SEZG566/SSZG566

Secure Software Engineering

Code and Database Security

BITS Pilani

Pilani | Dubai | Goa | Hyderabad

T V Rao

DISCLAIMERS

- *The slides presented here are obtained from the authors of the books, product documentations, and from various other contributors. I hereby acknowledge all the contributors for their material and inputs.*
- *I have added and modified slides to suit the requirements of the course.*

BITS Pilani

Pilani | Dubai | Goa | Hyderabad

Java Security

Inherent Java Security

The Java language is designed keeping security in mind.

- Every entity has an associated Access Level:
  - Public, Protected, Default/Package, Private
  - Provides encapsulation
- A strongly typed language:
  - Restrictions on how data types can be mixed
- No direct memory access
  - No notion of pointers
  - Entities are accessed via references (by name)
- Variables must be initialized before they are used
- Objects can’t be arbitrarily cast into other objects (ensures a type safe environment):
  - Strict use of extends, implements (inheritance)
  - Compile time type checking of casting
- Provides automatic memory management, garbage collection, and array range-checking

The Java Runtime Environment (JRE)

- Consists of the Java Virtual Machine (JVM) and Class Libraries
- JVM: available for most platforms, provides the environment for java bytecode to execute
    - Offers Platform Independence: “Write once, run anywhere!”
    - Is an abstract virtual machine
        - Diff. implementations: Sun, IBM, Oracle, MS
        - Each thread has its own stack
        - Typical instruction set: Load/Store, Arithmetic, etc.
    - Interprets bytecode generated by Java compilers
- Class Libraries: The Core Java API, contains classes for language support and added functionality

Java Security Evolution:

Figures 1-5, 1-6, 1-7 in pages 1-20, 21, 22 of Security Developer Guide for Java by Oracle.

Bytecode Verifier

When a class loader presents the bytecodes of a newly loaded class to JVM, these bytecodes are first inspected by a Verifier.

- All classes except for system classes are verified; but verification can be deactivated with undocumented –noverify option

Here are some of the checks that the verifier carries out:

- Variables are initialized before they are used.
- Method calls match the types of object references.
- Rules for accessing private data and methods are not violated.
- Local variable accesses fall within the runtime stack.
- The runtime stack does not overflow.
  - If any of these checks fails, then the class is considered corrupted and will not be loaded

A class file generated by a compiler for the Java programming language always passes verification. However, the bytecode can be changed by someone with some experience in assembly programming and a hex editor to manually produce a class file that contains valid but unsafe instructions for the Java virtual machine

Class Loader

A Java compiler converts source instructions into bytecode for the Java virtual machine.

- Each class file contains the definition and implementation code for one class or interface. These class files must be interpreted into the machine language of the target machine.

The virtual machine loads only those class files that are needed for the execution of a program.

The class loading mechanism doesn't just use a single class loader, has at least three class loaders:

- The bootstrap class loader : loads the system classes, typically from rt.jar; integral part of the JVM; usually implemented in C
- The extension class loader : loads "standard extensions" from jre/lib/ext directory; will find the classes in them, even without any class path
- The system class loader (also sometimes called the application class loader) : loads the application classes. It locates classes in the directories and JAR/ZIP files on the class path (CLASSPATH environment variable or –classpath option)
  - In Sun's Java implementation, the extension and system class loaders are implemented in Java

Java Security Sandbox

The base Java Security sandbox is comprised of three major components: the **byte code Verifier**, the **Class Loader**, and the **Security Manager**.

- The Security Manager depends on Class Loaders to correctly label code as trusted or untrusted. Class Loaders also shield the Security Manager from spoofing attacks by protecting local trusted classes making up the Java API.
- On the other hand, the class loader system is protected by the Security Manager, which ensures that an applet cannot create and use its own Class Loader.
- The Verifier protects both the Class Loaders and the Security Manager against language-based attacks meant to break the VM. All in all, the three parts intertwine to create a default sandbox.
- However, the three parts are not created or specified by a standards committee.
  - Java applications, including Java-enabled Web browsers, are allowed to customize two of the fundamental portions of the security model to suit their needs (the Class Loader and the Security Manager).
- A great deal of faith is placed in the ability of VM implementations to ensure that untrusted code remains properly contained. Bugs in the system will compromise the entire security model.

Java Security Weaknesses

- Fine-grained control with a lot of complexity. Learning curve for developers
- Relies on user to secure their own environment via a complex Policy Tool
- Multiple JVM Implementations: each have their own unique vulnerabilities
- Reverse engineering of class files to source code(software watermarking, code obfuscation can not be security)
- Several flaws have been addressed over the evolution of Java and the JVM

(Some) Java Security Guidelines

- Java offers several ways to allocate uninitialized objects. The easy way to protect yourself against this problem is to write your classes so that before any object does anything, it verifies that it has been initialized.
- If a class or method is non-final, an attacker could try to extend it in a dangerous and unforeseen way. Make something non-final only if there is a good reason, and document that reason.
- Do not depend on package scope - Classes, methods, and variables, by default, are accessible within the same package. Sometimes an attacker could introduce a new class inside the package, and use this new class to access the things programmer thought hidden
- Java language allows inner classes (class within class). Java byte code has no concept of inner class, so it becomes ordinary class in the package. Further inner class has privilege to access private members in the containing class.

Securing Java by Gary McGraw, Ed Felten, John Wiley

(Some) Java Security Guidelines

- Make Your Classes Uncloneable – Cloning creates new instances without executing constructor. If you must permit cloning, make the clone method final.
  - attacker can define a subclass of your class, and make the subclass implement java.lang.Cloneable. You can prevent subclass cloning by defining the following method in each of your classes:

```
public final void clone() throws java.lang.CloneNotSupportedException {
    throw new java.lang.CloneNotSupportedException();
}
```

- Make Your Classes Unserializeable - Serialized classes expose internal dynamic state of objects in byte code format to an attacker.

