

DATABASE SECURITY

Notes - Set1

Course Objectives

- Specifically, what are we trying to protect?
- How are we planning to protect?
- These questions form the basis for discussions on database security
- **Security Issues**
 - What are we trying to protect by ensuring database security?
 - What levels of information need to be safeguarded and how?
 - What are the types of problems and threats that deserve special attention?
 - Can we distinguish between threats from outside and internal threats?
 - Do these require different types of protection mechanisms?
 - What are the solution options?
 - How is protection of privacy related to database security?

Course Goals

- Three broad goals of database security
 - Denial of access to the database by unauthorized users
 - Guarantee of access to all authorized users
 - Protection of privacy of data

What is Database Security?

- Database security refers to the various measures organizations take to ensure their databases are protected from internal and external threats.
- Database security includes
 - Protecting the database itself,
 - The data it contains,
 - Its database management system, and
 - The various applications that access it
- Effective database security is key for remaining compliant, protecting organizations' reputations, and keeping their customers.
- Database security refers to the range of tools, controls, and measures designed to establish and preserve database confidentiality, integrity, and availability.

Why is it important?

- **Compromised intellectual property:** Your intellectual property—trade secrets, inventions, proprietary practices—may be critical to your ability to maintain a competitive advantage in your market. If that intellectual property is stolen or exposed, your competitive advantage may be difficult or impossible to maintain or recover.
- **Damage to brand reputation:** Customers or partners may be unwilling to buy your products or services (or do business with your company) if they don't feel they can trust you to protect your data or theirs. A 2018 IBM-sponsored Harris Poll survey of adults aged 18+ revealed that [63% rate quality of data protection against cyberattacks as "extremely important"](#) in their decision to purchase from a company.
- **Business continuity (or lack thereof):** Some business cannot continue to operate until a breach is resolved.
- **Fines or penalties for non-compliance:** The financial impact for failing to comply with global regulations such as the Sarbanes-Oxley Act (SAO) or Payment Card Industry Data Security Standard (PCI DSS), industry-specific data privacy regulations such as HIPAA, or regional data privacy regulations, such as Europe's General Data Protection Regulation (GDPR) can be devastating, with fines in the worst cases exceeding several million dollars *per violation*.
- **Costs of repairing breaches and notifying customers:** In addition to the cost of communicating a breach to customer, a breached organization must pay for forensic and investigative activities, crisis management, triage, repair of the affected systems, and more.

Databases often contain extremely sensitive information that must be protected from security vulnerabilities and exploits.

Database System (DBS)

- Collection of
 - Interrelated data and
 - Set of programs to access the data
- Convenient and Efficient processing of data using Database Management System (DBMS)
- Database Application Software

Abstraction

- View level: Different perspectives
 - Application programs hide irrelevant data
- Logical level: Data models
 - Logical representation of data
 - Different approaches: hierarchical, network, object oriented, semi-structured, etc.
 - Data independence principle
- Physical level: how data is stored

How Relational Databases Work

- Relational databases use a hierarchical system of tables to store information as opposed to a flat file.
- Data is organized in a structured manner using rows and columns.
- In relational databases, data is stored as “objects”.
- There are many database objects and they can be identified from views such as these:
 - Dba_objects (Oracle)
 - Sysobjects (MS SQL)

Significant Database Objects

- The more important objects that have security and controls significance include the following:
 - Tables: Database entity that contains rows and columns with a primary key which uniquely identifies each row.
 - Views: This represents data that a user can access and it is an important security mechanism
 - Stored Procedures: Business logic in the form of pre-compiled SQL statements that perform specific functions.
 - Triggers: These are typically a block of SQL statements that is executed on a table following a pre-determined event. Sometimes used for audit trails

What are the Different Categories of Databases?

- **End User Database** – The use of this kind of database is relative to end-users who consider software or application only as their work environment. End-User Databases mostly use shared databases to complete the demand of end-users only. The primary goal is to set up and complete all the requirements of an end-user.
- **Personal Database** – When data needs reside for small management or a group and data is preserved in the personal computer only. Personal Databases mostly used in short term project goals.
- **Centralized Database** – Remote access to data, data at distinct locations and Database at one location, all three makes a database centralized. Users from all locations had access to this centralized database that can be accessed at any time. A local area handler is the best example of this centralized database where procedures are followed to complete design flow.
- **Distributed Database** – This database is opposite in the implementation of centralized databases. The data in these databases is not centralized to one location (physical) but at different physical locations. All these locations are connected via some procedural communication links. They are designed to store and retrieve data faster.
- **Operational Database** – Business centric operations and flow are based on operational databases such as Customer Relationship Management and Enterprise resource planning software. CRMs and ERPs use functional kinds of databases.
- **Relational Database** – When data needed to fit into the predefined category of tables where schema, storage types and data types are present, and data is structured, relational databases are used as they are easy to extend, join and many standard and straightforward operations are easy to apply on those tables.
- **NoSQL Database** – Relational databases were not useful in solving [big data](#)-related issues but [NoSQL](#) databases resolved those issues, and moreover data from different distributed locations of cloud can also be accessed within NoSQL and Data doesn't need to be structured only.
- **Cloud Database** – When Scalability, storage cost, and bandwidth are essential, cloud databases come with the super solution of this. Cloud databases are kind of a virtual environment where data of all types can be stored, and moreover, big [data operations](#) are efficient to perform on these databases. The logic behind these is Software as a service to Database as a service.

