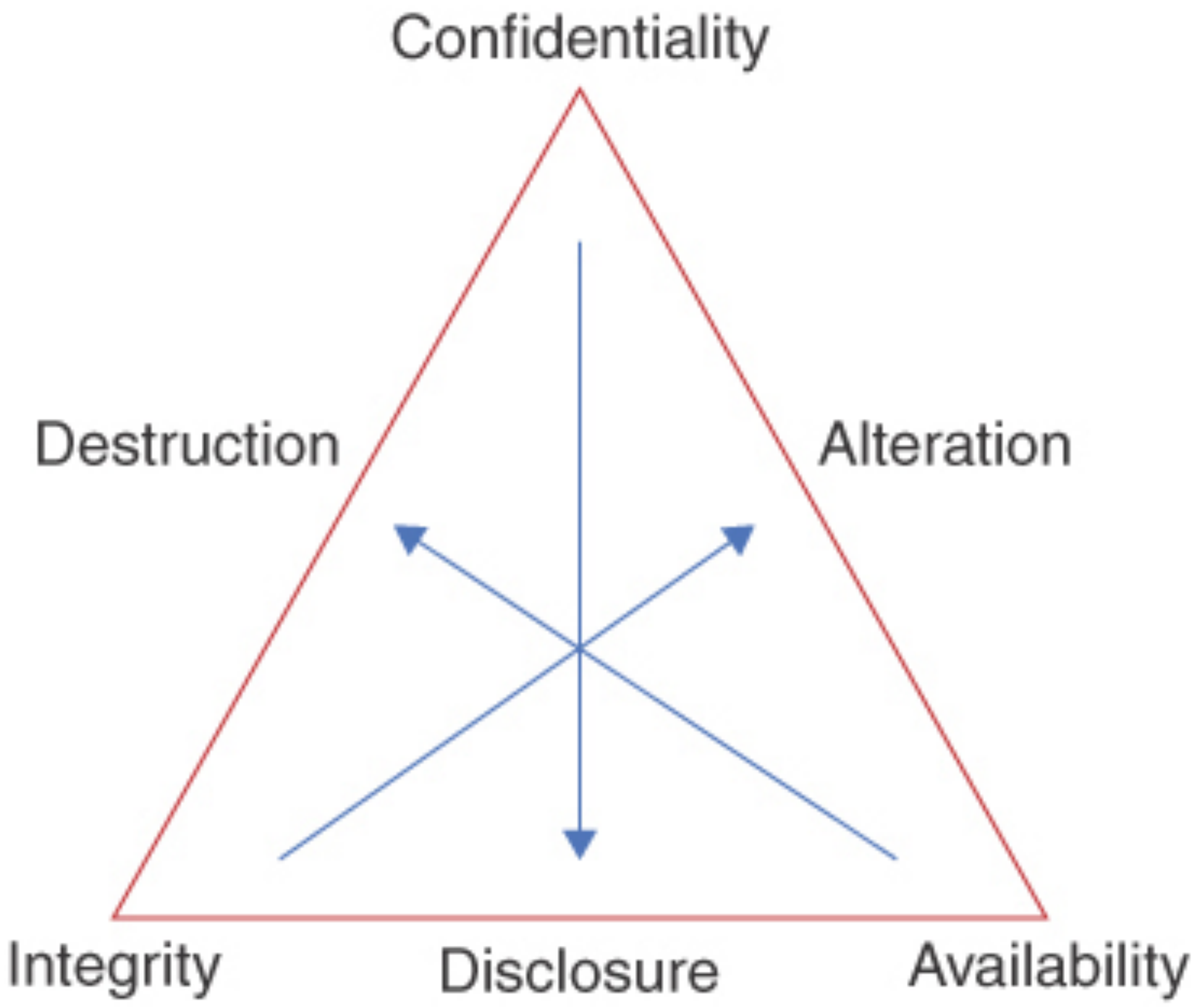
Database Security Principles

Thomas Schwarz, SJ

Security Principles

- CIA-DAD Triad



Security Principles

- I-A-A-A
  - Identification
  - Authentication
  - Authorization
  - Auditing / Accounting
- SMART
  - Specific, Measurable, Attainable, Realistic, and Time bound

Security Principles

- Firewalls, Access Controls, Access Control Lists

Security Principles

- Protecting Data:
  - Encrypt: Symmetric - Asymmetric, PKI
  - Compress
  - Index
  - Archive
- Follow NIST guidelines:
  - AES 128 for Secret Encryption
  - AES 256 for Top Secret
  - SHA 256 for Secret Hashing
  - SHA 384 for Top Secret

