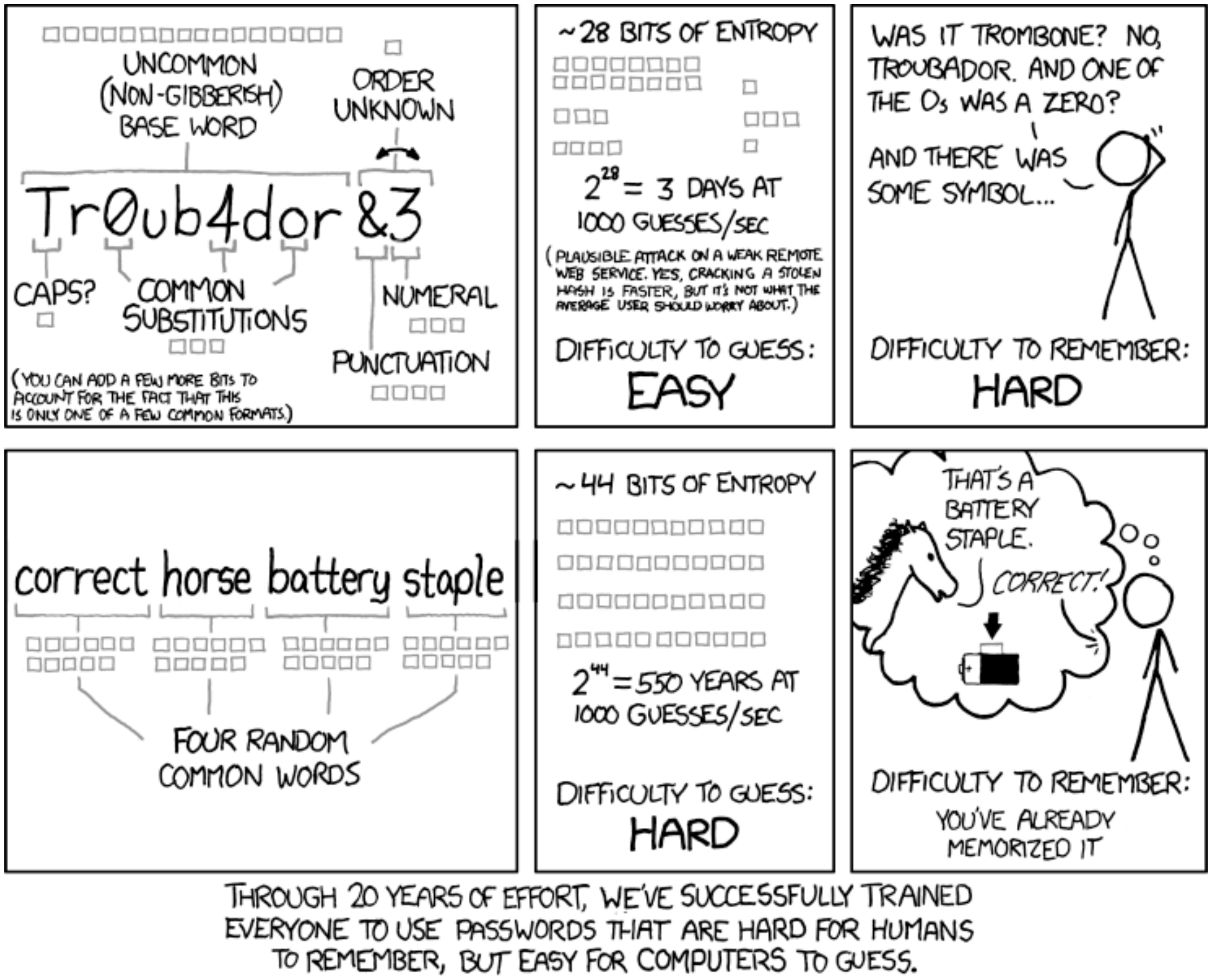
XCKD: License Plate


CHECK OUT MY PERSONALIZED LICENSE PLATE!
"1I1-III1"?
IT'S PERFECT!
NO ONE WILL BE ABLE TO CORRECTLY RECORD MY PLATE NUMBER!
I CAN COMMIT ANY CRIME I WANT!
SOUNDS FOOLPROOF.
SOON:
THE THIEF'S LICENSE PLATE WAS ALL "1"S OR SOMETHING.
OH. THAT GUY.
HIS ADDRESS IS ON A POST-IT IN THE SQUAD CAR.
DO NOT CROSS

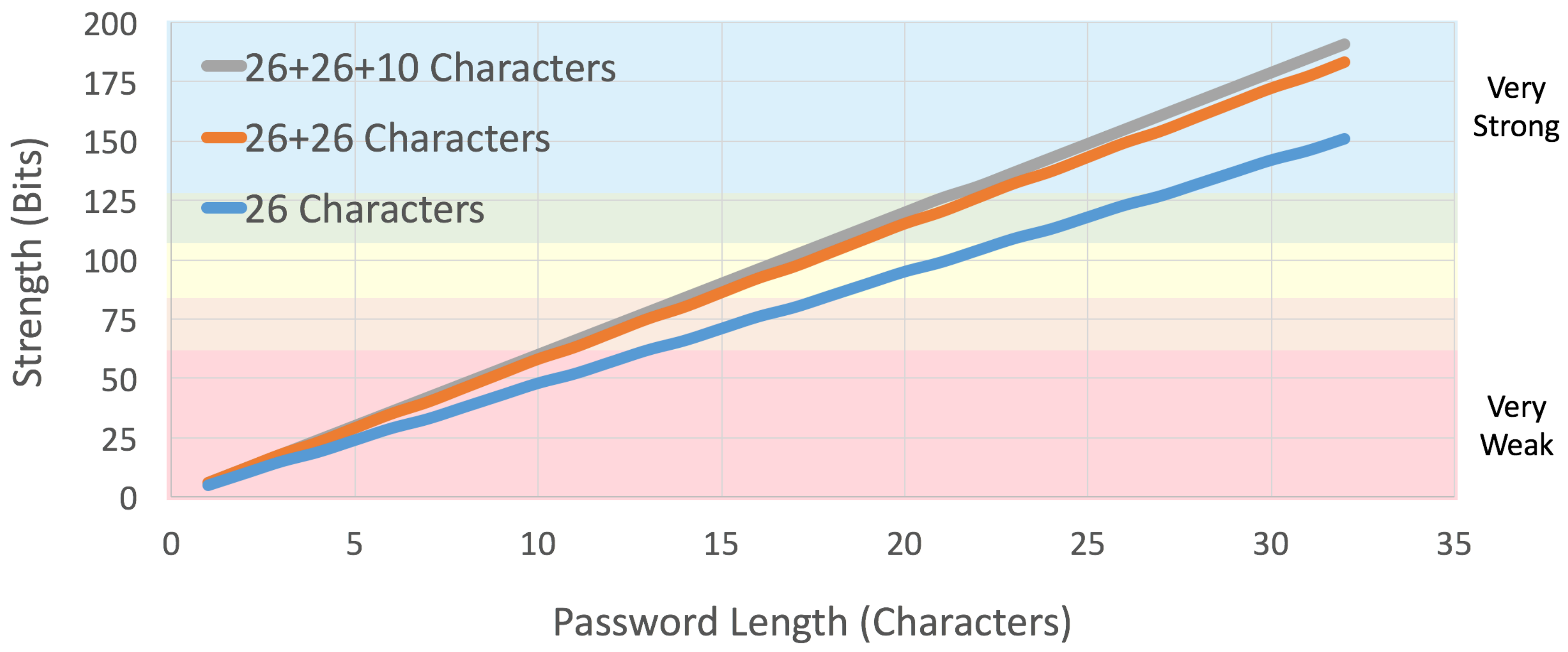
Authentication Policies

- Passwords
  - Enforce minimum length/complexity
    - Also maximum (more later w.r.t. DoS)
  - Require updates
  - Goal: make guessing/cracking difficult
    - Cross-service
- Attempts
  - Enforce limits to avoid brute force (iCloud)
- 2 Factor Authentication (2FA)
  - Often infeasible
  - Implementation may weaken
    - e.g. Social engineering

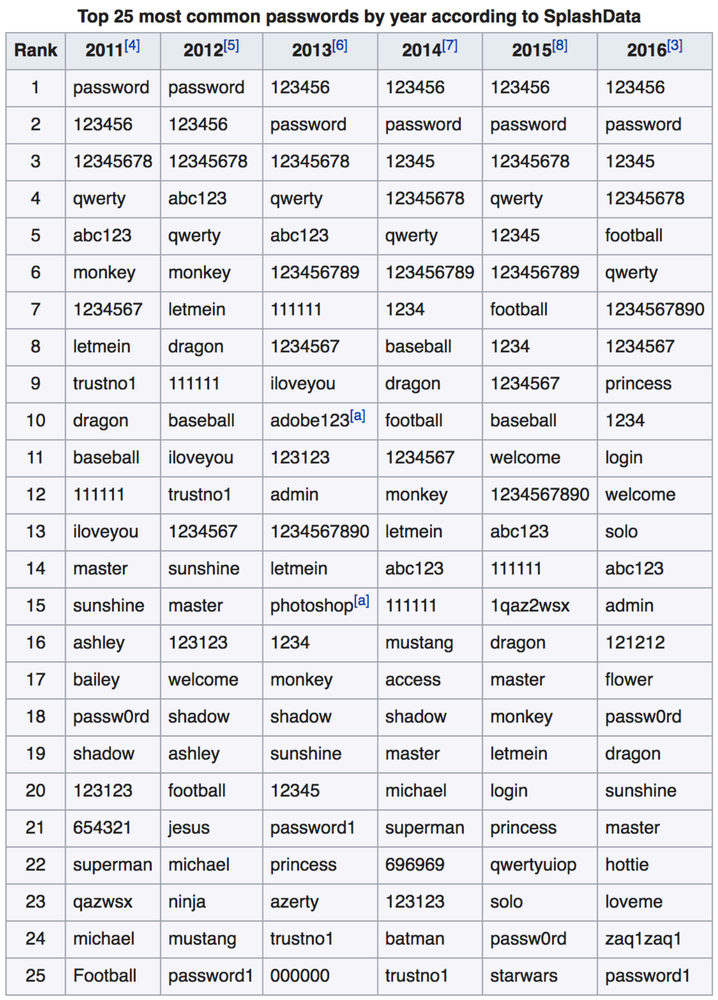
XKCD: Password Strength


UNCOMMON (NON-GIBBERISH) BASE WORD
ORDER UNKNOWN
Tr0ub4dor &3
CAPS?
COMMON SUBSTITUTIONS
NUMERAL
PUNCTUATION
(YOU CAN ADD A FEW MORE BITS TO ACCOUNT FOR THE FACT THAT THIS IS ONLY ONE OF A FEW COMMON FORMATS.)
~28 BITS OF ENTROPY
2^28 = 3 DAYS AT 1000 GUESSES/SEC
(PLAUSIBLE ATTACK ON A WEAK REMOTE WEB SERVICE. YES, CRACKING A STOLEN HASH IS FASTER, BUT IT'S NOT WHAT THE AVERAGE USER SHOULD WORRY ABOUT.)
DIFFICULTY TO GUESS: EASY
WAS IT TROMBONE? NO, TROUBADOR. AND ONE OF THE Os WAS A ZERO?
AND THERE WAS SOME SYMBOL...
DIFFICULTY TO REMEMBER: HARD
correct horse battery staple
FOUR RANDOM COMMON WORDS
~44 BITS OF ENTROPY
2^44 = 550 YEARS AT 1000 GUESSES/SEC
DIFFICULTY TO GUESS: HARD
THAT'S A BATTERY STAPLE.
CORRECT!
DIFFICULTY TO REMEMBER: YOU'VE ALREADY MEMORIZED IT
THROUGH 20 YEARS OF EFFORT, WE'VE SUCCESSFULLY TRAINED EVERYONE TO USE PASSWORDS THAT ARE HARD FOR HUMANS TO REMEMBER, BUT EASY FOR COMPUTERS TO GUESS.