Enterprise Databases

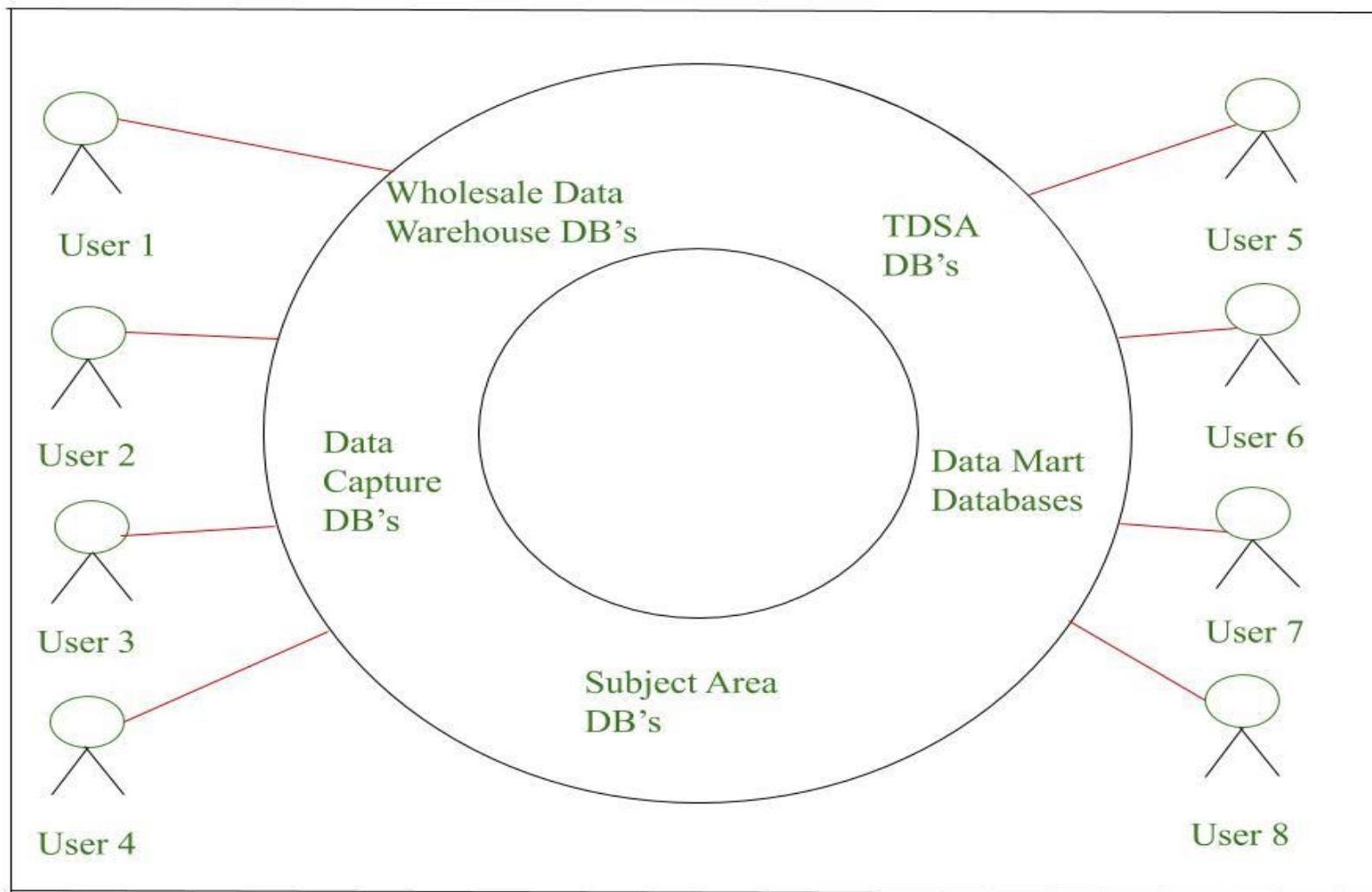
- Business databases are used by companies to strategize, schedule, and standardize various activities which company is going to carry on in coming time and which will help company to improve its productivity as well.
- An **enterprise DBMS** is such in which 100 to 10, 000 individuals can access simultaneously. Businesses and big companies use this to handle their vast data set. Such database allows businesses to increase their productivity. This kind of database can handle large organizations with thousands of employees and busy web server with lakhs of people accessing it simultaneously online.
- Typically DBMS is managed by Database administrator, or DBA, who is specialist in particular software product. DBA instructs-system to load, retrieve, or change data in database, as well as tells who can access data and what commands each one can use.
- Database servers are the most important systems in virtually all organizations.
- They store critical information that supports business including the following:
 - Email
 - Financial Data
 - Sales Data
 - Personnel Data
 - Intellectual Property
 - Operation and
 - Security Data, etc.

Modern Databases are created using Structured Query Language (SQL) which is the standard for database interoperability.

Enterprise Database Environment

- These databases provide an interconnected collection of **five types of database architecture**: original data capture, transaction data staging area, subject area, wholesale data warehouses, and data marts.
- All of these databases represent one or more functional areas that are present in every enterprise in general.
 - **Original data Capture** : Original data capture stores information about ongoing applications to database. Data can be created by any individual, or may it can be result of software development.
 - **TDSA** : TDSA stands for transaction data staging area that smoothes all various semantics, times, and units that can occur in original data capture packages since they come from different users, who run under different operating systems.
 - **Subject area database** : Subject area database draw data from one or more TDSA databases and create databases of large subject areas. Overall number of subject area databases is equal to number of independent user resources
 - **Warehouse Database** : A warehouse database is one that holds data taken from several different databases in subject field. Data in data centers is taken from databases of subject region.
 - **Data Mart Database** : data mart database is ad hoc and is created for specific need. Its architecture, loading software, and reporting from it are all custom created. Datamart database volumes can also be limited to wide range of data needed for individual offices or even individuals and can be downloaded Weekly or even nightly, to that office or person.

Enterprise Database Environment



Enterprise Database Environment

- **Various Enterprise Database Management System :** There are many enterprise databases such as:
 - Oracle Database 18c
 - Microsoft SQL Server
 - [IBM DB2](#)
 - Teradata (NCR)
 - SAP Sybase ASE
 - [PostgreSQL](#)
 - MySQL
 - [MariaDB Enterprise etc](#)

Features of Enterprise Database Management System :

- **Parallel query :** At the same time, several users will position queries in there. All the questions are responded to simultaneously.
- **Multi-process support :** Several processes can be handled by splitting work load between them all.
- **Clustering features :** That is, it combines more than one server or single database connecting case. Often one server can not be sufficient to handle data volume, so this is time where this function comes into play.

Current Demands

- Efficient data processing of large data sets
- Long running transactions
- Real-time demand
- Usability for specific applications
- ...

Database Protection -Why?