- Make Your Classes Undeserializeable - Even if a class is not serializeable, it may be deserializeable. An adversary can create a sequence of bytes that happens to deserialize to an instance of your class, and you do not have control over what state the deserialized object is in.

Similar serialization issue in Python: https://davidmatablog.wordpress.com/2020/06/07/owasp-insecure-deserialization-with-python/

Securing Java by Gary McGraw, Ed Felten, John Wiley

BITS Pilani

Pilani | Dubai | Goa | Hyderabad

Open Source Security

Open Source Security

What is open source software

- Open-source software is computer software that is released under a license in which the copyright holder grants users the rights to use, study, change, and distribute the software and its source code to anyone and for any purpose. - Wikipedia

Why Open Source?

- Open collaboration by online participation.
- High Reuse
- Examination of the code facilitates public trust

Linux Foundation/Snyk Survey of Open Source Security 2022: https://snyk.io/reports/open-source-security/

Open Source Security

Casson and Ryan have pointed out several policy-based reasons for adoption of open source –

- Security
- Affordability
- Transparency
- Perpetuity
- Interoperability
- Flexibility
- Localization

It has become double-edge sword

- Both high productivity and security if used proven packages
- Huge risk and challenges if package vulnerabilities are exposed

Python Security – An Empirical Study

- Level of code reusability supported by software ecosystems also makes the discovery of security vulnerabilities much more difficult

An empirical study of 550 vulnerability reports affecting 252 Python packages showed

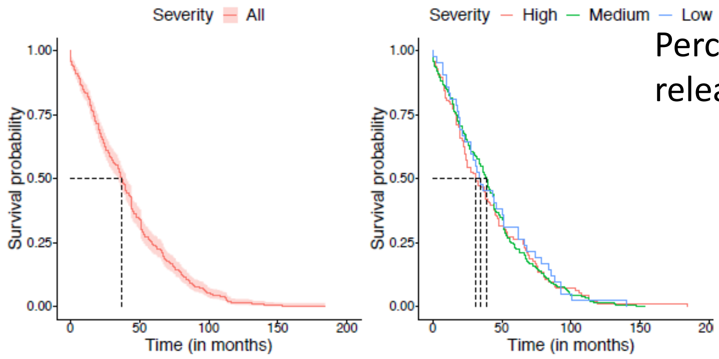
- The discovered vulnerabilities in Python packages are increasing over time
- Some take more than 3 years to be discovered
- The majority of these vulnerabilities (50.55%) are only fixed after being publicly announced, giving ample time for attackers exploitation

Empirical Analysis of Security Vulnerabilities in Python Packages Mahmoud Alfadel, Diego Elias Costa, Emad Shihab

Python Security – An Empirical Study

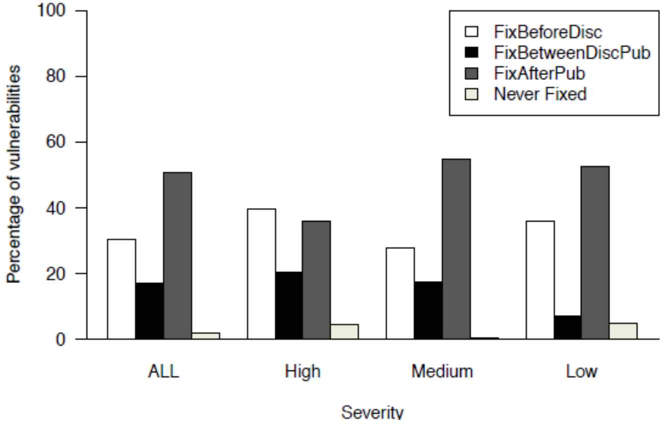
Ranking of the 5 most commonly found vulnerability types (CWE) in PyPi

Rank	Vulnerability type (CWE)	Freq.	Frequency by severity High	Medium	Low
1	Cross-Site-Scripting (XSS)	130	4	118	8
2	Denial of Service (DoS)	72	11	59	2
3	Arbitrary Code Execution	66	39	26	1
4	Information Exposure	60	8	44	8
5	Access Restriction Bypass	34	10	23	1



Kaplan-Meier survival probability for package vulnerabilities to get discovered for all vulnerabilities (left-side plot) and for vulnerabilities broken by severity (right-side plot)

Percentages of vulnerabilities according to the release time of the first fixed version by severity.



Empirical Analysis of Security Vulnerabilities in Python Packages Mahmoud Alfadel, Diego Elias Costa, Emad Shihab

BITS Pilani

Pilani | Dubai | Goa | Hyderabad

Case Study

Sony Hack

- At about 7 a.m. Pacific time on Monday, Nov. 24, 2014—a severe cyberattack was launched on Sony Pictures, US.
  - Employees logging on to its network were met with the sound of gunfire, scrolling threats, and the menacing image of a fiery skeleton looming over the tiny zombified heads of the studio's top two executives.
- Hackers' malware had spread across continents, wiping out half of Sony's global network
  - It erased 3,262 of the company's 6,797 personal computers and 837 of its 1,555 servers
  - Sony Pictures started using fax machines, postal services, and paid its 7,000 employees with paper checks

http://fortune.com/sony-hack-part-1/

Sony Hack

Hackers stole the data before destroying on Sony machines.

- they started dumping unfinished movie scripts and confidential emails to salary lists and more than 47,000 Social Security numbers.
- Five Sony films, four of them unreleased, were leaked to piracy websites for free viewing.