Random Passwords



But Passwords Are Not Random



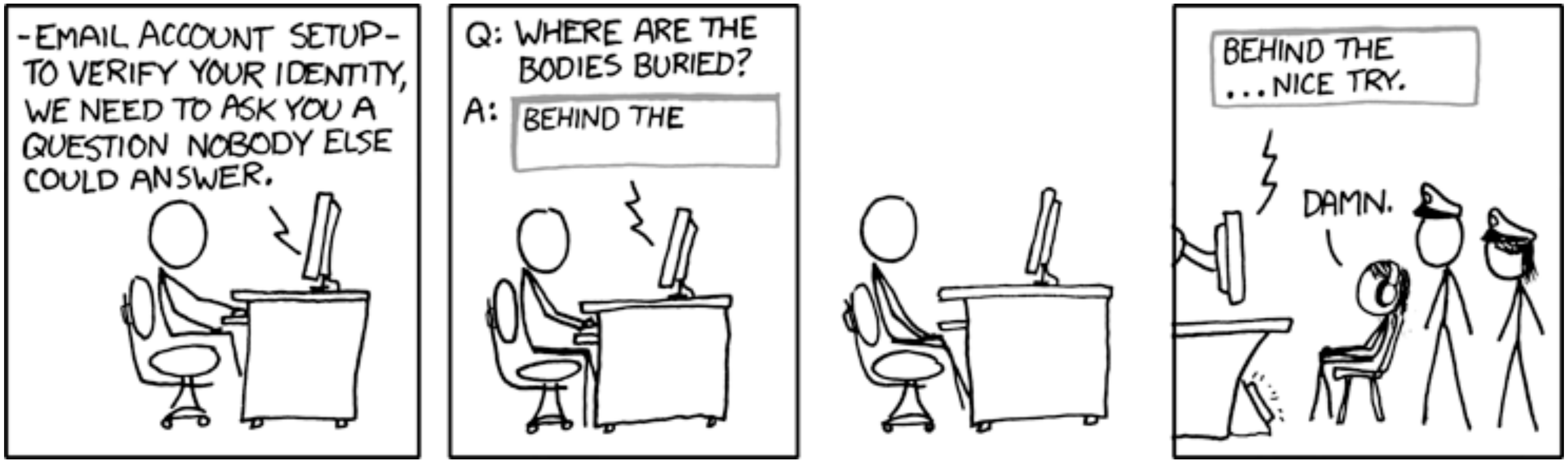
Top 25 most common passwords by year according to SplashData

Rank	2011[4]	2012[5]	2013[6]	2014[7]	2015[8]	2016[3]
1	password	password	123456	123456	123456	123456
2	123456	123456	password	password	password	password
3	12345678	12345678	12345678	12345	12345678	12345
4	qwerty	abc123	qwerty	12345678	qwerty	12345678
5	abc123	qwerty	abc123	qwerty	12345	football
6	monkey	monkey	123456789	123456789	123456789	qwerty
7	1234567	letmein	111111	1234	football	1234567890
8	letmein	dragon	1234567	baseball	1234	1234567
9	trustno1	111111	iloveyou	dragon	1234567	princess
10	dragon	baseball	adobe123[a]	football	baseball	1234
11	baseball	iloveyou	123123	1234567	welcome	login
12	111111	trustno1	admin	monkey	1234567890	welcome
13	iloveyou	1234567	1234567890	letmein	abc123	solo
14	master	sunshine	letmein	abc123	111111	abc123
15	sunshine	master	photoshop[a]	111111	1qaz2wsx	admin
16	ashley	123123	1234	mustang	dragon	121212
17	bailey	welcome	monkey	access	master	flower
18	passw0rd	shadow	shadow	shadow	monkey	passw0rd
19	shadow	ashley	sunshine	master	letmein	dragon
20	123123	football	12345	michael	login	sunshine
21	654321	jesus	password1	superman	princess	master
22	superman	michael	princess	696969	qwertyuiop	hottie
23	qazwsx	ninja	azerty	123123	solo	loveme
24	michael	mustang	trustno1	batman	passw0rd	zaq1zaq1
25	Football	password1	000000	trustno1	starwars	password1

Public Service Announcement

- Check: ';--have i been pwned? <**https://haveibeenpwned.com**>
  - User/e-mail
  - Services
  - Common passwords

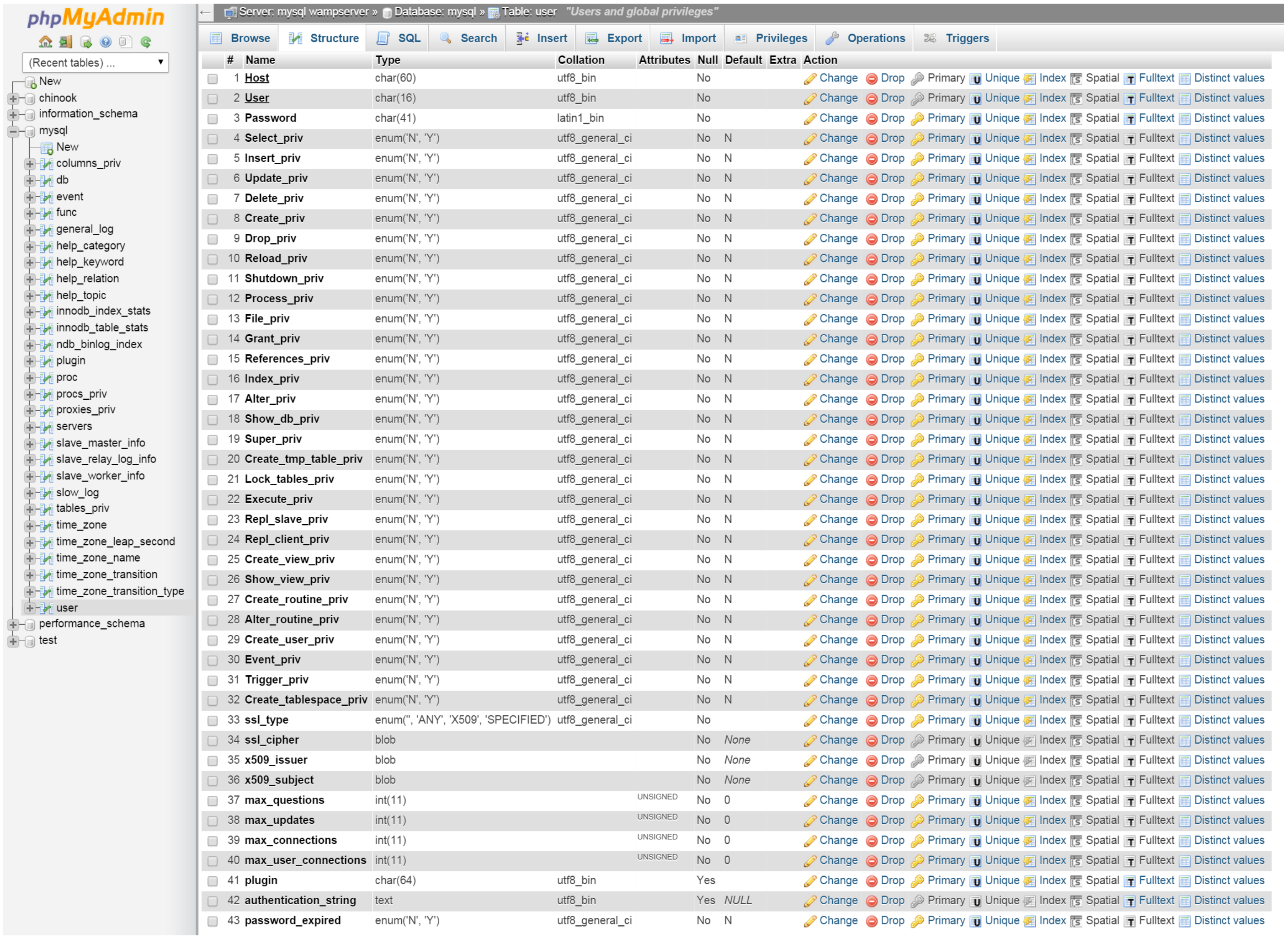
XKCD: Security Question


-EMAIL ACCOUNT SETUP-
TO VERIFY YOUR IDENTITY, WE NEED TO ASK YOU A QUESTION NOBODY ELSE COULD ANSWER.
Q: WHERE ARE THE BODIES BURIED?
A: BEHIND THE
BEHIND THE ...NICE TRY.
DAMN.

Discretionary Access Control

- Users **grant/revoke** privileges to other users
  - Starts with root/superuser/dba
  - with **`GRANT OPTION`**
- Privileges typically apply at multiple levels
  - Global, database, table, column
- Access matrix model
  - Users x Objects
- Fairly universal

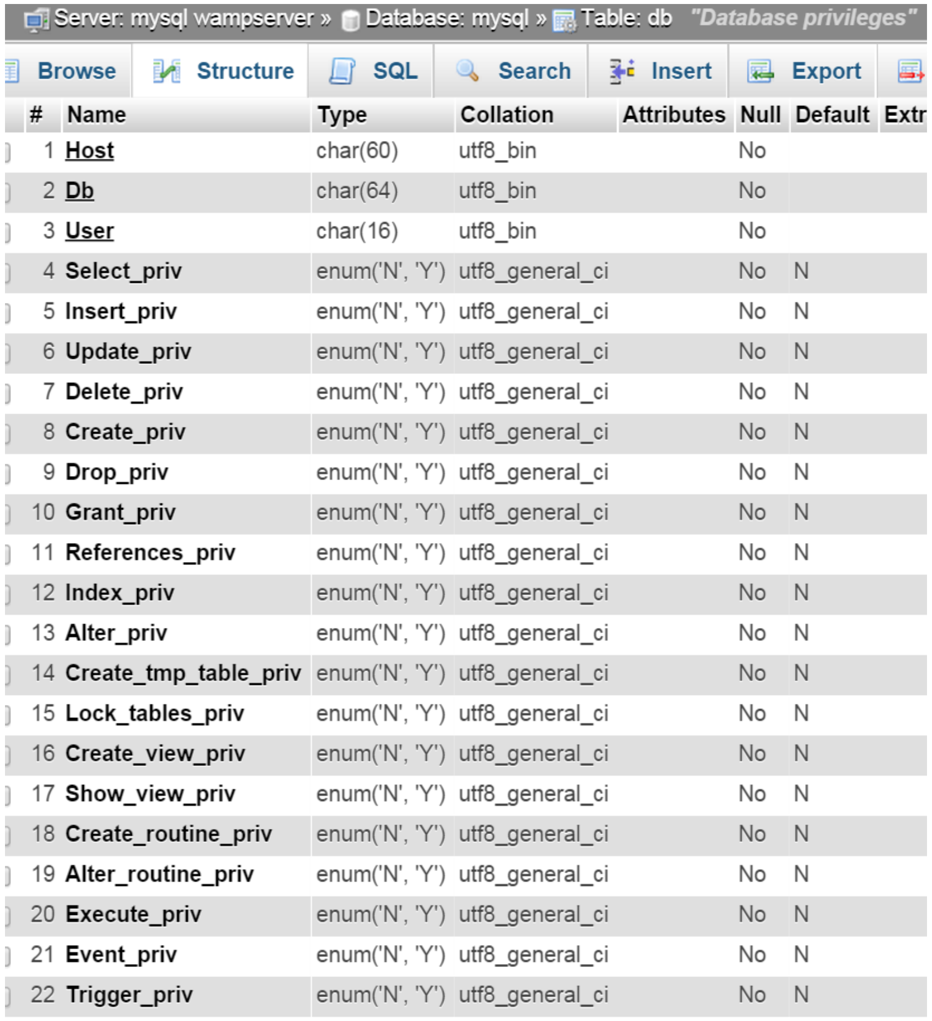
MySQL (user)