- Data are an important strategic and operational asset for any organization
- Damages and misuses of data, stored by a DBMS, affect not only a single user or an application; they may have disastrous consequences on the entire organization
- Additionally, the advent of the Internet as well as networking capabilities has made the access to data and information much easier

Database Security: Examples

- Consider a payroll database in a corporation, it must be ensured that:
 - salaries of individual employees are not disclosed to arbitrary users of the database
 - salaries are modified by only those individuals that are properly authorized
 - paychecks are printed on time at the end of each pay period
- In a military environment, it is important that:
 - the target of a missile is not given to an unauthorized user
 - the target is not arbitrarily modified
 - the missile is launched when it is fired

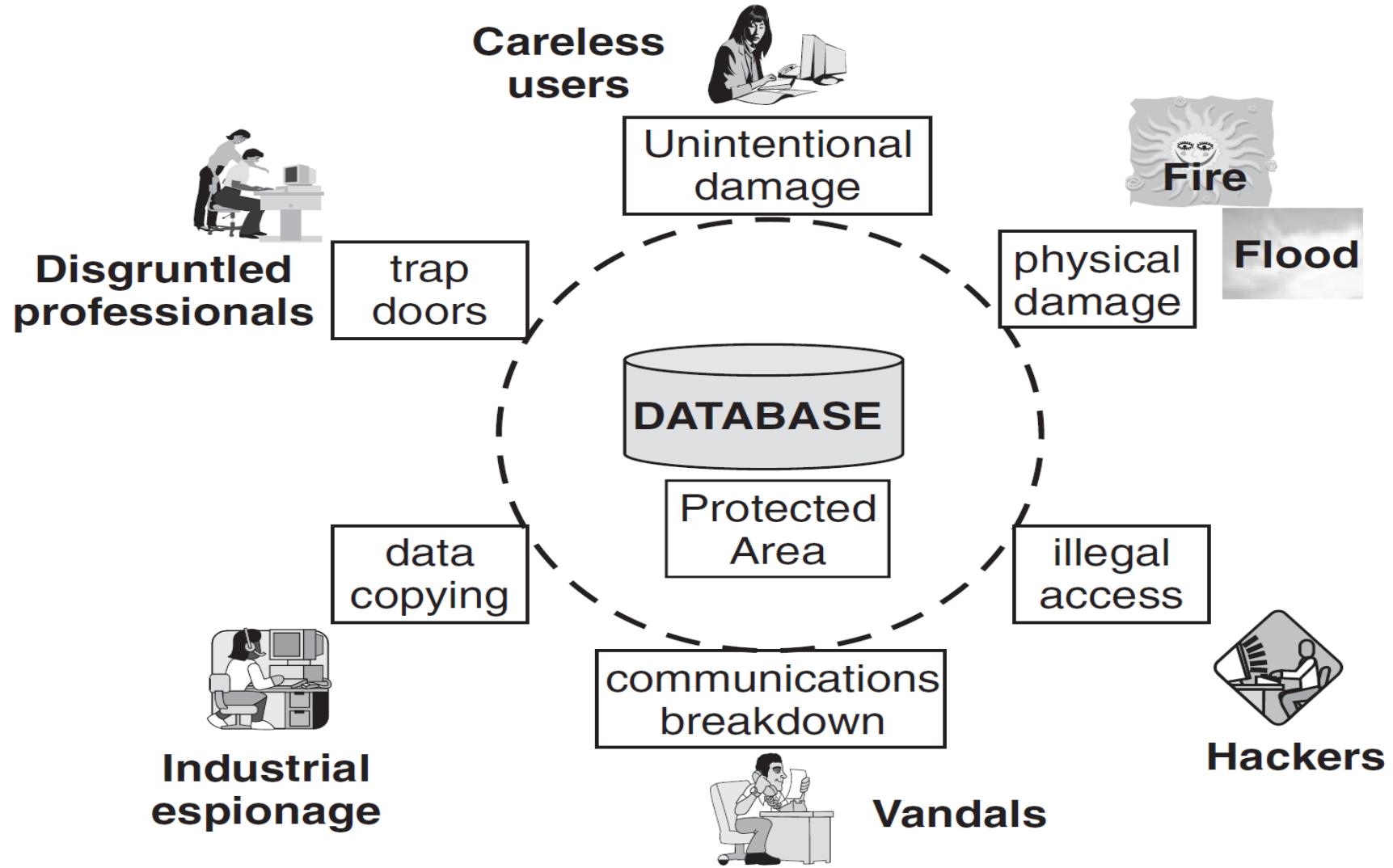
The types of security exposure in a Database Environment

- **Natural disasters:** Fire, floods, and other such catastrophes.
- **Human carelessness:** Unintended damage caused by authorized users, especially while running jobs in batch.
- **Malicious damage:** Sabotage, vandalism, actions of malicious programmers, technical support staff, and persons performing database administration functions.
- **Crime:** Theft, embezzlement, industrial espionage, and employees selling a company's secrets and data for mailing lists.
- **Privacy invasion:** Casual curiosity, data lookup by competitors, obtaining data for political or legal reasons.

