Database Security

The Need for Database Security

- Because databases play such an important role in storing large amounts of potentially valuable information, they are often the target of attacks by malicious parties seeking to gain access to this data; hence, we need good ways to secure them.
Tables and Queries

• A very common way to store information is to use a relational database.
• In this approach, information is organized into a collection of tables.
• Each row of a table is a record that stores related information about some entity.
• Each column is associated with an attribute that the entity can possess.

<table>
<thead>
<tr>
<th>No.</th>
<th>Name</th>
<th>Inaugural Age</th>
<th>Age at Death</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>George Washington</td>
<td>57.2</td>
<td>67.8</td>
</tr>
<tr>
<td>2</td>
<td>John Adams</td>
<td>61.3</td>
<td>90.7</td>
</tr>
<tr>
<td>3</td>
<td>Thomas Jefferson</td>
<td>57.9</td>
<td>83.2</td>
</tr>
<tr>
<td>4</td>
<td>James Madison</td>
<td>58.0</td>
<td>88.3</td>
</tr>
<tr>
<td>5</td>
<td>James Monroe</td>
<td>58.8</td>
<td>73.2</td>
</tr>
<tr>
<td>6</td>
<td>John Quincy Adams</td>
<td>57.6</td>
<td>80.6</td>
</tr>
<tr>
<td>7</td>
<td>Andrew Jackson</td>
<td>62.0</td>
<td>78.2</td>
</tr>
<tr>
<td>26</td>
<td>Theodore Roosevelt</td>
<td>42.9</td>
<td>60.2</td>
</tr>
<tr>
<td>27</td>
<td>William Howard Taft</td>
<td>51.5</td>
<td>72.5</td>
</tr>
<tr>
<td>28</td>
<td>Woodrow Wilson</td>
<td>56.2</td>
<td>67.1</td>
</tr>
<tr>
<td>29</td>
<td>Warren G. Harding</td>
<td>55.3</td>
<td>87.7</td>
</tr>
<tr>
<td>30</td>
<td>Calvin Coolidge</td>
<td>51.1</td>
<td>60.5</td>
</tr>
<tr>
<td>31</td>
<td>Herbert Hoover</td>
<td>54.6</td>
<td>90.2</td>
</tr>
<tr>
<td>32</td>
<td>Franklin D. Roosevelt</td>
<td>51.1</td>
<td>63.2</td>
</tr>
<tr>
<td>33</td>
<td>Harry S. Truman</td>
<td>60.9</td>
<td>88.6</td>
</tr>
<tr>
<td>34</td>
<td>Dwight D. Eisenhower</td>
<td>62.3</td>
<td>78.5</td>
</tr>
<tr>
<td>35</td>
<td>John F. Kennedy</td>
<td>43.6</td>
<td>46.5</td>
</tr>
<tr>
<td>36</td>
<td>Lyndon B. Johnson</td>
<td>55.2</td>
<td>64.4</td>
</tr>
<tr>
<td>37</td>
<td>Richard Nixon</td>
<td>56.0</td>
<td>81.3</td>
</tr>
<tr>
<td>38</td>
<td>Gerald Ford</td>
<td>61.0</td>
<td>93.5</td>
</tr>
<tr>
<td>39</td>
<td>Jimmy Carter</td>
<td>52.3</td>
<td></td>
</tr>
<tr>
<td>40</td>
<td>Ronald Reagan</td>
<td>70.0</td>
<td>93.3</td>
</tr>
<tr>
<td>41</td>
<td>George H.W. Bush</td>
<td>64.6</td>
<td></td>
</tr>
<tr>
<td>42</td>
<td>Bill Clinton</td>
<td>46.4</td>
<td></td>
</tr>
<tr>
<td>43</td>
<td>George W. Bush</td>
<td>51.5</td>
<td></td>
</tr>
<tr>
<td>44</td>
<td>Barack Obama</td>
<td>47.5</td>
<td></td>
</tr>
</tbody>
</table>

SQL Queries

• Most databases use a language known as SQL (Structured Query Language) to support queries and updates, using commands that include the following:
  – SELECT: to express queries
  – INSERT: to create new records
  – UPDATE: to alter existing data
  – DELETE: to delete existing records
  – Conditional statements using WHERE, and basic Boolean operations such as AND and OR: to identify records based on certain conditions
  – UNION: to combine the results of multiple queries into a single result
• These commands can be combined to produce queries that extract data, or updates that make changes to the database.
SQL Example