Server: mysql wampserver » Database: mysql » Table: user "Users and global privileges"

#	Name	Type	Collation	Attributes	Null	Default	Extra	Action
1	Host	char(60)	utf8_bin		No			Change Drop Primary Unique Index Spatial Fulltext Distinct values
2	User	char(16)	utf8_bin		No			Change Drop Primary Unique Index Spatial Fulltext Distinct values
3	Password	char(41)	latin1_bin		No			Change Drop Primary Unique Index Spatial Fulltext Distinct values
4	Select_priv	enum('N', 'Y')	utf8_general_ci		No	N		Change Drop Primary Unique Index Spatial Fulltext Distinct values
5	Insert_priv	enum('N', 'Y')	utf8_general_ci		No	N		Change Drop Primary Unique Index Spatial Fulltext Distinct values
6	Update_priv	enum('N', 'Y')	utf8_general_ci		No	N		Change Drop Primary Unique Index Spatial Fulltext Distinct values
7	Delete_priv	enum('N', 'Y')	utf8_general_ci		No	N		Change Drop Primary Unique Index Spatial Fulltext Distinct values
8	Create_priv	enum('N', 'Y')	utf8_general_ci		No	N		Change Drop Primary Unique Index Spatial Fulltext Distinct values
9	Drop_priv	enum('N', 'Y')	utf8_general_ci		No	N		Change Drop Primary Unique Index Spatial Fulltext Distinct values
10	Reload_priv	enum('N', 'Y')	utf8_general_ci		No	N		Change Drop Primary Unique Index Spatial Fulltext Distinct values
11	Shutdown_priv	enum('N', 'Y')	utf8_general_ci		No	N		Change Drop Primary Unique Index Spatial Fulltext Distinct values
12	Process_priv	enum('N', 'Y')	utf8_general_ci		No	N		Change Drop Primary Unique Index Spatial Fulltext Distinct values
13	File_priv	enum('N', 'Y')	utf8_general_ci		No	N		Change Drop Primary Unique Index Spatial Fulltext Distinct values
14	Grant_priv	enum('N', 'Y')	utf8_general_ci		No	N		Change Drop Primary Unique Index Spatial Fulltext Distinct values
15	References_priv	enum('N', 'Y')	utf8_general_ci		No	N		Change Drop Primary Unique Index Spatial Fulltext Distinct values
16	Index_priv	enum('N', 'Y')	utf8_general_ci		No	N		Change Drop Primary Unique Index Spatial Fulltext Distinct values
17	Alter_priv	enum('N', 'Y')	utf8_general_ci		No	N		Change Drop Primary Unique Index Spatial Fulltext Distinct values
18	Show_db_priv	enum('N', 'Y')	utf8_general_ci		No	N		Change Drop Primary Unique Index Spatial Fulltext Distinct values
19	Super_priv	enum('N', 'Y')	utf8_general_ci		No	N		Change Drop Primary Unique Index Spatial Fulltext Distinct values
20	Create_tmp_table_priv	enum('N', 'Y')	utf8_general_ci		No	N		Change Drop Primary Unique Index Spatial Fulltext Distinct values
21	Lock_tables_priv	enum('N', 'Y')	utf8_general_ci		No	N		Change Drop Primary Unique Index Spatial Fulltext Distinct values
22	Execute_priv	enum('N', 'Y')	utf8_general_ci		No	N		Change Drop Primary Unique Index Spatial Fulltext Distinct values
23	Repl_slave_priv	enum('N', 'Y')	utf8_general_ci		No	N		Change Drop Primary Unique Index Spatial Fulltext Distinct values
24	Repl_client_priv	enum('N', 'Y')	utf8_general_ci		No	N		Change Drop Primary Unique Index Spatial Fulltext Distinct values
25	Create_view_priv	enum('N', 'Y')	utf8_general_ci		No	N		Change Drop Primary Unique Index Spatial Fulltext Distinct values
26	Show_view_priv	enum('N', 'Y')	utf8_general_ci		No	N		Change Drop Primary Unique Index Spatial Fulltext Distinct values
27	Create_routine_priv	enum('N', 'Y')	utf8_general_ci		No	N		Change Drop Primary Unique Index Spatial Fulltext Distinct values
28	Alter_routine_priv	enum('N', 'Y')	utf8_general_ci		No	N		Change Drop Primary Unique Index Spatial Fulltext Distinct values
29	Create_user_priv	enum('N', 'Y')	utf8_general_ci		No	N		Change Drop Primary Unique Index Spatial Fulltext Distinct values
30	Event_priv	enum('N', 'Y')	utf8_general_ci		No	N		Change Drop Primary Unique Index Spatial Fulltext Distinct values
31	Trigger_priv	enum('N', 'Y')	utf8_general_ci		No	N		Change Drop Primary Unique Index Spatial Fulltext Distinct values
32	Create_tablespace_priv	enum('N', 'Y')	utf8_general_ci		No	N		Change Drop Primary Unique Index Spatial Fulltext Distinct values
33	ssl_type	enum('', 'ANY', 'X509', 'SPECIFIED')	utf8_general_ci		No			Change Drop Primary Unique Index Spatial Fulltext Distinct values
34	ssl_cipher	blob			No	*None*		Change Drop Primary Unique Index Spatial Fulltext Distinct values
35	x509_issuer	blob			No	*None*		Change Drop Primary Unique Index Spatial Fulltext Distinct values
36	x509_subject	blob			No	*None*		Change Drop Primary Unique Index Spatial Fulltext Distinct values
37	max_questions	int(11)		UNSIGNED	No	0		Change Drop Primary Unique Index Spatial Fulltext Distinct values
38	max_updates	int(11)		UNSIGNED	No	0		Change Drop Primary Unique Index Spatial Fulltext Distinct values
39	max_connections	int(11)		UNSIGNED	No	0		Change Drop Primary Unique Index Spatial Fulltext Distinct values
40	max_user_connections	int(11)		UNSIGNED	No	0		Change Drop Primary Unique Index Spatial Fulltext Distinct values
41	plugin	char(64)	utf8_bin		Yes			Change Drop Primary Unique Index Spatial Fulltext Distinct values
42	authentication_string	text	utf8_bin		Yes	*NULL*		Change Drop Primary Unique Index Spatial Fulltext Distinct values
43	password_expired	enum('N', 'Y')	utf8_general_ci		No	N		Change Drop Primary Unique Index Spatial Fulltext Distinct values

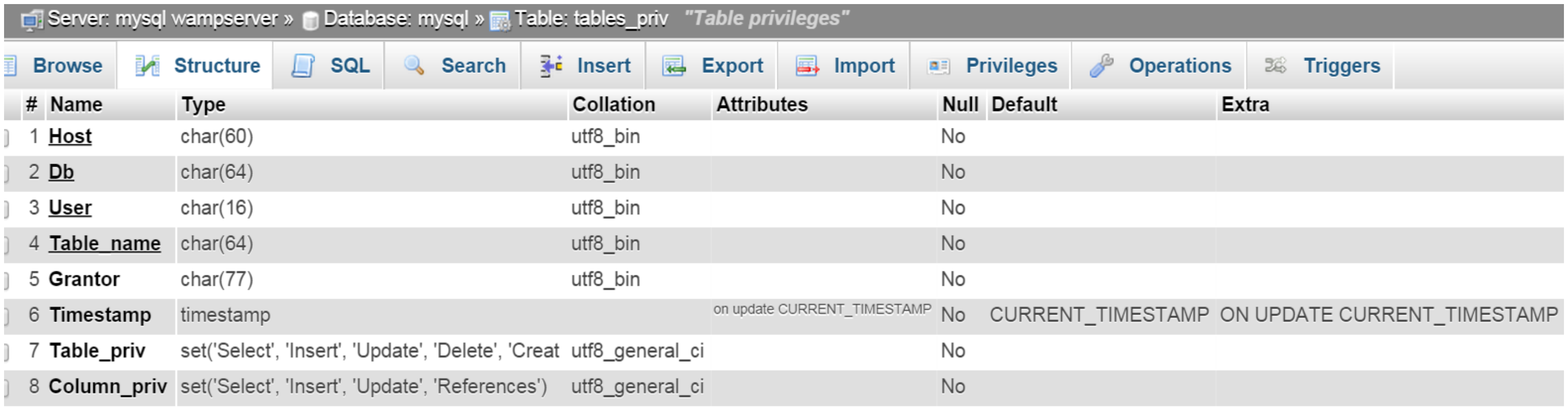
MySQL (db)



Server: mysql wampserver » Database: mysql » Table: db "Database privileges"

#	Name	Type	Collation	Attributes	Null	Default	Extr
1	Host	char(60)	utf8_bin		No		
2	Db	char(64)	utf8_bin		No		
3	User	char(16)	utf8_bin		No		
4	Select_priv	enum('N', 'Y')	utf8_general_ci		No	N	
5	Insert_priv	enum('N', 'Y')	utf8_general_ci		No	N	
6	Update_priv	enum('N', 'Y')	utf8_general_ci		No	N	
7	Delete_priv	enum('N', 'Y')	utf8_general_ci		No	N	
8	Create_priv	enum('N', 'Y')	utf8_general_ci		No	N	
9	Drop_priv	enum('N', 'Y')	utf8_general_ci		No	N	
10	Grant_priv	enum('N', 'Y')	utf8_general_ci		No	N	
11	References_priv	enum('N', 'Y')	utf8_general_ci		No	N	
12	Index_priv	enum('N', 'Y')	utf8_general_ci		No	N	
13	Alter_priv	enum('N', 'Y')	utf8_general_ci		No	N	
14	Create_tmp_table_priv	enum('N', 'Y')	utf8_general_ci		No	N	
15	Lock_tables_priv	enum('N', 'Y')	utf8_general_ci		No	N	
16	Create_view_priv	enum('N', 'Y')	utf8_general_ci		No	N	
17	Show_view_priv	enum('N', 'Y')	utf8_general_ci		No	N	
18	Create_routine_priv	enum('N', 'Y')	utf8_general_ci		No	N	
19	Alter_routine_priv	enum('N', 'Y')	utf8_general_ci		No	N	
20	Execute_priv	enum('N', 'Y')	utf8_general_ci		No	N	
21	Event_priv	enum('N', 'Y')	utf8_general_ci		No	N	
22	Trigger_priv	enum('N', 'Y')	utf8_general_ci		No	N	