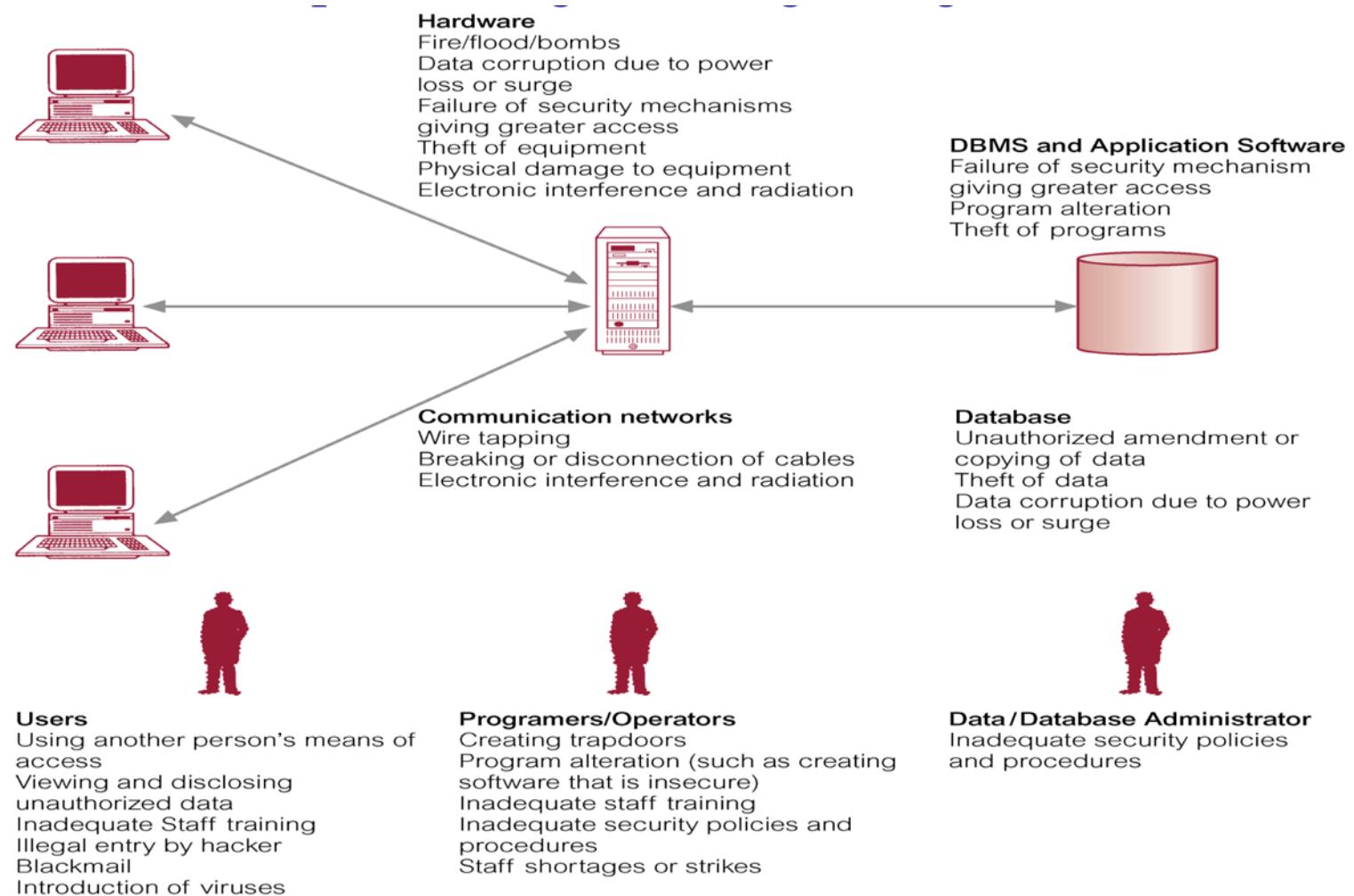
Security Threats

- Poor design
- Insufficient quality control
- Accidents
- Attacks

Threats to Database Security



Threats to Database Security



Most common areas of database security weaknesses

- **1. Deployment Fail**

- The most common cause of database vulnerabilities is the lack of care with which they are deployed. Sure, databases are often functionally tested to make sure they provide core functions for calling applications.
- In fact, the majority of pre-deployment tests are designed to verify that a database is doing what it should do; very few are checking to ensure that it isn't doing something it should not do.
- Every relational database platform (including Oracle, DB2, SQL Server, Sybase, Postgres, and MySQL) is insecure after a fresh installation, and it will remain that way until you fix it.

Most common areas of database security weaknesses

- **2. Broken Databases**

- Hackers make their living by finding and targeting vulnerabilities in all kinds of software, including database management software.
- All major commercial database software vendors and open source database management platforms issue regular security patches to address these vulnerabilities, but failure to apply these patches in a timely fashion can increase your exposure.
- A database-specific threat, these involve the insertion of arbitrary SQL or [non-SQL](#) attack strings into database queries served by web applications or HTTP headers.
- Organizations that don't follow secure web application coding practices and perform regular vulnerability testing are open to these attacks.
- Buffer overflow occurs when a process attempts to write more data to a fixed-length block of memory than it is allowed to hold. Attackers may use the excess data, stored in adjacent memory addresses, as a foundation from which to launch attacks.
- Malware is software written specifically to exploit vulnerabilities or otherwise cause damage to the database. Malware may arrive via any endpoint device connecting to the database's network.
- The [SQL Slammer worm](#) of 2003 was able to infect more than 90 percent of vulnerable computers within 10 minutes of deployment, taking down thousands of databases in minutes.
- This worm took advantage of a bug that was discovered in Microsoft's SQL Server database software the previous year, but few system administrators installed a fix, leaving computers vulnerable.
- This worm exploited a buffer-overflow vulnerability and allowed for an attacker to crash, or gain control over, any database it discovered.
- it was the catalyst that pushed vendors to start offering regular security patches
- unbelievably, many companies don't install security patches, leaving their database systems vulnerable to attack and often subject to complete compromise.
- It's true that it takes time to test patches, but most patches are released on a regular schedule -- often every three months.
- Vulnerabilities like command injection and buffer overflows don't make headlines like they used to; fewer issues are found, and vendors are fairly responsive with patches when they are.

Most common areas of database security weaknesses

- **3. Distributed Denial of Services**

- In a denial of service (DoS) attack, the attacker deluges the target server—in this case the database server—with so many requests that the server can no longer fulfill legitimate requests from actual users, and, in many cases, the server becomes unstable or crashes.
- In a distributed denial of service attack (DDoS), the deluge comes from multiple servers, making it more difficult to stop the attack.
- Load/stress testing and capacity testing of a database to ensure it does not crash in a distributed denial of service (DDoS) attack or user overload.