Authorization

Authorization

- Basic Structure
  - Subjects have rights over objects
    - Subjects can also be objects (e.g. processes can generate other processes and retain rights over them)
  - Basic Implementation
    - Rights matrix
      - Subjects - rows
      - Objects - columns
      - Entries - rights

Authorization

- Rights matrix:
  - Implemented as a sparse matrix
  - Implemented as a relational database table
  - Implemented as access control lists:
    - Each object has a list of users with rights over it
  - Implemented as Capabilities
    - Each subject has a list of objects with rights over them

Authorization

- Static Authorization
  - No generation of objects, subjects, rights
  - Theoretically and practically treatable
  - Can prove that certain actions remain prohibited
- Dynamic Authorization
  - Generation of new subjects, objects, and rights
  - Inheritance of rights
  - Determining whether certain actions remain prohibited is NP-complete

Access Control

- Intermediate Access Control Mechanisms
    - Groups
        - Permissions through belonging to a group
        - Denials implemented by exceptions
    - Protection Rings:
        - Subjects and objects are ordered in a linear hierarchy
            - E.g. Ultra — Top Secret — Secret — Confidential — Open

Access Control

- Protection Ring Example:
  - CPU Hardware allows for four levels of protection
    - OS - Kernel
    - OS
    - Utilities
    - User Processes
  - A process can only access an object if it belongs to the same or a lower level of control
  - Processes can create objects only at their own level

Access Control

- Intermediate Access Control Mechanisms
  - Security Classes (a.k.a.) Security Lables
    - Information control policies consists of
      - Security class definitions
      - Definition of a “can flow” relationship
      - A join operation A # B that combines rights and restrictions of two classes

Access Control

- Styles of control:
  - DAC — Discretionary Access Control
    - Access is granted based on identity of objects and subjects
  - MAC — Mandatory Access Control
    - Access mediated by security levels
    - Subject cannot pass information to subjects with lower classification
    - No read up: Subject can only read objects at the same or lower security level
    - No write down: Subject can only write to objects of the same or higher security level

Access Control

- Refined MAC
    - Instead of heaving a linear hierarchy, have a grid hierarchy
        - Example:
            - CRYPTO for cryptographic algorithms
            - COMSEC for communications security
            - OPSEC for operational security
        - Each object, subject has now three classifications
        - All the rules still apply

Access Control

- Role-Based Access Control
- A role describes an aspect of a subject
- A subject can change role (but not group)
- Rights depend only on the role

Access Control

- RBAC Example:
  - Hospital:
    - Roles:
      - Attending physician
      - Dietitian
      - Nurse
      - Pharmacist
      - Accountant
      - Chaplain
      - Social worker
      - …

Access Control

- RBAC Hospital example:
  - Chaplain can look up religious affiliation of a patient
  - Accountant can find home address and insurance information
  - Dietician cannot find home phone number
  - Doctor cannot find insurance carrier of a patient
  - Doctor cannot find health history of another doctor’s patient

Access Control

- EXERCISE
  - Read one of
    - http://crpit.com/confpapers/CRPITV32Evered.pdf
    - http://mjcs.fsktm.um.edu.my/document.aspx?FileName=99.pdf
  - Create 1-2 pages précis

Authorization

- RBAC:
  - Access control is designed:
    - Identifying roles
    - Identifying object classes
    - Establishing rules based on roles and object classes

Applications to Databases

- Use stored procedures to update and read tuples
    - Weak point: stored procedures might be accessed by the adversary

Applications to Databases

- Grant rights to users
  - Create Roles

```
CREATE ROLE Human_Resc_Read;

GRANT SELECT on Finance.EmpView, Finance.Employees to
Human_Resc_read;

CREATE ROLE Human_Resc_Write;

GRANT DELETE, INSERT, UPDATE on Finance.EmpView,
Finance.Employees to Human_Resc_read;
```

Applications to MySQL

- Create users

```
CREATE USER [IF NOT EXISTS] account_name
IDENTIFIED BY 'password';
```

- Check users:

```
SELECT
  user
FROM
  mysql.user;
```

```
create user bob@localhost
identified by 'Marquette';
```

```
select user from mysql.user;
```

Applications to MySQL

- Granting privileges:
  - Privilege levels:
    - Global
    - Database
    - Table
    - Column
    - Stored Routine

Applications to MySQL

- Global:

```
GRANT SELECT
ON *.*
TO bob@localhost;
```

Applications to MySQL

- Databases:

```
GRANT INSERT
ON classicmodels.*
TO bob@localhost;
```

```
GRANT DELETE
ON classicmodels.employees
TO bob@localhsot;
```

Applications to MySQL

- Attributes:

```
GRANT
   SELECT (employeeNumner,lastName,firstName,email),
   UPDATE(lastName)
ON employees
TO bob@localhost;
```