Hackers threatened a 9/11-style attack against theaters, where Sony is to do the movie *The Interview*'s Christmas release

- Sony abandoned the movie release
- Sony Pictures released it on internet through video on demand and in few select theatres

The Interview – A Sony film

The Interview is a 2014 American political satire comedy film - a comedy depicting the killing of North Korea's sovereign leader

- In June 2014, the North Korean government threatened action against the United States if the film is released.
- Producers delayed the release from October to December, and reportedly re-edited the film to make it more acceptable to North Korea.
- In November, the computer systems of Sony Pictures Entertainment were hacked by the "Guardians of Peace", a group the FBI claims has ties to North Korea.
- The hacker group also threatened terrorist attacks against cinemas that showed the film.

President Obama expressed dissatisfaction with the film producers' response

URL - Abstract

https://www.sans.org/reading-room/whitepapers/casestudies/paper/36022

On November 24, 2014, an incident almost pulled right out of a 90’s hacker mov ie transformed into a massive computer hack. A group calling itself The Guardians of Peace (GOP) managed to breach Sony Pictures Entertainment and bring their systems down to a screeching halt. Resulting from this breach the GOP claims to have stolen over 100 terabytes of data containing Social Security numbers, salaries, movies, and other personally identifiable information. Within days, the stolen data was posted on the Internet along with demands from the GOP group that included not releasing *The Interview*.

This paper will point out some of the Critical Controls that could have been utilized to minimize the impact the GOP had on the Sony breach. Utilizing even a few of the Critical Controls such as malware defenses, monitoring, audit logs, encryption, controlled use of administrative credentials, and incident response could have provided the necessary implementations required to prevent a 90’s hacker movie from turning into reality

Top 20 Critical Controls (SANS Institute)

Table 1: Top 20 Critical Controls (SANS Institute, 2015b).

Top 20 Critical Controls

1. Inventory of Authorized and Unauthorized Devices
2. Inventory of Authorized and Unauthorized Software
3. Secure Configurations for Hardware and Software on Mobile Devices, Laptops, Workstations, and Servers
4. Continuous Vulnerability Assessment and Remediation
5. Malware Defenses
6. Application Software Security
7. Wireless Access Control
8. Data Recovery Capability
9. Security Skills Assessment and Appropriate Training to Fill Gaps
10. Secure Configurations for Network Devices such as Firewalls, Routers, and Switches
11. Limitation and Control of Network Ports, Protocols, and Services
12. Controlled Use of Administrative Privileges
13. Boundary Defense
14. Maintenance, Monitoring, and Analysis of Audit Logs
15. Controlled Access Based on the Need to Know
16. Account Monitoring and Control
17. Data Protection
18. Incident Response and Management
19. Secure Network Engineering
20. Penetration Tests and Red Team Exercises

Steps for Reducing Risk with Critical Controls (SANS Institute)

Step 1: Perform Initial Gap Assessment - determining what has been implemented and where gaps remain for each control and sub-control.

Step 2: Develop an Implementation Roadmap - selecting the specific controls (and subcontrols) to be implemented in each phase, and scheduling the phases based on business risk considerations.

Step 3: Implement the First Phase of Controls - identifying existing tools that can be repurposed or more fully utilized, new tools to acquire, processes to be enhanced, and skills to be developed through training.

Step 4: Integrate Controls into Operations - focusing on continuous monitoring and mitigation and weaving new processes into standard acquisition and systems management operations.

Step 5: Report and Manage Progress against the Implementation Roadmap developed in Step 2. Then repeat Steps 3-5 in the next phase of the Roadmap.

BITS Pilani

Pilani | Dubai | Goa | Hyderabad

Assignment

Option 1 (Reactive Assignment)

Step – 1:

Refer to a prominent security breach that is widely published, e.g. one from https://www.csoonline.com/article/2130877/the-biggest-data-breaches-of-the-21st-century.html

Or OWASP top ten

Or Panama Papers Incident

Etc. (Any significant security incident of interest to you)

Step-2:

Identify root cause of the incident. List corresponding CVEs and CWEs.

Step-3:

Provide list of practices/recommendations that would have avoided the incident.

Option 2 (Proactive Assignment)

Step – 1:

Pick an industry (e.g. bank, telecom etc.) of your choice. Choose technology platforms / reference architecture that are commonly used.

Step-2:

Perform Threat Modelling for the organization and software to be developed. Generate necessary artifacts. Document in detail at least one of the artifact. Propose security-specific requirements.

Step-3:

Provide recommendations to the technology group w.r.t. development/operations.

Option 3 (Survey Assignment)

Step – 1:

Pick an area for deep exploration. Some suggested areas:

- Database Hardening
- Operating System Hardening
- Anti-Ransomware Solutions
- Dark Web
- Any other area (with prior approval of the faculty)

Step-2:

Cover Challenges, Best Practices, and Vendor offerings

Expect illustrations in the write-up to connect with listings from CVE/CWE/OWASPtop10 and other known events.

Individual or Group?

If you decide to work as a group, you need to submit a file that describes contributions made by each individual.

Assessment will be done individually based on quality, concepts, structure, and practical applicability of the work.

BITS Pilani

Pilani | Dubai | Goa | Hyderabad

Database Security Overview

Databases

- Structured collection of data stored for use by one or more applications
- Contains the relationships between data items and groups of data items
- Often contains sensitive data that needs to be secured