Most common areas of database security weaknesses

- **4. Leaked Data**

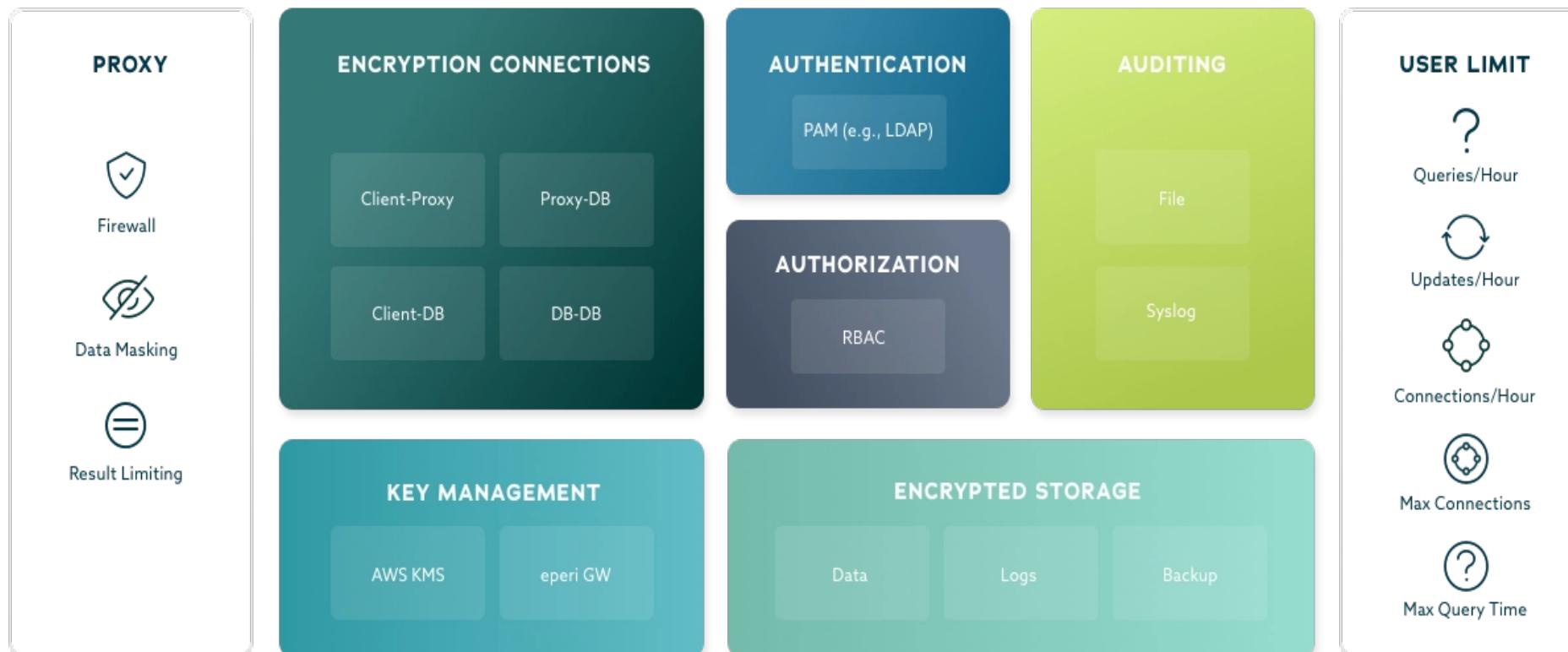
- Accidents, weak passwords, password sharing, and other unwise or uninformed user behaviors continue to be the cause of [nearly half \(49%\) of all reported data breaches](#).
- Some DBAs forget about network security.
- The common mindset is that the databases are in the "back office," a network secured from the Internet, so data communications to and from databases don't have to be encrypted.
- What these IT pros are forgetting -- or ignoring -- is the networking interface of their database. But make no mistake: It's trivial for an attacker to capture network traffic and parse interesting data from multiple user connections to the database -- in essence, seeing all data moving in and out.
- In all cases, you should enable Transport Layer Security.
- Secure Sockets Layer has minimal impact on network performance and makes it very difficult for someone to collect data from the wire.
- Most relational platforms provide SSL- or TLS-encrypted communications as part of the basic database package, enabled through a simple configuration setting change.
- Databases also contain a networking interface, and so hackers are able to capture this type of traffic to exploit it. To avoid such a pitfall, administrators should use SSL- or TLS-encrypted communication platforms.

Most common areas of database security weaknesses

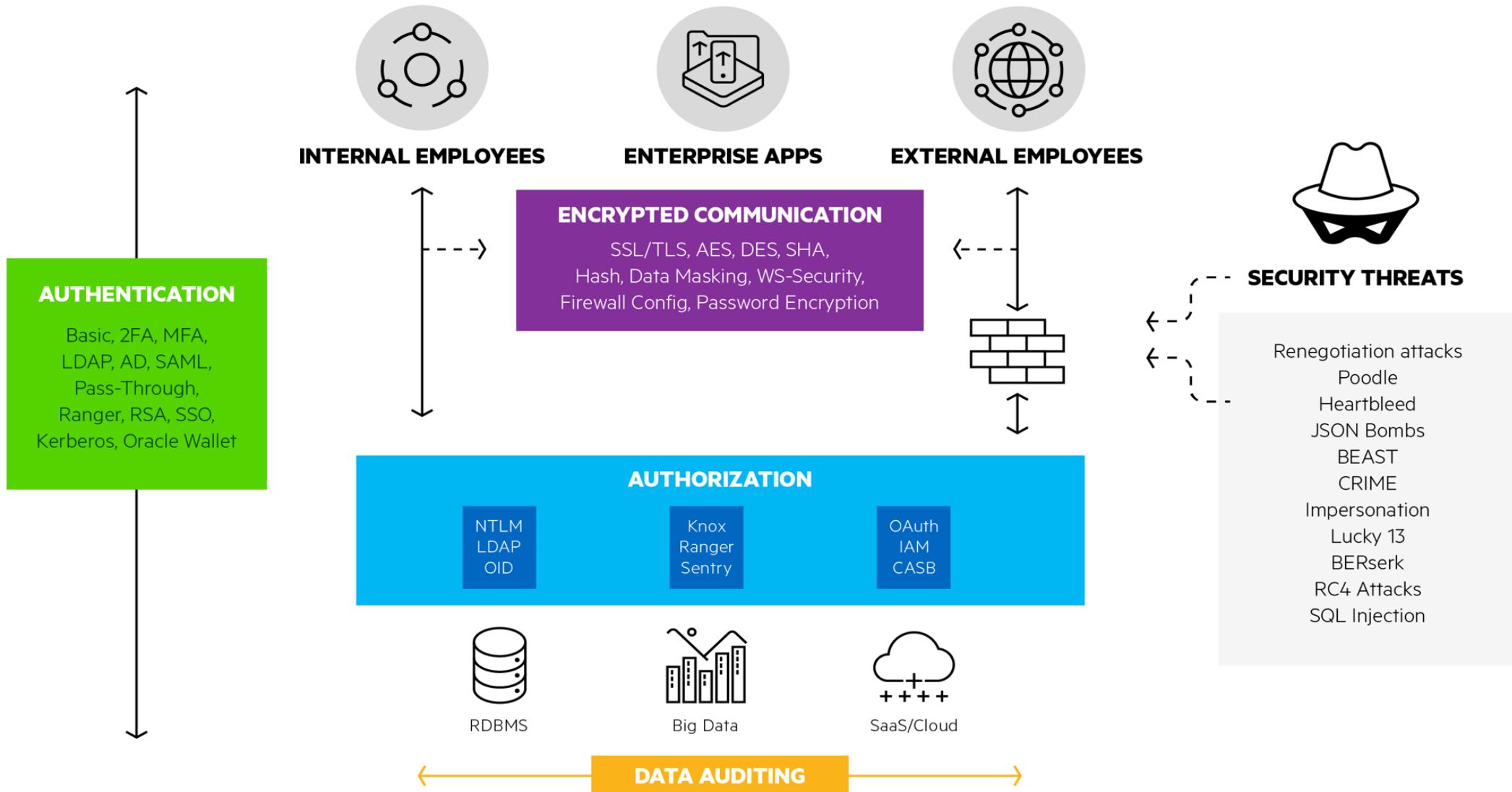
- **5. Stolen database backups**
 - External attackers who infiltrate systems to steal data are one threat, but what about those inside the corporation?
 - The report suggests that insiders are also likely to steal archives — including database backups — whether for money, profit or revenge.
 - This is a common problem for the modern enterprise, and businesses should consider encrypting archives to mitigate the insider-risk.
- Attacks on Backups
 - Organizations that fail to protect backup data with the same stringent controls used to protect the database itself can be vulnerable to attacks on backups.
 - These threats are exacerbated by the following:
 - **Growing data volumes:** Data capture, storage, and processing continues to grow exponentially across nearly all organizations. Any data security tools or practices need to be highly scalable to meet near and distant future needs.
 - **Infrastructure sprawl:** [Network environments](#) are becoming increasingly complex, particularly as businesses move workloads to [multicloud](#) or [hybrid cloud](#) architectures, making the choice, deployment, and management of security solutions ever more challenging.
 - **Increasingly stringent regulatory requirements:** The worldwide regulatory compliance landscape continues to grow in complexity, making adhering to all mandates more difficult.
 - **Cybersecurity skills shortage:** Experts predict there may be as many as [8 million unfilled cybersecurity positions by 2022](#).