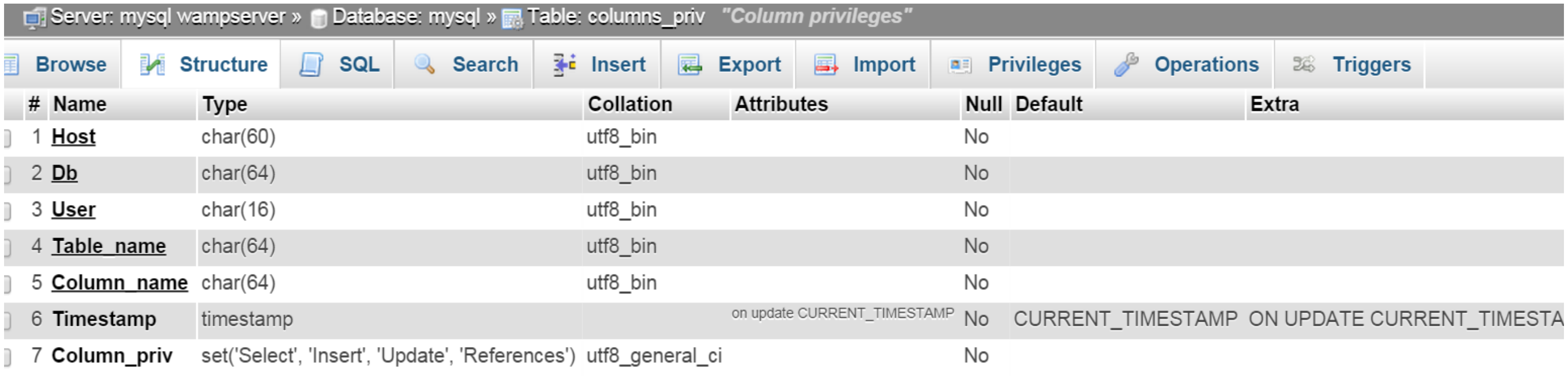
MySQL (tables_priv)



Server: mysql wampserver » Database: mysql » Table: tables_priv *"Table privileges"*

#	Name	Type	Collation	Attributes	Null	Default	Extra
1	Host	char(60)	utf8_bin		No		
2	Db	char(64)	utf8_bin		No		
3	User	char(16)	utf8_bin		No		
4	Table_name	char(64)	utf8_bin		No		
5	Grantor	char(77)	utf8_bin		No		
6	Timestamp	timestamp		on update CURRENT_TIMESTAMP	No	CURRENT_TIMESTAMP	ON UPDATE CURRENT_TIMESTAMP
7	Table_priv	set('Select', 'Insert', 'Update', 'Delete', 'Creat	utf8_general_ci		No		
8	Column_priv	set('Select', 'Insert', 'Update', 'References')	utf8_general_ci		No		

MySQL (columns_priv)



Server: mysql wampserver » Database: mysql » Table: columns_priv *"Column privileges"*

#	Name	Type	Collation	Attributes	Null	Default	Extra
1	Host	char(60)	utf8_bin		No		
2	Db	char(64)	utf8_bin		No		
3	User	char(16)	utf8_bin		No		
4	Table_name	char(64)	utf8_bin		No		
5	Column_name	char(64)	utf8_bin		No		
6	Timestamp	timestamp		on update CURRENT_TIMESTAMP	No	CURRENT_TIMESTAMP	ON UPDATE CURRENT_TIMESTA
7	Column_priv	set('Select', 'Insert', 'Update', 'References')	utf8_general_ci		No		

Mandatory Access Control

- Objects are classified with security levels
- Users are afforded security clearance
- Government model, not typically supported

Privilege Policies

- Principle of least privilege
- Privilege separation
  - Multiple users, each with least privilege
- Abuse
  - Unauthorized
    - Mitigate escalation attacks
  - Authorized
    - Teachers changing grades
    - Firing a DBA

SQL Injection

SQL manipulation for nefarious purpose

<u>Method</u>

- String manipulation
  - Parameters, function calls
- Code injection (e.g. buffer overflow)

<u>Goals</u>

- Fingerprinting
  - Learn about service via version, configuration
- DoS
- Bypass authentication/privilege escalation
- Remote execution

<u>Protection</u>

- Parameterized statements
- Filter input
- Limit use of custom functions

SQL Injection Examples

Original query:

```
    “SELECT name, description
     FROM items
     WHERE id=‘” + req.args.get(‘id’, ‘’) + “’”
```

Result after injection:

```
    SELECT name, description
    FROM items
    WHERE id='12’
UNION
    SELECT username, passwd FROM users;--';
```

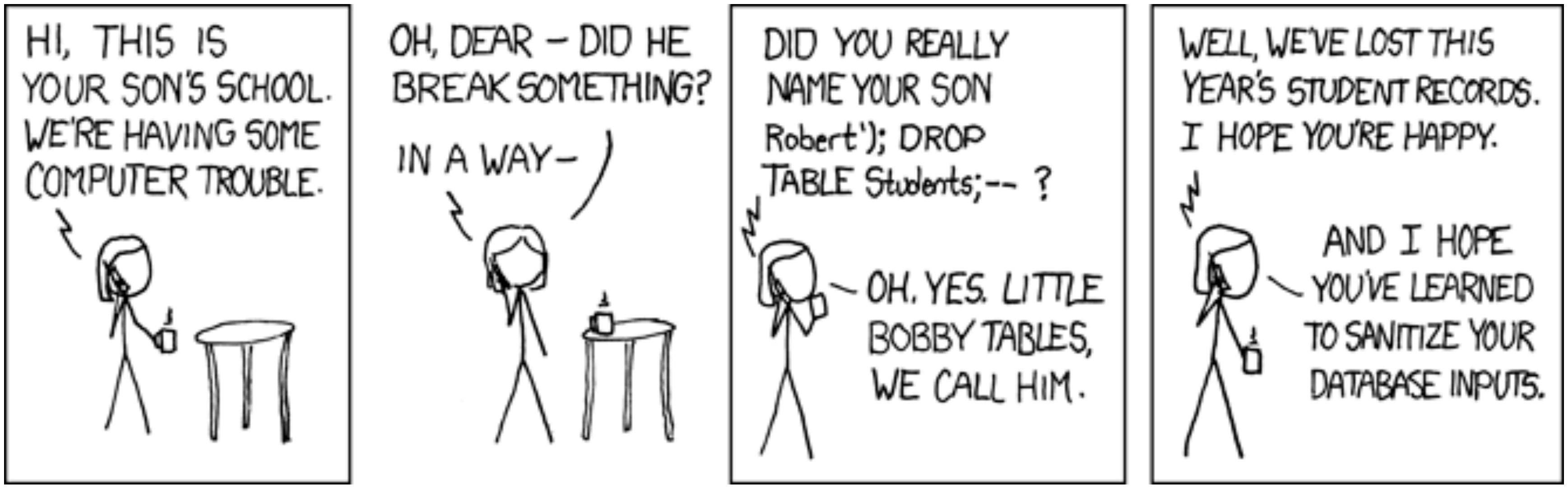
Original query:

```
    “UPDATE users
     SET passwd=‘” + req.args.get(‘pw’, ‘’) + “’
     WHERE user=‘” + req.args.get(‘user’, ‘’) + “‘”
```

Result after injection:

```
    UPDATE users
    SET passwd='...'
    WHERE user='dude' OR 1=1;--';
```

XKCD: Exploits of a Mom


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.

Denial of Service (DoS)

Any exposed interface
- Failed login
  - Lock out users
  - Resource utilization via long password verification
- Complex queries

Mitigation
- Resource limits
- Patching
- Monitoring

XCKD: CIA

HACKERS BRIEFLY TOOK DOWN THE WEBSITE OF THE CIA YESTERDAY...
WHAT PEOPLE HEAR:
SOMEONE HACKED INTO THE COMPUTERS OF THE CIA!!
WHAT COMPUTER EXPERTS HEAR:
SOMEONE TORE DOWN A POSTER HUNG UP BY THE CIA!!

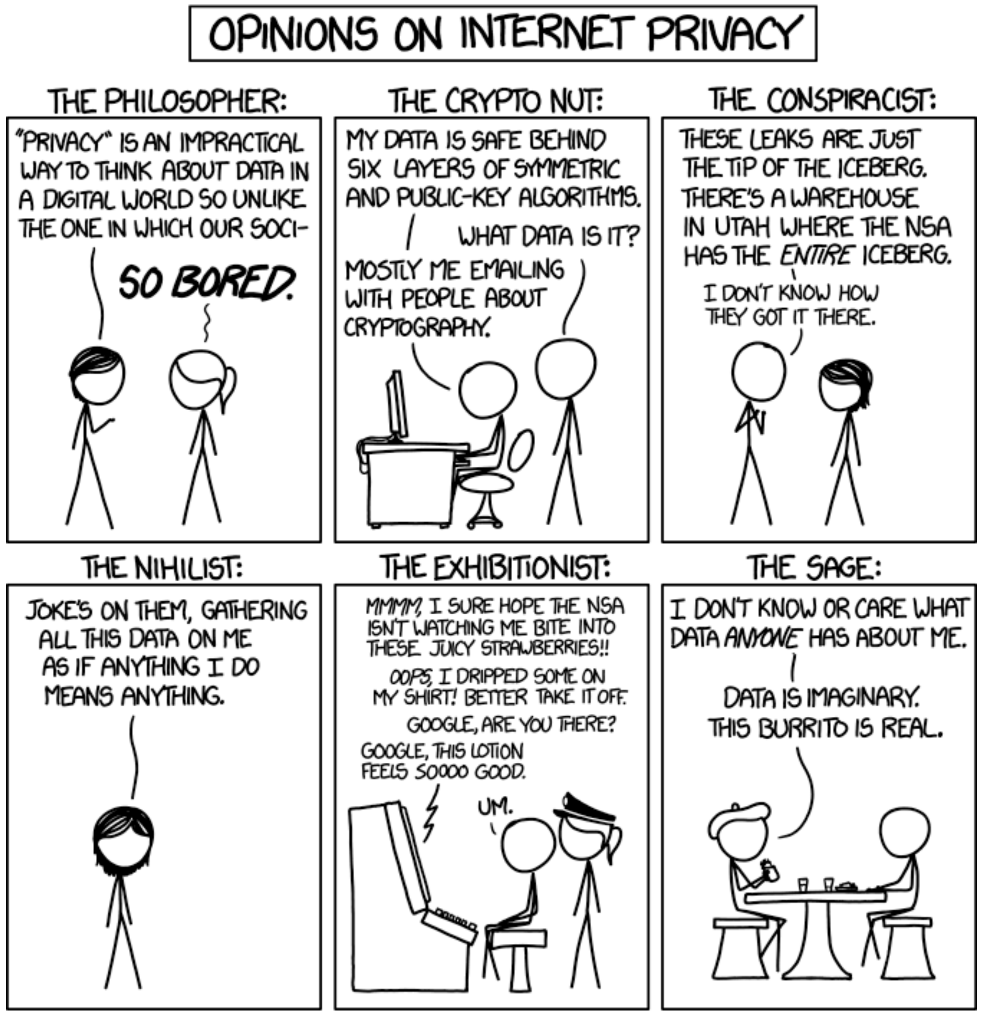
Protection

- Protect against internal attacks
  - Oracle: up to 80% of data loss
- Isolate DBMS
  - Separate machine, VM
- Regular patching policies
- Audit logs