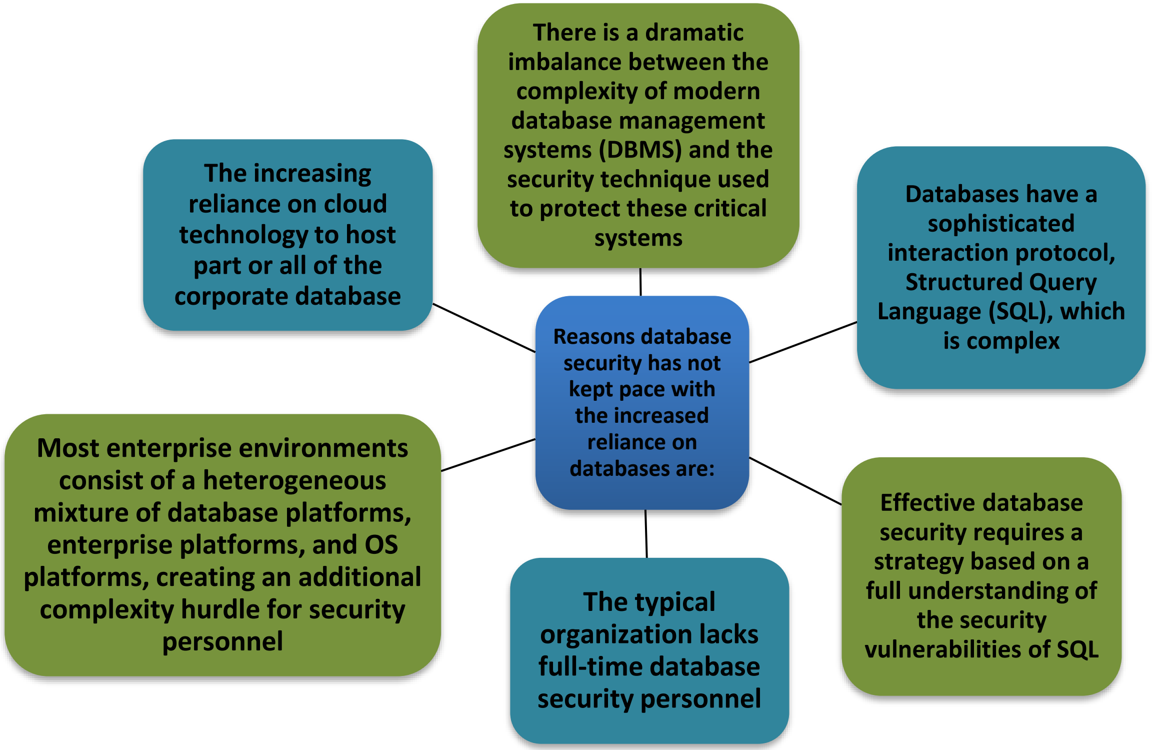
Query language

- Provides a uniform interface to the database for users and applications

Database management system (DBMS)

- Suite of programs for constructing and maintaining the database
- Offers ad hoc query facilities to multiple users and applications

Database Security



Reasons database security has not kept pace with the increased reliance on databases are:

There is a dramatic imbalance between the complexity of modern database management systems (DBMS) and the security technique used to protect these critical systems

Databases have a sophisticated interaction protocol, Structured Query Language (SQL), which is complex

Effective database security requires a strategy based on a full understanding of the security vulnerabilities of SQL

The typical organization lacks full-time database security personnel

Most enterprise environments consist of a heterogeneous mixture of database platforms, enterprise platforms, and OS platforms, creating an additional complexity hurdle for security personnel

The increasing reliance on cloud technology to host part or all of the corporate database

Database Security

Threats to databases

- Loss of **integrity**
  - Information be protected from improper modification
- Loss of **availability**
  - Available to user or program with legitimate right
- Loss of **confidentiality**
  - Protection of data from unauthorized disclosure

To protect databases against these types of threats four kinds of countermeasures can be implemented:

- **Access control**
- **Inference control**
- **Flow control**
- **Encryption**

Introduction to Database Security

- The security mechanism of a DBMS must include provisions for restricting access to the database as a whole
  - –This function is called **access control** and is handled by creating user accounts and passwords to control login process by the DBMS.
- The security problem associated with databases is that of controlling the access to a **statistical database**, which is used to provide statistical information or summaries of values based on various criteria.
  - –The countermeasures to **statistical database security** problem is called **inference control measures**.
- Another security is that of **flow control**, which prevents information from flowing in such a way that it reaches unauthorized users.
  - –Channels that are pathways for information to flow implicitly in ways that violate the security policy of an organization are called **covert channels**.
- A final security issue is **data encryption**, which is used to protect sensitive data (such as credit card numbers) that is being transmitted via some type communication network.
  - –The data is **encoded** using some **encoding algorithm**.
  - –An unauthorized user who access encoded data will have difficulty deciphering it, but authorized users are given decoding or decrypting algorithms (or keys) to decipher data.

Information Security vs Information Privacy

- Questions of who has what rights to information about individuals for which purposes become more important in this information age
- There is a considerable overlap between issues related to access to resources (security) and issues related to appropriate use of information (privacy).
- Security in information technology refers to many aspects of protecting a system from unauthorized use, including authentication of users, information encryption, access control, firewall policies, and intrusion detection.
- The concept of privacy goes beyond security. Privacy examines how well the use of personal information that the system acquires about a user conforms to the explicit or implicit assumptions regarding that use.
  - Two different perspectives for privacy: preventing storage of personal information versus ensuring appropriate use of personal information.

Database Security and DBA

The DBA (database administrator)'s responsibilities include safeguarding database security. The DBA holds privileged account & among others activities, performs

- –Account creation. This action creates a new account and password for a user or a group of users to enable access to the DBMS.
- –Privilege granting. This action permits the DBA to grant certain privileges to certain accounts.
- –Privilege revocation. This action permits the DBA to revoke (cancel) certain privileges that were previously given to certain accounts.
- –Security level assignment. This action consists of assigning user accounts to the appropriate security clearance level.

Database Logs and Database Audit

The database system must also keep **track of all operations** on the database that are applied by a certain user throughout **each login session**.

– To keep a record of all updates applied to the database and of the particular user who applied each update, we can modify **system log**, which includes an entry for each operation applied to the database that may be required for recovery from a transaction failure or system crash.

If any tampering with the database is suspected, a database audit is performed

– A database audit consists of reviewing the log to examine all accesses and operations applied to the database during a certain time period.