Cloud-based databases need new approaches to ensure data security

- Database-as-a-Service (DBaaS) offers high availability, multi-tenancy and the ability to scale as demand grows without the high cost of buying new hardware and related maintenance costs.



Data Security Landscape



Top Database Security Threats

- Privilege Abuse
 - Abusing legitimate privileges for unauthorized purposes
 - Excessive privileges that exceeds job function requirement
- Weak Authentication
 - Weak or ineffective password policies
 - Theft of login credentials, social engineering
 - Poor encryption
- Weak Systems Configuration
 - Use of default Configurations
 - Installation of improper tools and services
 - Lack of security baseline

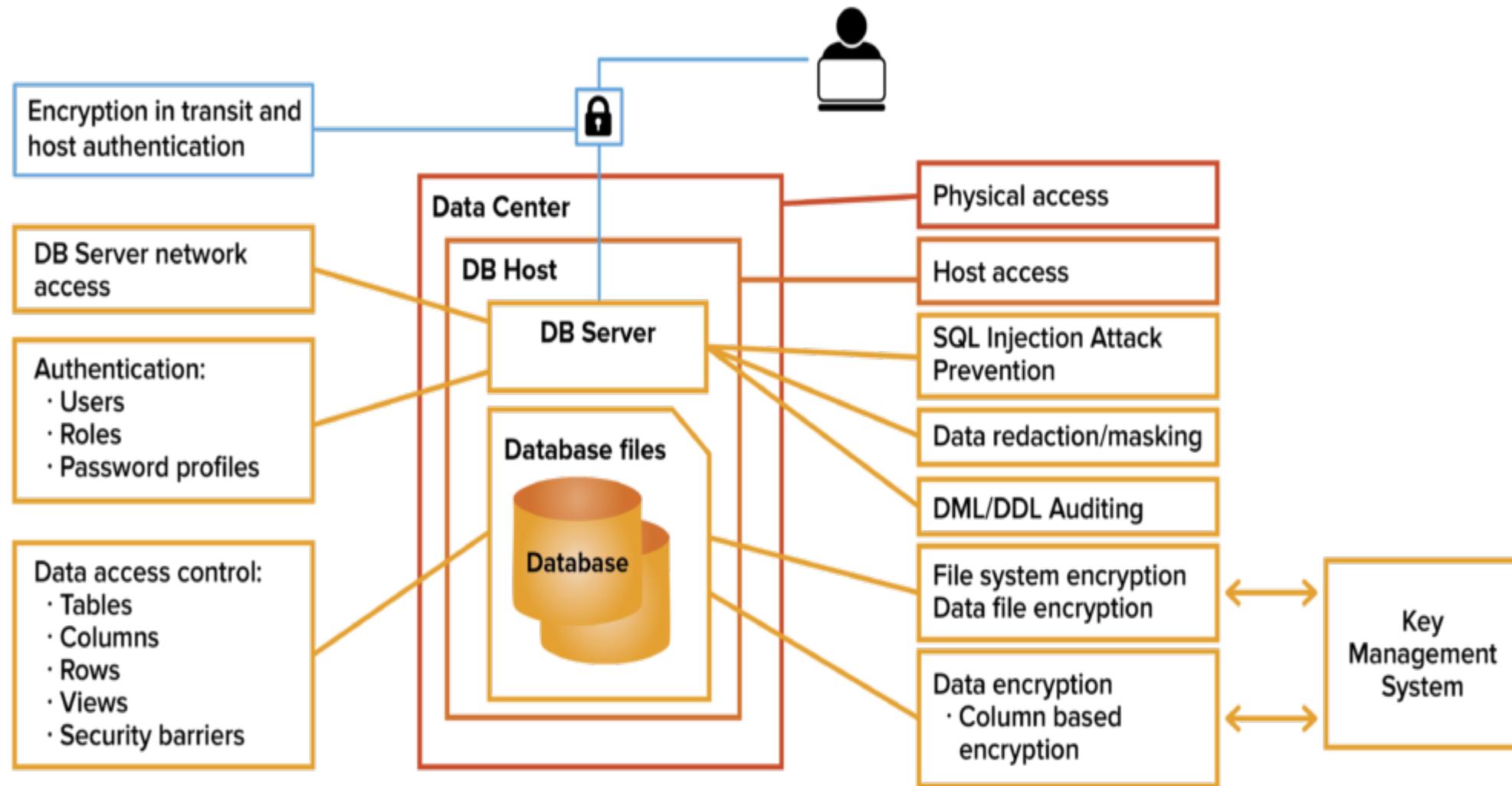
Top Database Security Threats

- Database and Operating System Vulnerabilities
 - SQL Injections
 - Cross Site Scripting
 - Root Kits
 - Weak communication protocols
- Poor Audit Trail
- Front-End Application Vulnerabilities
- Backup
 - Incomplete and failed backups
 - Theft or improper storage of backup storage media or hard drives

Key Control Layers in Database Security

- Applications as well as databases typically contain other control mechanisms which should be considered during risk assessments and audits.
- They include the following:
 - Operating Systems
 - Network Components
 - Applications Systems
 - Physical Security
 - Database Object Security

Multi-layer security architecture (Postgres)



Multi-layer security architecture (Postgres)