• Suppose, for example, we were to issue the following query on the Presidents table:
  ```sql
  SELECT * FROM Presidents WHERE Inaugural_Age < 50
  ```
• This query is designed to find and return all the U.S. presidents who were younger than 50 when they were inaugurated. The star symbol (*) specifies to return all the attributes of the resulting records.
• This query would return the following table:

<table>
<thead>
<tr>
<th>Num</th>
<th>Name</th>
<th>Inaugural_Age</th>
<th>Age_at_Death</th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
<td>James K. Polk</td>
<td>49.3</td>
<td>53.6</td>
</tr>
<tr>
<td>14</td>
<td>Franklin Pierce</td>
<td>48.3</td>
<td>64.9</td>
</tr>
<tr>
<td>18</td>
<td>Ulysses S. Grant</td>
<td>46.9</td>
<td>63.2</td>
</tr>
<tr>
<td>20</td>
<td>James A. Garfield</td>
<td>49.3</td>
<td>49.8</td>
</tr>
<tr>
<td>22</td>
<td>Grover Cleveland</td>
<td>48.0</td>
<td>71.3</td>
</tr>
<tr>
<td>26</td>
<td>Theodore Roosevelt</td>
<td>42.9</td>
<td>60.2</td>
</tr>
<tr>
<td>35</td>
<td>John F. Kennedy</td>
<td>43.6</td>
<td>46.5</td>
</tr>
<tr>
<td>42</td>
<td>Bill Clinton</td>
<td>46.4</td>
<td></td>
</tr>
<tr>
<td>44</td>
<td>Barack Obama</td>
<td>47.5</td>
<td></td>
</tr>
</tbody>
</table>

Another SQL Example

• More complex queries are also possible, such as one to find all U.S. presidents who were less than 50 when they took office and died during their first term:
  ```sql
  SELECT * FROM Presidents WHERE (Inaugural_Age < 50) AND (Age_at_Death - Inaugural_Age < 4.0)
  ```
• This query would return the following set of records:

<table>
<thead>
<tr>
<th>Num</th>
<th>Name</th>
<th>Inaugural_Age</th>
<th>Age_at_Death</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>James A. Garfield</td>
<td>49.3</td>
<td>49.8</td>
</tr>
<tr>
<td>35</td>
<td>John F. Kennedy</td>
<td>43.6</td>
<td>46.5</td>
</tr>
</tbody>
</table>
Database Deletions

- In addition to queries that extract information from a database, authorized users can also update the contents of a database using SQL commands.
- For example, the following update operation would delete all of those records from the Presidents table that correspond to U.S. presidents who were less than 50 years old when they were inaugurated:
  
  ```sql
  DELETE FROM Presidents WHERE Inaugural_Age < 50
  ```

Database Insertions

- In addition, the following update operation would add a new record to the Presidents table:
  
  ```sql
  INSERT INTO Presidents
  VALUES (45, 'Arnold Schwarzenegger', 65.5, NULL)
  ```
- Database updates can be more fine-grained than just inserting and deleting entire records, however.
- We can also alter the contents of individual attribute values in specific records.
Two-Phase Commit

• To cope with consistency and reliability issues, most databases employ a protocol called **two-phase commit** for performing updates.
  1. The first phase is a request phase, in which all the parts of the database that need to change as a result of this update are identified and flagged as being intended for this change. The result of this phase is either that it completes successfully, and every change requested is available and now flagged to be changed, or it aborts, because it couldn’t flag all the parts it wanted (say, because someone else already flagged it) or because of a network or system failure.
    – If the first phase aborts, then all its requested changes are reset, which is always possible, because no permanent changes have been made yet.
    – If the first phase completes successfully, then the protocol continues to the second phase.
  2. The second phase is the commit phase, in which the database locks itself against other changes and performs the sequence of changes that were identified in the request phase.
    – If it completes successfully, then it clears all the flags identifying requested changes and it releases the lock on the database.
    – If, on the other hand, this operation fails, then it rolls back, that is, reverses, all the changes made back to the state the database was in just after completing the first phase.
• This two-phase commit protocol is a feature that a database can use to help achieve both integrity and availability.