A database log that is used mainly for security purposes is sometimes called an audit trail

BITS Pilani

Pilani | Dubai | Goa | Hyderabad

Discretionary & Mandatory Access Control

Database Security Mechanisms

- A DBMS typically includes a database security and authorization subsystem that is responsible for ensuring the security portions of a database against unauthorized access.
- Two types of database security mechanisms:
  - Discretionary security mechanisms
    - used to grant privileges to users, including the capability to access specific data files, records, or fields in a specified mode (such as read, insert, delete, or update)
  - Mandatory security mechanisms
    - used to enforce multilevel security by classifying the data and users into various security classes (or levels) and then implementing the appropriate security policy of the organization

Discretionary Access Control

The typical method of enforcing **discretionary access control** in a database system is based on the **granting** and **revoking privileges**.

Types of Discretionary Privileges

- The account level:
  - At this level, the DBA specifies the particular privileges that each account holds independently of the relations in the database.
- The relation level (or table level):
  - At this level, the DBA can control the privilege to access each individual relation or view in the database.

The privileges at the **account level** apply to the capabilities provided to the account itself and can include

the **CREATE SCHEMA** or **CREATE TABLE** privilege, to create a schema or base relation;

the **CREATE VIEW** privilege;

the **ALTER** privilege, to apply schema changes such adding or removing attributes from relations;

the **DROP** privilege, to delete relations or views;

the **MODIFY** privilege, to insert, delete, or update tuples;

and the **SELECT** privilege, to retrieve information from the database by using a **SELECT** query.

Mandatory Access Control

- The discretionary access control is an all-or-nothing method, e.g. a user has privilege to access customer data or Not.
  - Business may not be treating all customers equal. Discretionary access control has no direct mechanism to handle such situation.
- Many applications require additional security policy that classifies data and users based on security classes. This is called Mandatory Access Control.
  - This approach as mandatory access control, would typically be combined with the discretionary access control mechanisms
- To incorporate **multilevel security** notions into the relational database model, it is common to consider attribute values and tuples as data objects.
- Hence, each attribute A is associated with a classification attribute C in the schema, and each attribute value in a tuple is associated with a corresponding security classification.
- In addition, in some models, a tuple classification attribute TC is added to the relation attributes to provide a classification for each tuple as a whole.
- Hence, a multilevel relation schema R with n attributes would be represented as

  R(A1,C1,A2,C2, …, An,Cn,TC)

  where each Ci represents the classification attribute associated with attribute Ai.

Security Classes - Mandatory Access Control and Multilevel Security

- Typical security classes are top secret (TS), secret (S), confidential (C), and unclassified (U), where TS is the highest level and U the lowest: TS ≥ S ≥ C ≥ U

- The commonly used model for multilevel security, known as the Bell-LaPadula model, classifies each subject (user, account, program) and object (relation, tuple, column, view, operation) into one of the security classifications, T, S, C, or U:
  - Represent clearance (classification) of a subject S as class(S) and to the classification of an object O as class(O).

- A multilevel relation will appear to contain different data to subjects (users) with different clearance levels.
  - In some cases, it is possible to store a single tuple in the relation at a higher classification level and produce the corresponding tuples at a lower-level classification through a process known as **filtering**.
  - In other cases, it is necessary to store two or more tuples at different classification levels with the same value for the **apparent key**.
    - The apparent key of a multilevel relation is the set of attributes that would have formed the primary key in a regular (single-level) relation

- This leads to the concept of **polyinstantiation** where several tuples can have the same apparent key value but have different attribute values for users at different classification levels.

Confidentiality - Bell-LaPadula Model

- A confidentiality policy prevents the unauthorized disclosure of information.
  - e.g., the navy must keep confidential the date on which a troop ship will sail.
- Bell-LaPadula Model is a multi-level model proposed by Bell and LaPadula of MITRE for enforcing access control in government and military applications
- It uses military-style classifications. Subjects and Objects are often partitioned into different security levels
- A subject can only access objects at certain levels determined by his/her/its security level

The Model

- The confidentiality classification is a set of security clearances arranged in a linear (total) ordering.
- Clearances represent the security levels. The higher the level, the more sensitive the info.
  - A subject has a security clearance. An object has a security classification. While referring to both subject clearances and object classifications, the term “classification” is used

TOP SECRET (TS)	Tamara, Thomas	Personnel Files
SECRET (S)	Sally, Samuel	Electronic Mail Files
CONFIDENTIAL (C)	Claire, Clarence	Activity Log Files
UNCLASSIFIED (UC)	Ulaley, Ursula	Telephone List Files

The basic confidentiality classification system. The four security levels are arranged with the most sensitive at the top and the least sensitive at the bottom.

Security Requirements - Bell-LaPadula Model

Let L(S)=ls be the security clearance of subject S.

Let L(O)=lo be the security classification of object).

For all security classification l_i, i=0,…, k-1, $l_i < l_{i+1}$

Simple Security Condition:
S can read O if and only if $l_o <= l_s$ and
S has discretionary read access to O.

*-Property (Star property):
S can write O if and only if $l_s <= l_o$ and
S has discretionary write access to O.

TS subject can not write documents lower than TS.

– Prevent classified information leak.

BITS Pilani

Pilani | Dubai | Goa | Hyderabad

Inference Control

Statistical Database Security

- Statistical databases are used mainly to produce statistics on various populations.
- The database may contain confidential data on individuals, which should be protected from user access.
- Users are permitted to retrieve statistical information on the populations, such as averages, sums, counts, maximums, minimums, and standard deviations.
- A population is a set of tuples of a relation (table) that satisfy some selection condition.
- Statistical queries involve applying statistical functions to a population of tuples.