Inferential Security

- Relevant when offering parameterized access to aggregate data
  - But must protect sensitive individual data!
- Prior knowledge and/or clever exploration might yield queries that reveal private information
  - Find "average" salary of <insert conditions that identify single individual>
- Techniques
  - Minimum result set size threshold
  - Added noise
  - Group partitioning

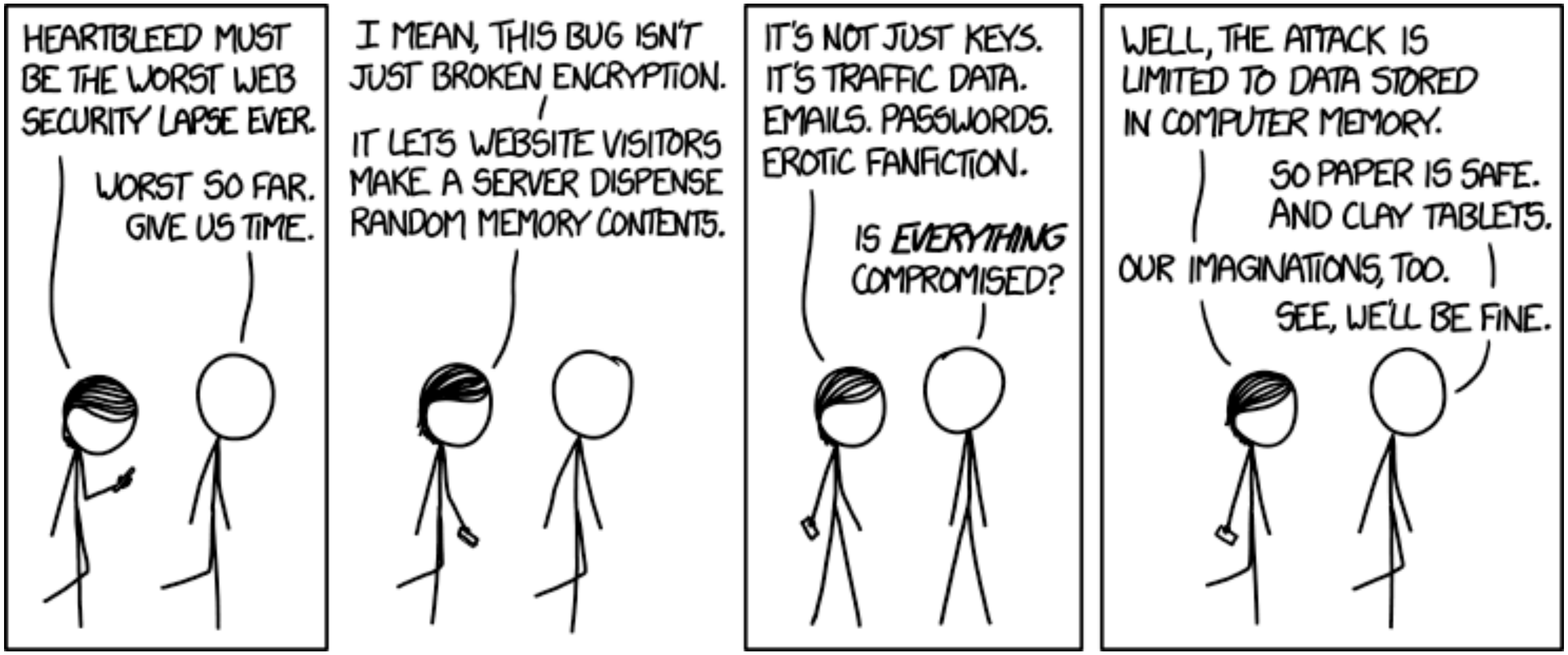
XKCD: Privacy Opinions


OPINIONS ON INTERNET PRIVACY
THE PHILOSOPHER:
"PRIVACY" IS AN IMPRACTICAL WAY TO THINK ABOUT DATA IN A DIGITAL WORLD SO UNLIKE THE ONE IN WHICH OUR SOCI-
SO BORED.
THE CRYPTO NUT:
MY DATA IS SAFE BEHIND SIX LAYERS OF SYMMETRIC AND PUBLIC-KEY ALGORITHMS.
WHAT DATA IS IT?
MOSTLY ME EMAILING WITH PEOPLE ABOUT CRYPTOGRAPHY.
THE CONSPIRACIST:
THESE LEAKS ARE JUST THE TIP OF THE ICEBERG. THERE'S A WAREHOUSE IN UTAH WHERE THE NSA HAS THE ENTIRE ICEBERG.
I DON'T KNOW HOW THEY GOT IT THERE.
THE NIHILIST:
JOKE'S ON THEM, GATHERING ALL THIS DATA ON ME AS IF ANYTHING I DO MEANS ANYTHING.
THE EXHIBITIONIST:
MMMM, I SURE HOPE THE NSA ISN'T WATCHING ME BITE INTO THESE JUICY STRAWBERRIES!!
OOPS, I DRIPPED SOME ON MY SHIRT! BETTER TAKE IT OFF.
GOOGLE, ARE YOU THERE?
GOOGLE, THIS LOTION FEELS SOOOO GOOD.
UM.
THE SAGE:
I DON'T KNOW OR CARE WHAT DATA ANYONE HAS ABOUT ME.
DATA IS IMAGINARY. THIS BURRITO IS REAL.

Encryption

- Symmetric
  – Single key encrypts/decrypts
- Asymmetric
  – 2 Keys: public encryption, private decryption
- Hashing
  – No decryption
- Encryption theory is solid, implementation is tricky
  – High-quality randomness
  – Bug-free code

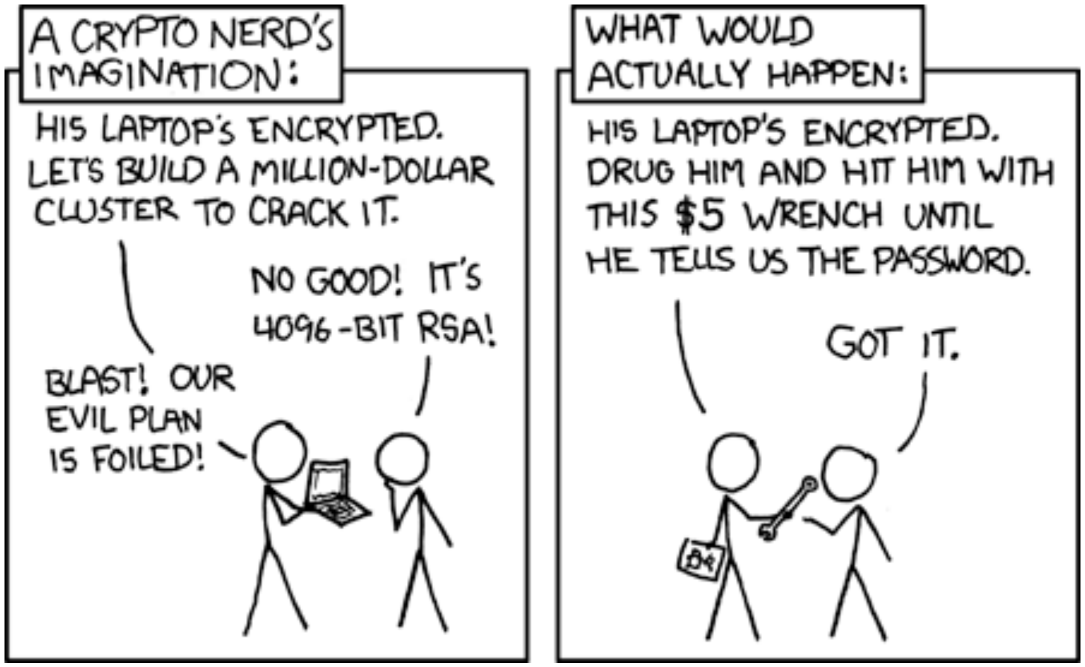
XCKD: Heartbleed


HEARTBLEED MUST BE THE WORST WEB SECURITY LAPSE EVER.
WORST SO FAR. GIVE US TIME.
I MEAN, THIS BUG ISN'T JUST BROKEN ENCRYPTION.
IT LETS WEBSITE VISITORS MAKE A SERVER DISPENSE RANDOM MEMORY CONTENTS.
IT'S NOT JUST KEYS. IT'S TRAFFIC DATA. EMAILS. PASSWORDS. EROTIC FANFICTION.
IS *EVERYTHING* COMPROMISED?
WELL, THE ATTACK IS LIMITED TO DATA STORED IN COMPUTER MEMORY.
SO PAPER IS SAFE. AND CLAY TABLETS.
OUR IMAGINATIONS, TOO.
SEE, WE'LL BE FINE.

Basics

- Encrypt database files
  - Including backups!
  - Native or 3rd-party wrapper
  - Can be difficult to implement while being resilient to restarts, high-performance
- Encrypt application communication
  - Use https, SSH
  - NOT http, telnet/FTP

XCKD: Security


A CRYPTO NERD'S IMAGINATION:
HIS LAPTOP'S ENCRYPTED. LET'S BUILD A MILLION-DOLLAR CLUSTER TO CRACK IT.
NO GOOD! IT'S 4096-BIT RSA!
BLAST! OUR EVIL PLAN IS FOILED!
WHAT WOULD ACTUALLY HAPPEN:
HIS LAPTOP'S ENCRYPTED. DRUG HIM AND HIT HIM WITH THIS $5 WRENCH UNTIL HE TELLS US THE PASSWORD.
GOT IT.

Sensitive Data

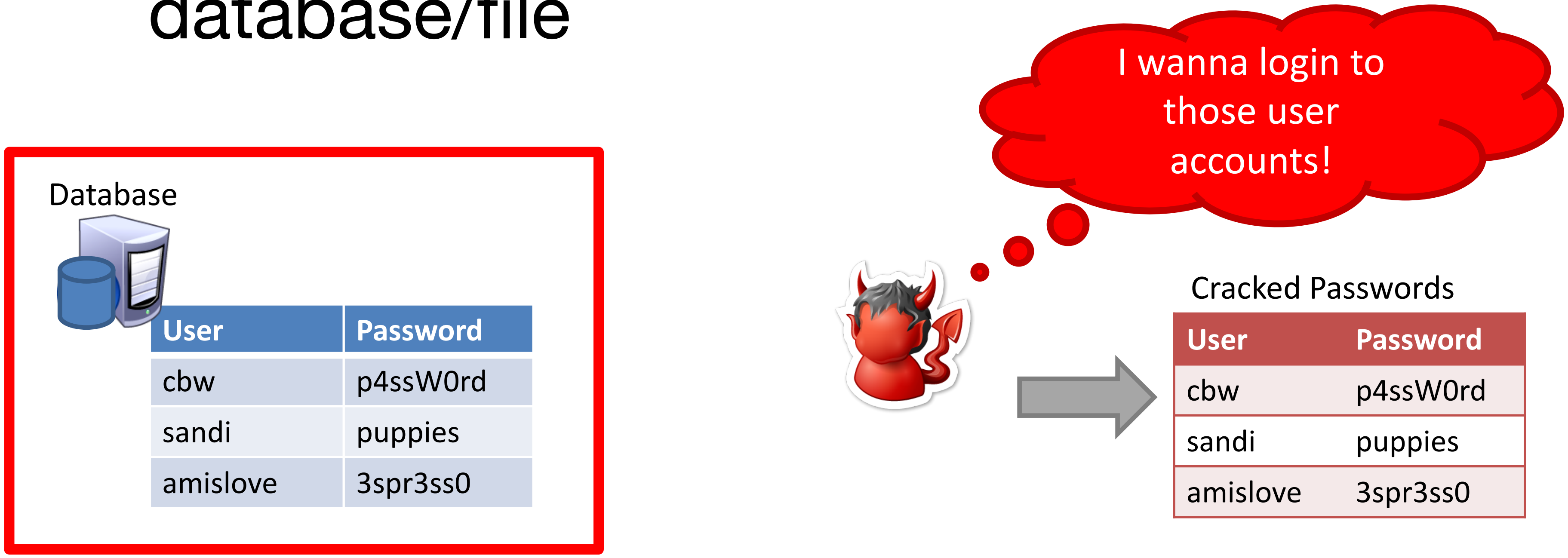
- When dealing with sensitive data, always consider how it needs to be used
- If only verification (e.g. password), hash
- If usage, encrypt
  - NOT clear text CC entry
  - Better: encrypt CC
  - Best: encrypt last 4 of CC + use private payment processing server