Applications to MySQL

- Stored Procedures

```
GRANT EXECUTE
ON PROCEDURE CheckCredit
TO bob@localhost;
```

Applications to MySQL

- Proxy:
  - Allows one user to act for another

```
GRANT PROXY
ON bob@localhost
TO alice@localhost;


SHOW GRANTS FOR super@localhost;
```

Applications to MySQL

- Revoking:

```
REVOKE privilegee [,privilege]..
ON [object_type] privilege_level
FROM user1 [, user2] ..;
```

Applications to MySQL

```
REVOKE
    ALL [PRIVILEGES],
    GRANT OPTION
FROM user1 [, user2];
```

Applications to MySQL

- Creating roles
  - Example:
    - create a database crm

      ```
      CREATE DATABASE crm;
      ```

    - switch to the database

      ```
      USE crm;
      ```

Example

- Create a table

```
CREATE TABLE customers(
    id INT PRIMARY KEY AUTO_INCREMENT,
    first_name VARCHAR(255) NOT NULL,
    last_name VARCHAR(255) NOT NULL,
    phone VARCHAR(15) NOT NULL,
    email VARCHAR(255)
);
```

Example

- Populate table

```
INSERT INTO
customers(first_name,last_name,phone,email)
VALUES
('John','Doe','4081234567','j.doe@marquette.edu'),
('Bambi','Roe','4087654321','b.roe@marquette.edu');
```

Example

- Create three roles:

```
CREATE ROLE
    crm_dev,
    crm_read,
    crm_write;
```

Example

- Grant rights:

```
GRANT ALL
ON crm.*
TO crm_dev;
```

```
GRANT SELECT
ON crm.*
TO crm_read;
```

Example

- Granting rights

```
GRANT INSERT, UPDATE, DELETE
ON crm.*
TO crm_write;
```

Example

- Create users

```
CREATE USER crm_dev1@localhost IDENTIFIED BY
'Secure$1782';

CREATE USER crm_read1@localhost IDENTIFIED BY
'Secure$5432';

CREATE USER crm_write1@localhost IDENTIFIED BY
'Secure$9075';
CREATE USER crm_write2@localhost IDENTIFIED BY
'Secure$3452';
```

Example

- Assign roles to users:

```
GRANT crm_dev
TO crm_dev1@localhost;


GRANT crm_read
TO crm_read1@localhost;


GRANT crm_read,
    crm_write
TO crm_write1@localhost,
    crm_write2@localhost;
```

Example

- Display grants

```
SHOW GRANTS FOR crm_dev1@localhost;
```

- Display privileges

```
SHOW GRANTS
FOR crm_write1@localhost
USING crm_write;
```

Example

- Users still need to **activate** roles

```
SET ROLE NONE;

SET ROLE DEFAULT;

SET ROLE
    granted_role_1
```

Example

```
REVOKE INSERT, UPDATE, DELETE
ON crm.*
FROM crm_write;
```

Injection Attacks

SQL Injection

- Scenario: Website input is made into an sql query to a database

```
string sql = “select * from client where name = ‘ “
+ uname + “ ’ ”;
```

- User enters “Schwarz"

```
string sql = “select * from client where name
= ‘ Schwarz’ ”;
```

- User enters “'Schwarz' or 1=1 ”

```
string sql = “select * from client where name =
‘Schwarz’  or 1=1”;
```

SQL Injections

- Some database servers allow more than one SQL statement
  - Use: “Schwarz' drop table client”
  - Result makes a lookup and then destroys the table
- Results are magnified when the database runs with administrator privileges

SQL Injection

- URL query string for an article

  http://somesite.com/store/itemdetail.asp?id=666

- Without filtering passed to SQL gives:

```
SELECT name, picture, description price FROM
products WHERE id=666
```

```
$SQLquery = “SELECT * FROM users WHERE  username=`”.
$_POST[“username”].”’ AND password=‘”.$_POST[“password”].”’”;

$DBresult=db_query($SQLQuery);
if($DBresult)      {
        // username-password is correct, log the user on
}
else  {
        //username-password is incorrect
}
```

```
SELECT accountdata FROM acountinfo
WHERE accountid = ` ‘;
INSERT INTO accountdata (accountid,password)
VALUES (`thomas`,’12345’) – ‘ AND password = ‘ ‘
```

Examples

- 2008 Heartland Payment System
  - Approximately 130 million credit and debit card numbers were exposed.
- 2011 Sony Pictures
  - 77 million PlayStation Network accounts
  - estimated $170 million damage

Examples

- TalkTalk (2015)
    - 157,000 customers

Try It Out

- https://portswigger.net/web-security/sql-injection