Statistical Queries

- Statistical database security techniques must prohibit the retrieval of individual data.
- This can be achieved by prohibiting queries that retrieve attribute values and by allowing only queries that involve statistical aggregate functions such as COUNT, SUM, MIN, MAX, AVERAGE, and STANDARD DEVIATION.
    - Such queries are sometimes called statistical queries.

Inference Control

- It is DBMS's responsibility to ensure confidentiality of information about individuals, while still providing useful statistical summaries of data about those individuals to users. Provision of privacy protection of users in a statistical database is paramount.
- In some cases it is possible to infer the values of individual tuples from a sequence statistical queries.
  - This is particularly true when the conditions result in a population consisting of a small number of tuples.

Consider the following statistical queries:

Q1: SELECT COUNT (*) FROM PERSON
WHERE <*condition*>;

Q2: SELECT AVG (Income) FROM PERSON
WHERE <*condition*>;

Suppose that we are interested in finding the Salary of Jane Smith, and we know that she has a Ph.D. degree and that she lives in the city of Bellaire, Texas.

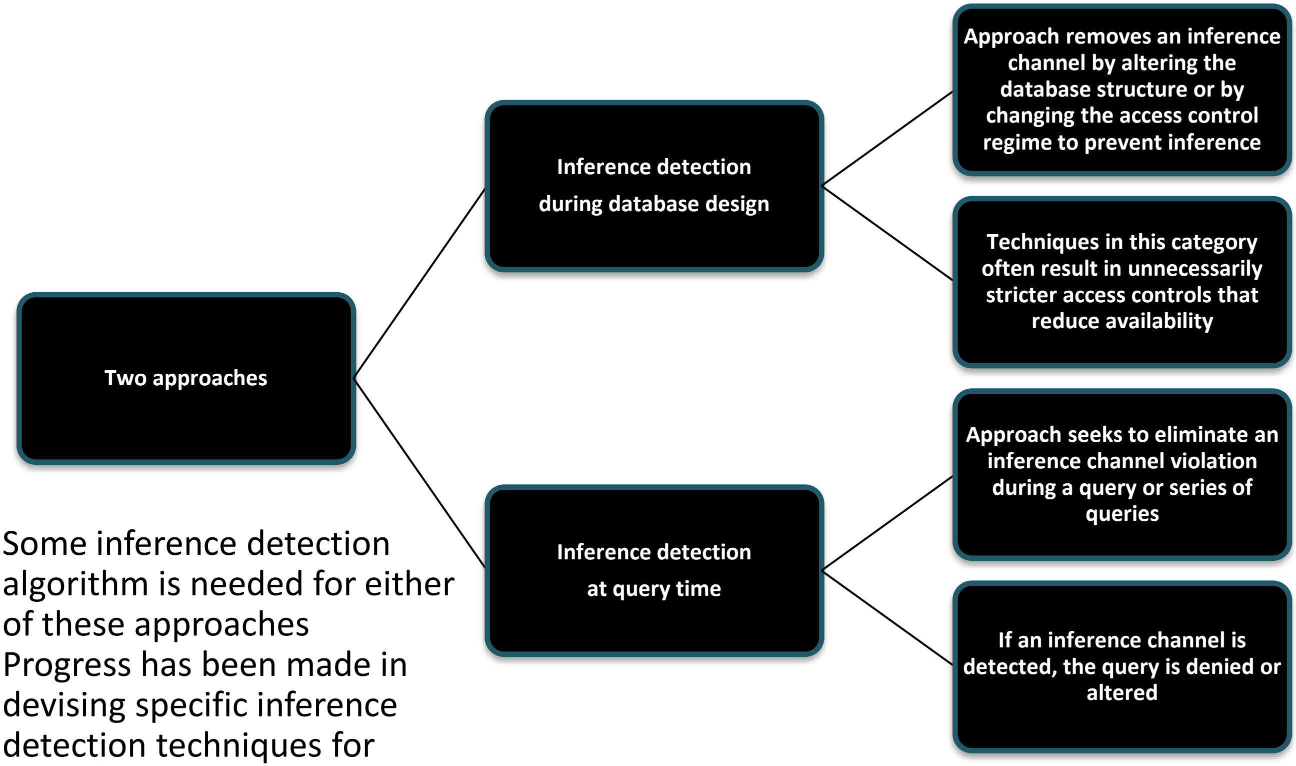
We issue the statistical query Q1 with the following condition:

(Last_degree='Ph.D.' AND Sex='F' AND City='Bellaire' AND State='Texas')

If we get a result of 1 for this query, we can issue Q2 with the same condition and find the Salary of Jane Smith

Even if Q1 gives higher than 1, we may be able to infer if the number is small

Inference Detection



Two approaches

Inference detection during database design

Approach removes an inference channel by altering the database structure or by changing the access control regime to prevent inference

Techniques in this category often result in unnecessarily stricter access controls that reduce availability

Inference detection at query time

Approach seeks to eliminate an inference channel violation during a query or series of queries

If an inference channel is detected, the query is denied or altered

- Some inference detection algorithm is needed for either of these approaches
- Progress has been made in devising specific inference detection techniques for multilevel secure databases and statistical databases

Inference Control

The risk of inferring individual information from statistical queries is reduced if no statistical queries are permitted

- Whenever the number of tuples in the population specified by the selection condition falls below some threshold.
- Another technique for prohibiting retrieval of individual information is to prohibit sequences of queries that refer repeatedly to the same population of tuples.
- Another technique is partitioning of the database. Partitioning implies that records are stored in groups of some minimum size; queries can refer to any complete group or set of groups, but never to subsets of records within a group.

BITS Pilani

Pilani | Dubai | Goa | Hyderabad

Flow Control

Introduction to Flow Control

- **Flow control** regulates the distribution or flow of information among accessible objects.
- A **flow** between object X and object Y occurs when a program reads values from X and writes values into Y.
  - Flow controls check that information contained in some objects does not flow explicitly or implicitly into less protected objects.
- A **flow policy** specifies the channels along which information is allowed to move.
  - The simplest flow policy with just two classes of information:
    - confidential (C) and nonconfidential (N)
  - allows all flows except those from class C to class N.