Password Storage

- Many applications require authentication
  - Website, mobile
- Sometimes you can use external authentication
  - LDAP, OAuth 2.0 via Google or Facebook
- Sometimes you need your own system
  - So now we consider how to securely store authentication secrets in a database

Attacker Goals and Threat Model

- Assume we have a system storing usernames and passwords
- The attacker has access to the password database/file



User	Password
cbw	p4ssW0rd
sandi	puppies
amislove	3spr3ss0

User	Password
cbw	p4ssW0rd
sandi	puppies
amislove	3spr3ss0

Checking Passwords

- System must validate passwords provided by users
- Thus, passwords must be stored somewhere
- Basic storage: plain text

password.txt

cbw	p4ssw0rd
sandi	i heart doggies
amislove	93Gd9#jv*0x3N
bob	security

Problem: Password File Theft

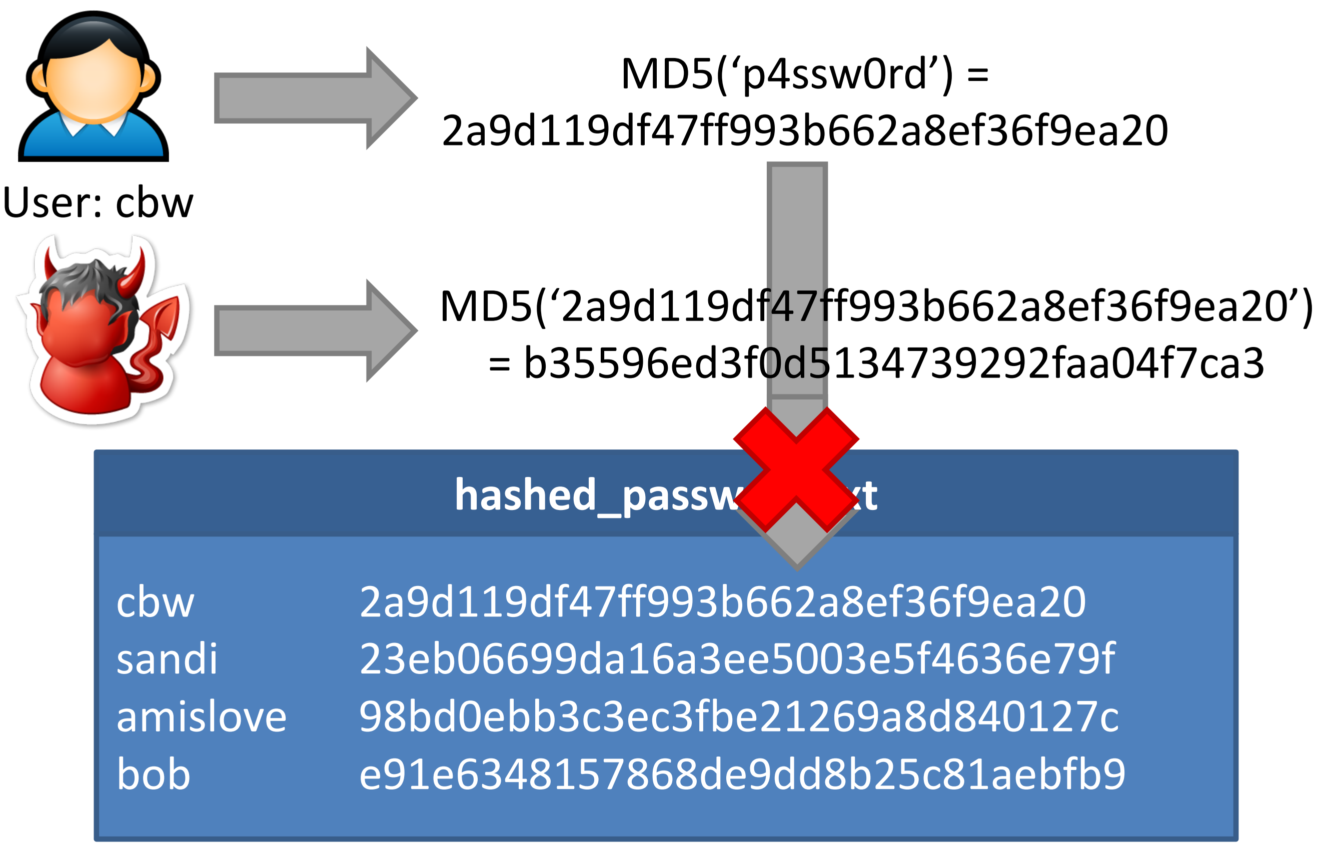
- Attackers often compromise systems
- They may be able to steal the password file
  - Linux: /etc/shadow
  - Windows: c:\windows\system32\config\sam
- If the passwords are plain text, what happens?
  - The attacker can now log-in as any user, including root/administrator
  - The attacker can/will use them elsewhere >:(
- **Passwords should never be stored in plain text**

Hashed Passwords

- Key idea: store encrypted versions of passwords
  - Use one-way cryptographic hash functions
  - Examples: MD5, SHA1, SHA256, SHA512, bcrypt, PBKDF2, scrypt

- Cryptographic hash function transform input data into scrambled output data
  - Deterministic: hash(A) = hash(A)
  - High entropy:
    - MD5('security') = e91e6348157868de9dd8b25c81aebfb9
    - MD5('security1') = 8632c375e9eba096df51844a5a43ae93
    - MD5('Security') = 2fae32629d4ef4fc6341f1751b405e45
  - Collision resistant
    - Locating A' such that hash(A) = hash(A') takes a long time (hopefully)
    - Example: 221 tries for md5

Hashed Password Example



User: cbw

MD5(‘p4ssw0rd’) = 2a9d119df47ff993b662a8ef36f9ea20

MD5(‘2a9d119df47ff993b662a8ef36f9ea20’) = b35596ed3f0d5134739292faa04f7ca3

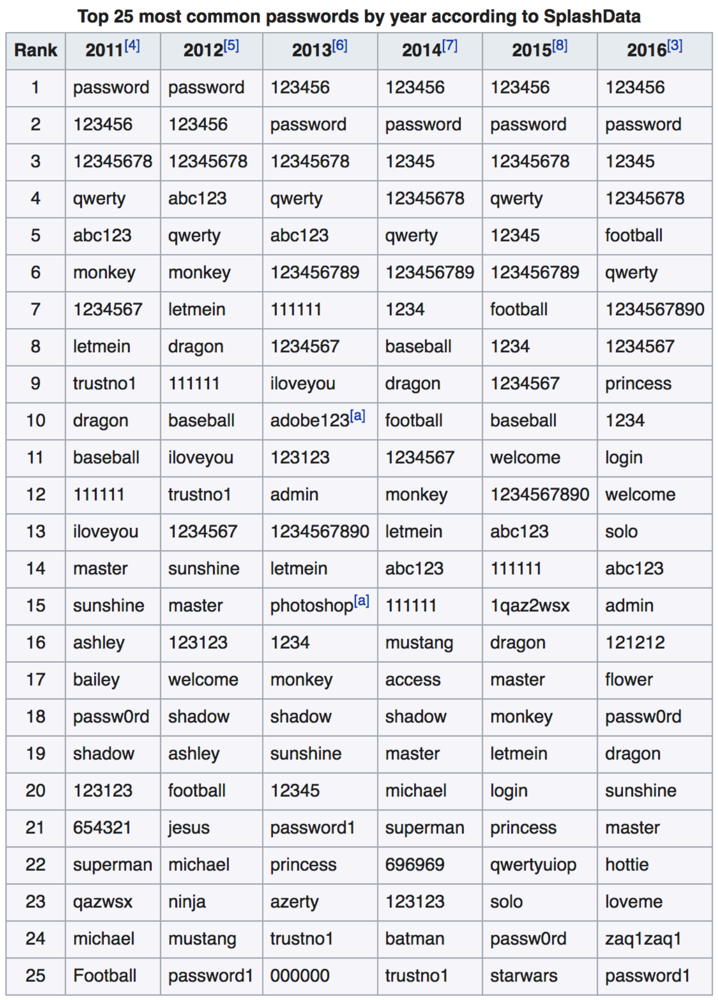
hashed_passw xt

cbw	2a9d119df47ff993b662a8ef36f9ea20
sandi	23eb06699da16a3ee5003e5f4636e79f
amislove	98bd0ebb3c3ec3fbe21269a8d840127c
bob	e91e6348157868de9dd8b25c81aebfb9

Attacking Password Hashes

- Recall: cryptographic hashes are collision resistant
  - Locating A’ such that hash(A) = hash(A’) takes a long time (hopefully)
- Are hashed password secure from cracking?
  - No!
- Problem: users choose poor passwords
  - Most common passwords: 123456, password
  - Username: cbw, Password: cbw
- Weak passwords enable dictionary attacks