- **SQL injection attack prevention:** Make sure your DBMS has this feature and use it to block corruption or co-opting of a database, including unauthorized relations, utility commands, SQL tautology, and unbounded DML.
- **Database authentication:** Users should be authenticated using passwords, LDAP, Kerberos, certificates or using operating systems credentials. Database authentication should be tied with overall user management to make sure access credentials get revoked when users move to another assignment, leave the enterprise or any other reason.
- **Database authorization and access control:** Users must be granted permissions to view and work with data in the database. Privileges should only be granted in order for a user to perform a job. Shared or group login credentials should be avoided. A principle of least privilege should be applied.
- **Data encryption (pgCrypto):** If a user gets past file system encryption, they can access a database that's been logged into. Encrypting data at the column level keeps the database information secure.
- **Auditing:** Make sure your DBMS has this feature and use its auditing capabilities so that database administrators, auditors, and operators can track and analyze database activities, like the creation, changing, or deletion of data. This makes it possible to closely monitor what those with permissions are doing. We recommend auditing based on user connections, DDL changes, data changes, and data views.
- **Data redaction:** Make sure your DBMS has this feature which can be used to shield (or mask) certain data elements from certain types of users, like Social Security numbers, for example.

Multi-layer security architecture (Postgres)

- **Physical access (locks on doors, cameras, etc.):** Let's start with actual security of the data center. If a data center is not physically protected, all other data security measures become significantly less valuable.
- **Host access (Operating System controls):** Securing access at the host-level by stringent user access controls like authorization and authentication mechanisms such as Active Directory, Kerberos and LDAP or hardware keys to enable multi-factor authentication. These ensure no users have unfettered access to the database host.
- **File system encryption (through native Linux or others):** Encrypting the file system protects the files on the drive if the drive is stolen or compromised in some way. Other solutions like Vormetric can also be used for this purpose.
- **Database server network access:** In the case of Postgres, the configuration file 'hba.conf' is used to control authentication connections to the database server. It consists of a set of records that specify a connection type, a client IP address range (if relevant for the connection type), a database name, a user name, and the authentication method to be used for connections matching these parameters.
- **Encryption in transit and host authentication (SSL):** All data — including passwords and usernames — should be encrypted on the network using SSL and certificates ensure the user communicates with the intended host machine.

Different Levels of Database Security

- There are different levels of securing your databases, including:
 - Data Level Security: The process of protecting the data itself from getting stolen or tampered with in the servers.
 - System Level Security: Protecting networking servers, hardware and other inbound/outbound communications from acting as a funnel to distribute malicious software.
 - User Level Security: A server is always attacked from a user level, which is why organizations rely on real-time protection software to monitor transactions and also restrict the user from visiting unauthorized websites or downloads from untrusted sources.

How Can I Deploy Database Security?

- There are three layers of database security
 - The database level : Security at the database level occurs within the database itself, where the data live
 - The access level : Access layer security focuses on controlling who is allowed to access certain data or systems containing it
 - The perimeter level : Security at the perimeter level determines who can and cannot get into databases.
- Each level requires unique security solutions.
- Multi-layer security architecture typically contains five components:
 - Secure physical access to the host (perhaps the most important)
 - Limited access to your general corporate network
 - Limited access to the database host
 - Limited access to the database application
 - Limited access to the data contained within

Security Level	Database Security Solutions
Database Level	<ul style="list-style-type: none">• Masking• Tokenization• Encryption
Access Level	<ul style="list-style-type: none">• Access Control Lists• Permissions
Perimeter Level	<ul style="list-style-type: none">• Firewalls• Virtual Private Networks

Database Security Best Practices

- These database security best practices enable organizations to minimize their vulnerabilities while maximizing their database protection.
- Although these approaches can be deployed individually, they work best together to protect against a variety of circumstances impacting database security.
- 1. Physical database security
 - It's critical to not overlook the physical hardware on which the data is stored, maintained, and manipulated.
 - Physical database security includes locking the rooms that databases and their servers are in—whether they are on-premise assets or accessed through the cloud. It also involves having security teams monitor physical access to that equipment.
 - A crucial aspect of this database security best practice is to have backups and disaster recovery measures in place in case of a physical catastrophe.
 - It's also important not to host web servers and applications on the same server as the database the organization wants to secure.

Database Security Best Practices

- 2. Web applications and firewalls

- The use of web applications and firewalls is a database security best practice at the perimeter layer.
- Firewalls prevent intruders from accessing an organization's IT network via the internet; they're a crucial prerequisite for cyber security concerns.
- Web applications that interact with databases can be protected by application access management software.
- This database security measure is similar to access control lists and determines who can access web applications and how they can do so.
- There are also firewalls for individual web applications that deliver the same benefits as traditional firewalls.
- **Application/web server security:** Any application or web server that interacts with the database can be a channel for attack and should be subject to ongoing security testing and best practice management.

Database Security Best Practices

- 3. Database encryption
 - Encryption is one of the most effective database security practices because it's implemented where the data are in the database.
 - However, organizations can encrypt data in motion as well as at rest, so that it's protected as it flows between IT systems in an organization.
 - Encrypted data is transfigured so it appears as gibberish unless it's decrypted with the proper keys.
 - Therefore, even if someone is able to access encrypted data, it will be meaningless to them.
 - Database encryption is also key for maintaining data privacy, and can be effective for [IoT security](#).

Database Security Best Practices

- 4. Access Control (Manage passwords and permissions)
 - Managing passwords and permissions is critical for maintaining database security.
 - This task is usually overseen by dedicated security employees or IT teams. In some instances, this database security best practice involves access control lists.
 - Organizations can take many different steps to manage passwords, such as using dual or multiple factor authentication measures, or giving users a finite amount of time to input credentials.
 - However, this practice requires constant updating of access and permissions lists. It can be time consuming, but the results are worth it.
 - The practical minimum number of users should have access to the database, and their permissions should be restricted to the minimum levels necessary for them to do their jobs.
 - Likewise, network access should be limited to the minimum level of permissions necessary.

Database Security Best Practices

- 5. Isolate sensitive databases
 - It's very difficult to penetrate database security if sensitive databases are isolated.
 - Depending on how the isolation techniques are deployed, unauthorized users might not even know sensitive databases exist.
 - Software defined perimeters are useful means of isolating sensitive databases so that they don't appear to be on a particular user's network.
 - This approach makes it difficult to take over databases with [lateral movement](#) attacks; it's also effective against [zero-day attacks](#). Isolation strategies are one of the best ways to solidify database security at the access level.
 - Competitive isolation solutions combine this approach with database layer security like public keys and encryption.
 - **Backup security:** All backups, copies, or images of the database must be subject to the same (or equally stringent) security controls as the database itself.