Flow Control & Access Control

- Access control mechanisms help in Flow control.
- Flow controls can be enforced by an extended access control mechanism, which involves assigning a security class (usually called the clearance) to each running program.
  - The program is allowed to read a particular memory segment only if its security class is as high as that of the segment.
  - It is allowed to write in a segment only if its class is as low as that of the segment.
- This automatically ensures that no information transmitted by the person can move from a higher to a lower class.
  - For example, a military program with a secret clearance can only read from objects that are unclassified and confidential and can only write into objects that are secret or top secret.

Covert Channels

- A **covert channel** allows a transfer of information that violates the security or the policy.
- A **covert channel allows** information to pass from a higher classification level to a lower classification level through **improper means**.
- **Covert channels** can be classified into two broad categories:
  - **Storage channels** do not require any temporal synchronization, in that information is conveyed by accessing system information or what is otherwise inaccessible to the user.
  - **Timing channel** allow the information to be conveyed by the timing of events or processes.

Covert Channels - Challenges

- Difficult to detect
- Can operate for a long time and leak a substantial amount of classified data
- Can compromise an otherwise secure system, including one that has been formally verified.

Covert Channels - Examples

- Two virtual machines share cylinders 100 through 200 on a disk. The disk uses a SCAN algorithm to schedule disk accesses. One virtual machine has security class *High*, and the other has class *Low*.
- A process on the *High* machine should not send information to a process on the *Low* machine.
- The process on the *Low* machine reads data on cylinder 150. When that request completes, it relinquishes the CPU. The process on the *High* machine runs, issues a seek to cylinder 140, and relinquishes the CPU. The process on the *Low* machine runs and issues seek requests to cylinders 139 and 161.
- Because the disk arm is moving over the cylinders in descending order, the seek issued to cylinder 139 is satisfied first, followed by the seek issued to cylinder 161. This knowledge of ordering is a bit of information.

Covert Channels - Examples

- A programmer for a bank has no need to access the names or balances in depositors’ accounts.
- Programmers for brokerage firms do not need to know what buy and sell orders exist for clients.
- During program testing, access to a form of real data or some sample test data may be justifiable, but not after the program has been accepted for regular use.
- This represents information leakage thru covert channel.

BITS Pilani

Pilani | Dubai | Goa | Hyderabad

Injection Attacks

Injection Attacks

- The injection is a mechanism used by an attacker to introduce (or "inject") code into a vulnerable computer program and change the course of execution
- Code injection vulnerabilities occur when an application sends untrusted data to an interpreter – An input validation failure

Injection flaws are most often found in

- SQL,
- LDAP,
- XPath,
- NoSQL queries,
- OS commands,
- XML parsers,
- Program arguments,
- etc.

Even XSS is really just a form of HTML injection

Injection Attacks

- Injection attacks target those parsers (Every interpreter has a parser)
  - Trick the parser into interpreting data as commands
- Real world parsers have many corner cases and flaws that may not match the spec.
- Typically the interpreters run with a lot of access, so a successful attack can easily result in significant data breaches
- Injection attacks are at the top of OWASP top 10.

BITS Pilani

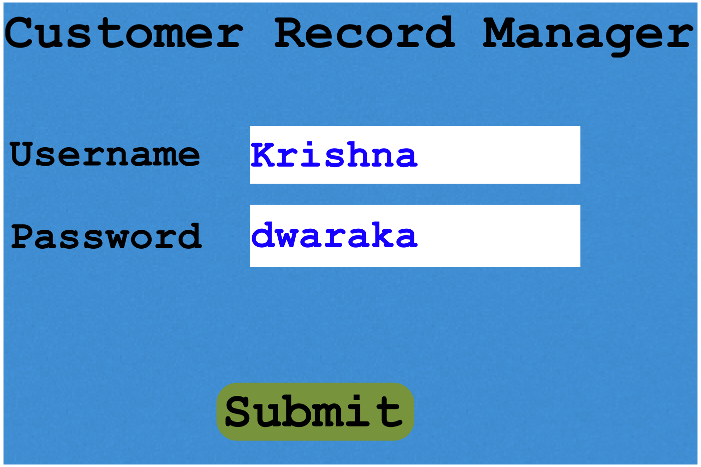
Pilani | Dubai | Goa | Hyderabad

SQL Injection

SQL Injection - Introduction

- Web applications (that access a database) often have 2-way communication with database
  - can send commands and data to the database
  - display data retrieved from the database through the Web browser.
- In an SQL Injection attack, the attacker injects a (malicious)string input through the application, which changes or manipulates the SQL statement to the attacker’s advantage.
- An SQL Injection attack can harm the database in various ways, such as
  - unauthorized manipulation of the database, or
  - retrieval of sensitive data.
  - It can also be used to execute system level commands
- Most common attack goal is bulk extraction of data

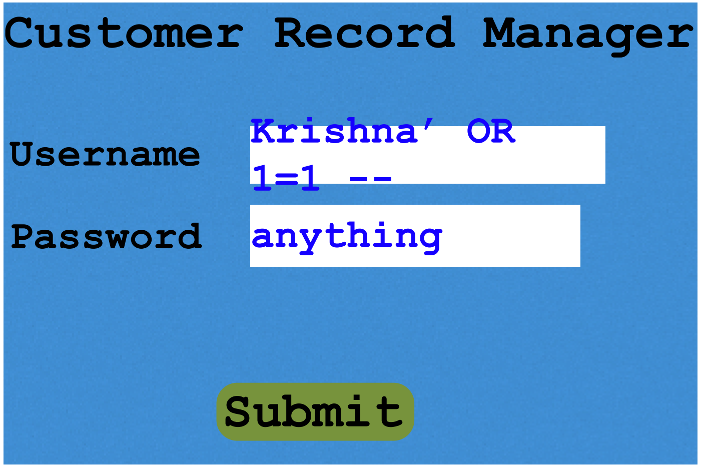
SQL Injection : Typical Query



SELECT * FROM Customers WHERE username = 'Krishna' AND password = 'dwaraka'