Database Access Control

• A proper set of database access controls should implement a **least-privilege principle**, so that each user has the necessary rights to perform their required tasks, but no rights beyond that.
• A proper set of database access controls should also implement a **separation of privilege principle**, so that different users have different privileges, depending on the different tasks that they need to perform.
Access Control Using SQL

• SQL defines an access control framework that is commonly used for defining database privileges.
• When a table is created, the owner of the table has the sole rights to perform operations on that table.
• The owner can then grant privileges to other users, which is known as privilege delegation.
• These privileges may be broad, such as the ability to do anything to a particular table, or fine-grained, such as the ability to perform only SELECT queries on certain columns.
• For example, the owner of a table may issue the following SQL command to give Alice the ability to search through table employees:
  
  \[
  \text{GRANT SELECT ON employees TO Alice;}
  \]

• Other permissions that can be provided using the GRANT keyword include DELETE, INSERT, and UPDATE. In addition, to grant all available rights one can use the ALL keyword.

Privilege Delegation

• In addition to being able to grant certain privileges to other users, table owners can also allow other users to grant privileges for those tables, which is known as policy authority delegation.
• Specifically, when granting a privilege to a user as in the above examples, the grantor can include the clause WITH GRANT OPTION to give the recipient the ability to further delegate that privilege.
• For example, an administrator might create a view for Alice and give her permission to delegate SELECT permissions on that view to other users as follows:
  
  \[
  \text{CREATE VIEW employees_alice AS}
  \text{SELECT * FROM employees}
  \text{WHERE name = 'Alice';}
  \text{GRANT SELECT ON employees_alice TO Alice WITH GRANT OPTION;}
  \]
Privilege Revocation

• The propagation of privileges in a database can be visualized using a diagram, where nodes represent users and directed edges represent granted privileges.

• If Alice grants a set of rights, A, to Bob, then we draw a directed edge labeled with A from Alice to Bob.

• A user, Alice, who has granted privileges to another, Bob, can opt to revoke those privileges at a later time, which would be visualized by deleting or relabeling the edge from Alice to Bob.

• A command that could perform such a revocation is as follows:
  
  `REVOKE SELECT ON employees FROM Bob;`

• This command should result in the revocation of all SELECT privileges for Alice as well as all the people to which she had delegated this privilege.

---

Privilege Revocation Example

(a) First, two administrators, Charles and Diane, each grant Alice two sets of privileges, C and D, after which Alice grants those privileges to Bob, giving him the set of rights in the union, C U D.

(b) If Charles subsequently revokes the set of privileges, C, he granted to Alice, then the privileges Bob inherited indirectly from Charles, through Alice, should also be revoked, leaving Bob with just the privileges in D.
Sensitive Data

- In addition to ensuring that databases have appropriate access-control measures in place, care must be taken to guarantee that sensitive data is stored in a way that protects the privacy of users and any confidentiality requirements for sensitive data.

Using Cryptography

- If information being stored in a database has confidentiality requirements, then it should not be stored in plaintext, but should instead be stored as the output of a cryptographic function.
- Confidential information kept in a database should be stored in encrypted form, where the decryption key should be known by authorized users but not stored in the database itself.
Privacy Protection

• Besides measures designed to protect the confidentiality of sensitive user information, database owners should be careful to consider the privacy impacts of publishing or granting access to sensitive information.

• If a database is to be released to the public, say, to be used for research purposes, then all identifying information, such as names, addresses, Social Security numbers, employee numbers, and student numbers, should be removed or changed to masking values, which are nondescript values that lack all identifying information.

Inference Attacks

• Even if identifying information is removed or masked out, it may still be possible to use the database in conjunction with additional information available to the attacker to learn more about the underlying data.

• This is referred to as an inference attack.

• As an example, consider a database of employee records, whose attributes are name, gender, ID number, and salary.
  – Suppose a party is granted access to a sanitized version of the table, where the name attribute is removed, for the purpose of creating statistics on salary by gender. Another party may have a list of pairings associating ID numbers to names for a reporting task.
  – If these two parties were to communicate, they could easily infer the salary of each employee, despite the intent of the database owner. In general, when granting access to modified versions of a database, administrators should consider whether collusion among grantees can allow them to gain unauthorized information.
Protecting Against Inference Attacks

• To protect a database from inference attacks, the following techniques can be used prior to making the database public.
  
  • **Cell suppression.** In using this technique, some of the cells in a database are removed and left blank in the published version.
  
  • **Generalization.** In using this technique, some values in a published database are replaced with more general values.
  
  • **Noise addition.** In using this technique, values in a published database have random values added to them, so that the noise across all records for the same attribute averages out to zero.

---

Example

![Table Example](image)

Obfuscation techniques for protecting the privacy of individuals included in a public database: (a) A table with individual names removed. (b) A table anonymized using cell suppression. (c) A table anonymized using generalization. (d) A table anonymized using noise.