Database Security Best Practices

- 6. Change management
 - Change management requires outlining—ideally in advance—what procedures have to take place to safeguard databases during change.
 - Examples of changes include mergers, acquisitions, or simply different users gaining access to various IT resources.
 - It's necessary to document what changes will take place for secure access of databases and their applications.
 - It's also important to identify all the applications and IT systems that'll use that database, in addition to their data flows.
 - **Database software security:** Always use the latest version of your database management software, and apply all patches as soon as they are issued.

Database Security Best Practices

- 7. Database auditing
 - Database auditing usually requires regularly [reading the log files](#) for databases and their applications.
 - This information reveals who accessed which repository or app, when they accessed it, and what they did there.
 - If there is unauthorized access to data, timely audits can help reduce the overall impact of breaches by alerting database administrators.
 - The quicker organizations can react to data breaches, the more time they have to notify any customers involved and limit the damage done.
 - Database auditing provides centralized oversight for database security as a final step for protection.
 - Always be aware of who is accessing the database and when and how the data is being used
 - All user devices connecting to the network housing the database should be physically secure (in the hands of the right user only) and subject to security controls at all times.

Common Security Features in Databases

- Basic Security Mechanism in databases includes
 - Identification and Authentication requirements
 - System Privilege and Object Access Control
 - Audit Trail Mechanism
 - Data Encryption
 - Network Security
 - Auditing/Fine Grain Auditing.

1. Identification and Authentication

- Users can access databases through a variety of means including remotely, wireless access, scanners, through the internal network, etc. There are associated risks with each access means.
- Each user may be identified and authenticated by either the operating system or the database system.
- For example:
 - The Administrator can specify an Oracle password for each Oracle user when the account is created, or
 - In UNIX a user account e.g. DavidO can be Oracle user “OPSS\$DavidO” and connect without a password
 - This feature is disabled by default for remote users
- MSSQL has mixed authentication. Users may log in with either an operating system ID, or a separate database user account.

2. Reviewing Users and Passwords

- Obtain a list of all Database User Accounts
 - Describe dba_users
 - Select * from sys.dba_users
- Identify the purpose and use of each user account
 - Identify generic accounts
 - Identify shared account
 - Service Accounts
 - Guest or anonymous logins
- Review Password Policies defined in both the database and operating systems
- Check for the use of common default passwords or blank passwords.

3. Reviewing DB Users and Passwords

- Review user profiles as they contain the following important password control mechanism such as these settings:
 - Failed login attempts
 - Password Expiration
 - Account lockout duration
 - Minimum and Maximum password length
 - Password reuse
 - Password grace period before the account expires
- Earlier versions of SQL prior to SQL 2000 lack controls such as password complexity, expiration, lockout, and password history.

4. Application Systems Connections

- Popular Application Systems such as JDE, PeopleSoft, SAP/R3 and other applications also connect directly to the database
- Home grown and legacy applications also connect to the database and in many cases these are done with hard coded passwords
- Key Security and Control Issues include the following
 - Access to data outside the Application System
 - Application System Access and Security
 - Application Systems Internal Controls

5. Application Systems Connections

- Many popular applications as well as home grown applications are configured to use their passwords to login to the database.
- Such application may supply its UserID and password and not the end user's. In this case
 - Neither the Operating System nor Database is aware of the end user's identity
 - Neither the OS or Database (e.g. Oracle) can enforce access control decisions or monitoring based on end user identity
 - Such passwords are sometimes hard coded in the application or script and rarely changed
 - The password may be stored in a user-accessible or unencrypted file

6. Bypassing Application Controls

- This can occur when a user has an ID with direct access to the database and the underlying tables.
 - They can update compensation tables or other sensitive data
 - SOL*PLUS would allow the user update access outside the application
 - Risk Management professionals should evaluate database and application security to determine if level of protection is sufficient.
- Users should not be directly defined in the database if they login with a front end application.

7. Reviewing Access Control

- Roles, Responsibilities and Privileges
 - Rules defining what users can do
- System Privileges
 - Allows a user to perform a particular database command or operation
- Object Privileges
 - Allows a user to access an object in a particular manner
- Statement Privileges
 - selective auditing of related groups of statements regarding a particular type of database structure or schema object
- Review the operating system permissions of all key database files
- Privileges are provided directly to users or through roles.

8. Logging and Monitoring

- Define what actions and abuses that need to be checked such as the following:
 - Successful database access
 - Changes to database structure
 - Failed logon attempts
 - Attempts to access the database with non-existent user names
 - Attempts to access the database at unusual hours.
 - Checks for users sharing database accounts
 - Multiple access attempts using different usernames from the same terminal
- Database auditing is viewed as being complex and slow but this is generally not true.

9. Backup and Recovery

- Ensure that an appropriate backup and recovery strategy exists
- Oracle has a number of backup and recovery mechanism including the following
 - Redo and archive logs
 - Import and export utilities
 - Partial and Full database backup
- SQL Server has a number of back-up features including
 - Database backup
 - Transaction log
 - Differential backup
 - File / File group backups

10. Vulnerability Analysis

- Conduct periodic Vulnerability Assessments
- Vulnerability Assessment Tests probe for known vulnerabilities
 - Identify vulnerable software versions
 - Identify vulnerable services
 - Probes the database for known weaknesses
 - Test for default and weak passwords
- Common tools that are used for the following include
 - Nessus, Retina, Saint (Generic Assessments)
 - App Detective, NGSSquirrel, (Specific to Databases)

11. Privacy Issues

- Data privacy concerns those kinds of information that relate to individuals and external organizations that are part of the company's database.
 - Who owns this information—the company that has the database or the individuals and organizations to whom the information relates?
 - Who can access this information?
 - Can this information be sold to others?
 - What are the regulations?
- Legislation about privacy and confidentiality of information varies from region to region
- Privacy concerns escalate with the widespread use of the Internet.
- Although formal regulations may not be adequate, organizations are ethically obliged to prevent misuse of the information they collect about individuals and third-party institutions

Common Database Security Issues

- Users with excessive privileges
 - Users with administrative privileges
 - Users whose privileges are higher than their role requires
 - Users who have moved or changed job roles
- Lack of logging and No Auditing of Privileged user s
- Failure to Segregate Duties appropriately
- Inadequate review of Audit logs
- Generic, shared, and terminated users with access to production systems
- Guest user in production databases
- Improperly configured systems and weak patch management
- Incomplete backup and failure to encrypt backup data