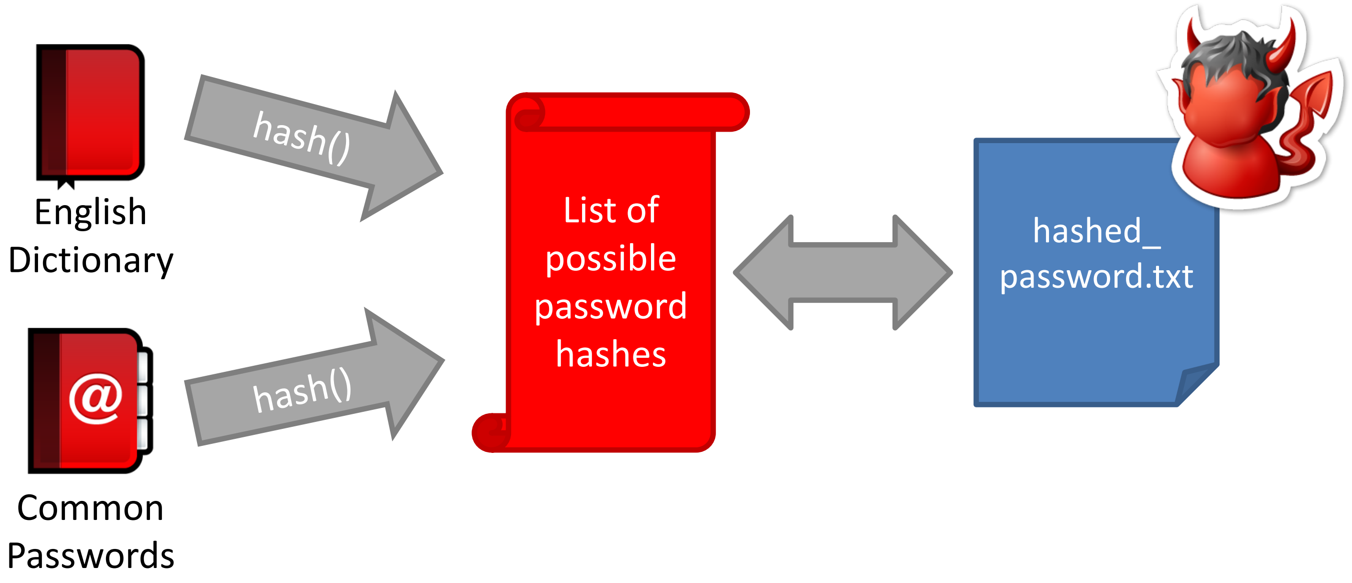
Remember: Passwords Are Not Random



Top 25 most common passwords by year according to SplashData

Rank	2011[4]	2012[5]	2013[6]	2014[7]	2015[8]	2016[3]
1	password	password	123456	123456	123456	123456
2	123456	123456	password	password	password	password
3	12345678	12345678	12345678	12345	12345678	12345
4	qwerty	abc123	qwerty	12345678	qwerty	12345678
5	abc123	qwerty	abc123	qwerty	12345	football
6	monkey	monkey	123456789	123456789	123456789	qwerty
7	1234567	letmein	111111	1234	football	1234567890
8	letmein	dragon	1234567	baseball	1234	1234567
9	trustno1	111111	iloveyou	dragon	1234567	princess
10	dragon	baseball	adobe123[a]	football	baseball	1234
11	baseball	iloveyou	123123	1234567	welcome	login
12	111111	trustno1	admin	monkey	1234567890	welcome
13	iloveyou	1234567	1234567890	letmein	abc123	solo
14	master	sunshine	letmein	abc123	111111	abc123
15	sunshine	master	photoshop[a]	111111	1qaz2wsx	admin
16	ashley	123123	1234	mustang	dragon	121212
17	bailey	welcome	monkey	access	master	flower
18	passw0rd	shadow	shadow	shadow	monkey	passw0rd
19	shadow	ashley	sunshine	master	letmein	dragon
20	123123	football	12345	michael	login	sunshine
21	654321	jesus	password1	superman	princess	master
22	superman	michael	princess	696969	qwertyuiop	hottie
23	qazwsx	ninja	azerty	123123	solo	loveme
24	michael	mustang	trustno1	batman	passw0rd	zaq1zaq1
25	Football	password1	000000	trustno1	starwars	password1

Dictionary Attacks



- Common for 60-70% of hashed passwords to be cracked in <24 hours

Hardening Password Hashes

- Key problem: cryptographic hashes are deterministic
  - hash('p4ssw0rd') = hash('p4ssw0rd')
  - This enables attackers to build lists of hashes

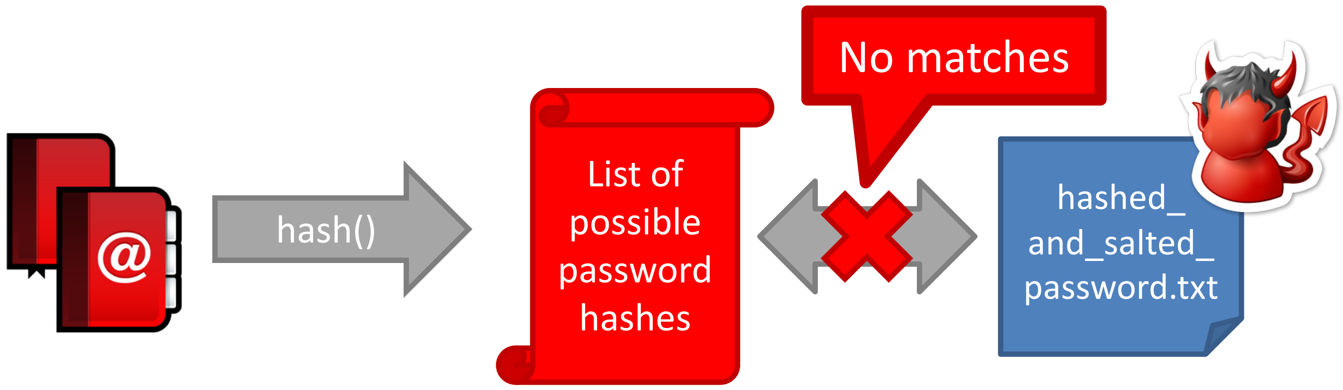
- Solution: make each password hash unique
  - Add a salt to each password before hashing
  - hash(salt + password) = password hash
  - Each user has a unique, random salt
  - Salts can be stores in plain text

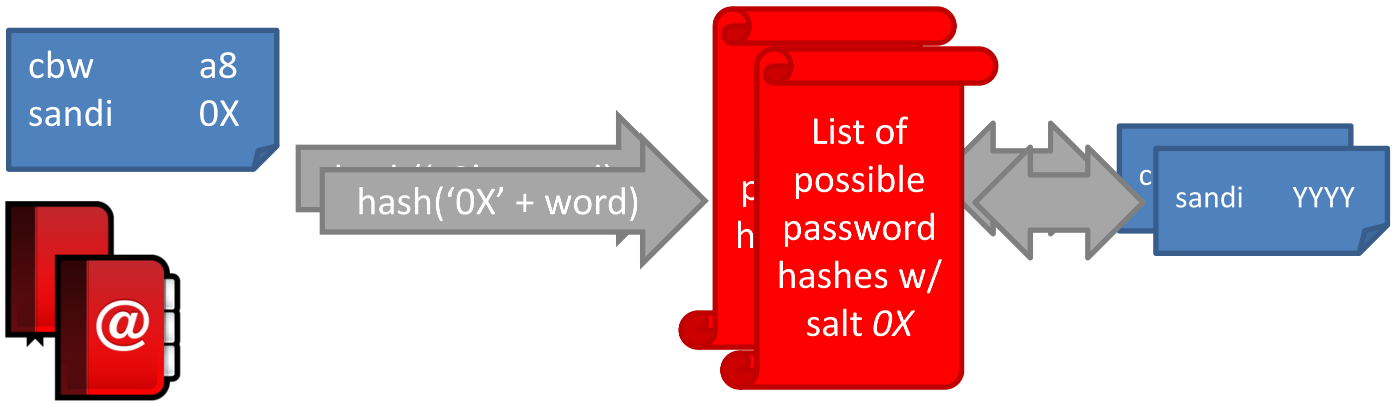
Example Salted Hashes

hashed_password.txt	
cbw	2a9d119df47ff993b662a8ef36f9ea20
sandi	23eb06699da16a3ee5003e5f4636e79f
amislove	98bd0ebb3c3ec3fbe21269a8d840127c
bob	e91e6348157868de9dd8b25c81aebfb9

hashed_and_salted_password.txt		
cbw	a8	af19c842f0c781ad726de7aba439b033
sandi	0X	67710c2c2797441efb8501f063d42fb6
amislove	hz	9d03e1f28d39ab373c59c7bb338d0095
bob	K@	479a6d9e59707af4bb2c618fed89c245

Attacking Salted Passwords


hash()
List of possible password hashes
No matches
hashed_ and_salted_ password.txt



Breaking Hashed Passwords

- Stored passwords should always be salted
  - Forces the attacker to brute-force each password individually

- Problem: it is now possible to compute hashes very quickly
  - GPU computing: hundreds of small CPU cores
  - nVidia GeForce GTX Titan Z: 5,760 cores
  - GPUs can be rented from the cloud very cheaply
    - 2x GPUs for $0.65 per hour (2014 prices)

Examples of Hashing Speed

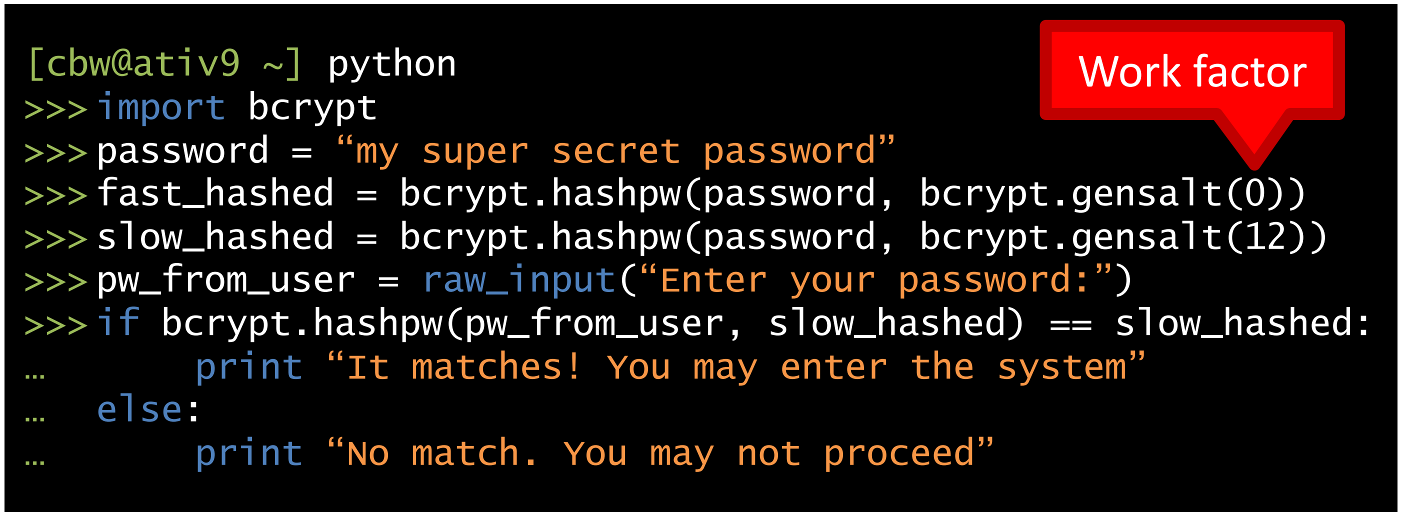
- A modern x86 server can hash all possible 6 character long passwords in 3.5 hours
  - Upper and lowercase letters, numbers, symbols
  - (26+26+10+32)6 = 690 billion combinations
- A modern GPU can do the same thing in 16 minutes
- Most users use (slightly permuted) dictionary words, no symbols
  - Predictability makes cracking much faster
  - Lowercase + numbers → (26+10)6 = 2B combinations

Hardening Salted Passwords

- Problem: typical hashing algorithms are too fast
  - Enables GPUs to brute-force passwords
- Old solution: hash the password multiple times
  - Known as **key stretching**
  - Example: crypt used 25 rounds of DES
- New solution: use hash functions that are designed to be slow
  - Examples: bcrypt, PBKDF2, scrypt
  - These algorithms include a work factor that increases the time complexity of the calculation
  - scrypt also requires a large amount of memory to compute, further complicating brute-force attacks

bcrypt Example

- Python example; install the **`bcrypt`** package



```
[cbw@ativ9 ~] python
>>> import bcrypt
>>> password = “my super secret password”
>>> fast_hashed = bcrypt.hashpw(password, bcrypt.gensalt(0))
>>> slow_hashed = bcrypt.hashpw(password, bcrypt.gensalt(12))
>>> pw_from_user = raw_input(“Enter your password:”)
>>> if bcrypt.hashpw(pw_from_user, slow_hashed) == slow_hashed:
…       print “It matches! You may enter the system”
…   else:
…       print “No match. You may not proceed”
```

Dealing With Breaches

- Suppose you build an extremely secure password storage system
  - All passwords are salted and hashed by a high-work factor function
- It is still possible for a dedicated attacker to steal and crack passwords
  - Given enough time and money, anything is possible
  - E.g. The NSA
- Question: is there a principled way to detect password breaches?