Krishna's customer entries are displayed
No harm done

SQL Injection : Malicious Query

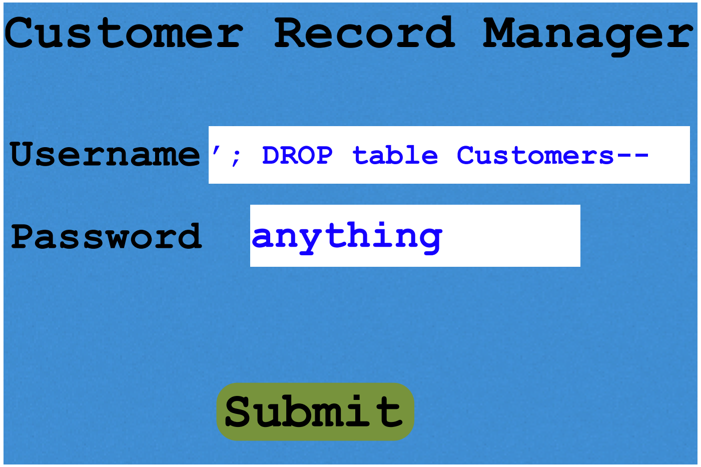

Customer Record Manager

Username	Krishna' OR 1=1 --
Password	anything

SELECT * FROM Customers WHERE username = 'Krishna' OR 1=1 -- AND password = 'anything'

All customer entries are displayed
Confidentiality lost.

SQL Injection : Malicious Query



SELECT * FROM Customers WHERE username = '''; DROP table Customers -- AND password = 'anything'

All customer entries are eliminated.
Availability lost.

SQL Injection - Function Call Injection

- A database function or operating system function call may be inserted into a vulnerable SQL statement to manipulate the data or make a privileged system call
  – e.g., the dual table is used in the FROM clause of SQL in Oracle when a user needs to run SQL that does not logically have a table name.
- Here, TRANSLATE is used to replace a string of characters with another string of characters.

  SELECT TRANSLATE ('user input', 'from_string', 'to_string') FROM dual;
- This type of SQL statement can be subjected to a function injection attack

  SELECT TRANSLATE ('' || UTL_HTTP.REQUEST ('http://129.107.2.1/') || '', '98765432', '9876') FROM dual;

  –The attacker could make the server access a URL to get (possibly privileged) content.

Protection against SQL Injection

- Certain programming rules to all Web-accessible procedures and functions (to protect from SQL Injection)
- Bind Variables (Using Parameterized Statements). The use of bind variables (Instead of embedding the user input into the statement) protects against injection attacks). For e.g. in Java and JDBC:

```
PreparedStatement stmt = conn.prepareStatement( “SELECT * FROM EMPLOYEE
WHERE EMPLOYEE_ID=? AND PASSWORD=?”);
stmt.setString(1, employee_id);
stmt.setString(2, password);
```

- Filtering Input (Input Validation): remove escape characters(non-whitelisted characters) from input strings by using the SQL Replace function.
- Function Security: Database functions, both standard and custom, should be restricted, as they can be exploited in the SQL function injection attacks.

Some Serious SQL Injection

- On August 17, 2009, the United States Department of Justice charged an American citizen, Albert Gonzalez, and two unnamed Russians with the theft of 130 million credit card numbers using an SQL injection attack. In reportedly "the biggest case of identity theft in American history", the man stole cards from a number of corporate victims after researching their payment processing systems.
- In October 2015, SQL injection was believed to be used to attack the British telecommunications company Talk Talk's servers, stealing the personal details of up to four million customers.

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

What do you think about NoSQL Database Security?

BITS Pilani

Pilani | Dubai | Goa | Hyderabad

Injection Attacks

Injection Attacks

- SQL injection is one type of Injection Attacks
- Injection Attack is listed as the number one web application security risk in the OWASP Top 10
- In an injection attack, an attacker supplies malicious input to a program. This input gets processed by an interpreter as part of a command or query. In turn, this alters the execution of that program
- The primary reason for injection vulnerabilities is insufficient user input validation
- Injection attack can result in data theft, data loss, loss of data integrity, denial of service, as well as full system compromise

Injection Attacks

Some of the well-known types of Injection Attacks are

- SQL injection
- Cross-site Scripting (XSS) – JavaScript injection
    - The attacker injects an arbitrary script (usually in JavaScript) that is then executed inside the victim’s browser.
- LDAP Injection
- OS Command Injection
- Email Header Injection
    - The attacker sends IMAP/SMTP commands to a mail server
- Code injection
- XPath injection
    - The attacker injects data to execute crafted XPath queries

Injections – Why They Happen?

Software environment has large number of interpreters

- SQL, LDAP, Operating System, XPath, XQuery, and many more

Interpreters run with a lot of access, so a successful attack can result in significant breaches

- Subtleties of data flow, parsers, contexts, capabilities, and escaping make it difficult to detect
- Every interpreter requires a parser supporting a grammar
  - Injection attackers trick the parsers into interpreting data as commands
- Real-world parsers may not have well-defined grammar, may have flaws, and misbehaving corner cases

Computer Security: Principles and Practice by William Stallings, and Lawrie Brown Pearson, 2008.
Ramez Elmasri & Shamkant B. Navathe, Fundamentals of Database Systems, Pearson Education, 7th Edition, 2017
sei.cmu.edu/cert
www.owasp.com
www.cigital.com

Thank You!