Honeywords

- Key idea: store multiple salted/hashed passwords for each user
  - As usual, users create a single password and use it to login
  - User is unaware that additional honeywords are stored with their account
- Implement a honeyserver that stores the index of the correct password for each user
  - Honeyserver is logically and physically separate from the password database
  - Silently checks that users are logging in with true passwords, not honeywords
- What happens after a data breach?
  - Attacker dumps the user/password database…
  - But the attacker doesn't know which passwords are honeywords
  - Attacker cracks all passwords and uses them to login to accounts
  - If the attacker logs-in with a honeyword, the honeyserver raises an alert!

Honeywords Example

cbw

SHA512("fl" | "p4ssW0rd") → bHDJ8l

Cracked Passwords

User	PW 1	PW 2	PW 3
cbw	123456	p4ssW0rd	Turtles!
sandi	puppies	iloveyou	blizzard
amislove	coff33	3spr3ss0	qwerty

Database

User	Salt 1	H(PW 1)	Salt 2	H(PW 2)	Salt 3	H(PW 3)
cbw	aB	y4DvF7	fl	bHDJ8l	52	Puu2s7
sandi	0x	plDS4F	K2	R/p3Y8	8W	S8x4Gk
amislove	9j	0F3g5H	/s	03d5jW	cV	1sRbJ5

Honeyserver

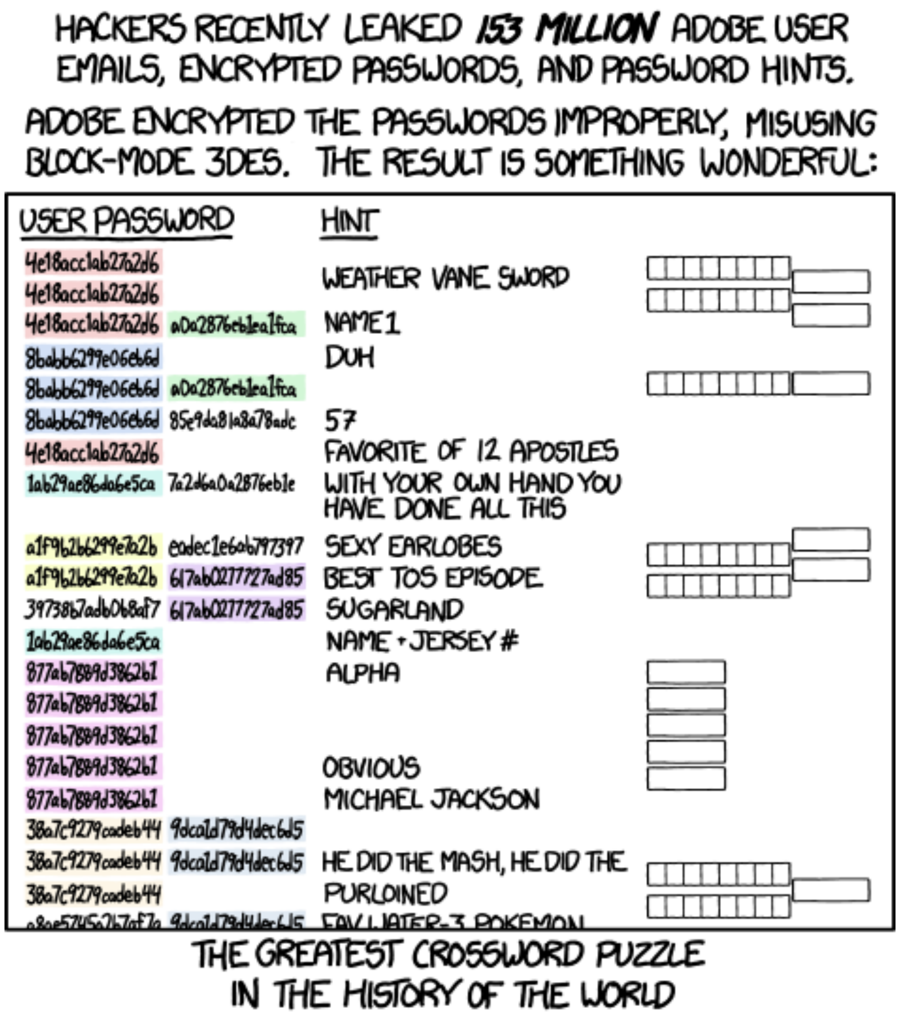
User	Index
cbw	2
sandi	3
amislove	1

Password Storage Summary

- Never store passwords in plain text
  - Always salt and hash passwords before storing them
- Use modern hash functions with a high work factor (e.g. avoid md5)
- Implement honeywords to detect breaches

- These rules apply to any system that needs to authenticate users
  - Operating systems, websites, etc.

XCKD: Encryptic



HACKERS RECENTLY LEAKED *153 MILLION* ADOBE USER EMAILS, ENCRYPTED PASSWORDS, AND PASSWORD HINTS.

ADOBE ENCRYPTED THE PASSWORDS IMPROPERLY, MISUSING BLOCK-MODE 3DES. THE RESULT IS SOMETHING WONDERFUL:

USER PASSWORD		HINT
4e18acc1ab27a2d6		WEATHER VANE SWORD
4e18acc1ab27a2d6		
4e18acc1ab27a2d6	a0a2876eb1ea1fca	NAME1
8babb6299e06eb6d		DUH
8babb6299e06eb6d	a0a2876eb1ea1fca	
8babb6299e06eb6d	85e9da81a8a78adc	57
4e18acc1ab27a2d6		FAVORITE OF 12 APOSTLES
1ab29ae86da6e5ca	7a2d6a0a2876eb1e	WITH YOUR OWN HAND YOU HAVE DONE ALL THIS
a1f96266299e7a2b	eadec1e6ab797397	SEXY EARLOBES
a1f96266299e7a2b	617ab0277727ad85	BEST TOS EPISODE
3973867adb068af7	617ab0277727ad85	SUGARLAND
1ab29ae86da6e5ca		NAME + JERSEY #
877ab7889d3862b1		ALPHA
877ab7889d3862b1		
877ab7889d3862b1		
877ab7889d3862b1		OBVIOUS
877ab7889d3862b1		MICHAEL JACKSON
38a7c9279cadeb44	9dca1d79d4dec6d5	
38a7c9279cadeb44	9dca1d79d4dec6d5	HE DID THE MASH, HE DID THE
38a7c9279cadeb44		PURLOINED
a8ae5745a2b7af7a	9dca1d79d4dec6d5	FAV WATER-3 POKEMON

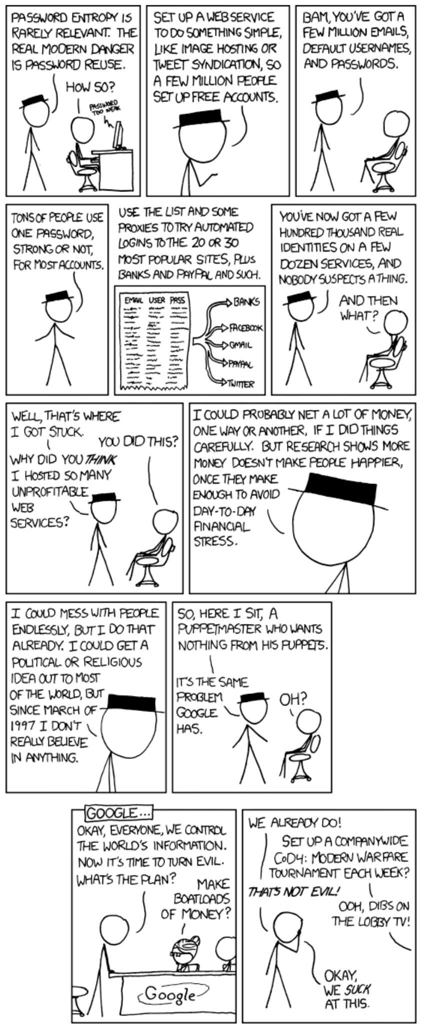
THE GREATEST CROSSWORD PUZZLE IN THE HISTORY OF THE WORLD

Summary

- When dealing with database applications, security needs to be a first-class citizen, considered at all levels, preparing for failure (the weakest link!)
  - Obscurity ≠ Security

- We covered issues/best practices related to authentication/authorization, common attacks, inference control, and encryption

XKCD: Password Reuse


PASSWORD ENTROPY IS RARELY RELEVANT. THE REAL MODERN DANGER IS PASSWORD REUSE.
HOW SO?
PASSWORD TOO WEAK
SET UP A WEB SERVICE TO DO SOMETHING SIMPLE, LIKE IMAGE HOSTING OR TWEET SYNDICATION, SO A FEW MILLION PEOPLE SET UP FREE ACCOUNTS.
BAM, YOU'VE GOT A FEW MILLION EMAILS, DEFAULT USERNAMES, AND PASSWORDS.
TONS OF PEOPLE USE ONE PASSWORD, STRONG OR NOT, FOR MOST ACCOUNTS.
USE THE LIST AND SOME PROXIES TO TRY AUTOMATED LOGINS TO THE 20 OR 30 MOST POPULAR SITES, PLUS BANKS AND PAYPAL AND SUCH.
EMAIL USER PASS
BANKS
FACEBOOK
GMAIL
PAYPAL
TWITTER
YOU'VE NOW GOT A FEW HUNDRED THOUSAND REAL IDENTITIES ON A FEW DOZEN SERVICES, AND NOBODY SUSPECTS A THING.
AND THEN WHAT?
WELL, THAT'S WHERE I GOT STUCK.
YOU DID THIS?
WHY DID YOU THINK I HOSTED SO MANY UNPROFITABLE WEB SERVICES?
I COULD PROBABLY NET A LOT OF MONEY, ONE WAY OR ANOTHER, IF I DID THINGS CAREFULLY. BUT RESEARCH SHOWS MORE MONEY DOESN'T MAKE PEOPLE HAPPIER, ONCE THEY MAKE ENOUGH TO AVOID DAY-TO-DAY FINANCIAL STRESS.
I COULD MESS WITH PEOPLE ENDLESSLY, BUT I DO THAT ALREADY. I COULD GET A POLITICAL OR RELIGIOUS IDEA OUT TO MOST OF THE WORLD, BUT SINCE MARCH OF 1997 I DON'T REALLY BELIEVE IN ANYTHING.
SO, HERE I SIT, A PUPPETMASTER WHO WANTS NOTHING FROM HIS PUPPETS.
IT'S THE SAME PROBLEM GOOGLE HAS.
OH?
GOOGLE...
OKAY, EVERYONE, WE CONTROL THE WORLD'S INFORMATION. NOW IT'S TIME TO TURN EVIL. WHAT'S THE PLAN?
MAKE BOATLOADS OF MONEY?
Google
WE ALREADY DO!
SET UP A COMPANYWIDE CoD4: MODERN WARFARE TOURNAMENT EACH WEEK?
THAT'S NOT EVIL!
OOH, DIBS ON THE LOBBY TV!
OKAY, WE SUCK AT THIS.