Week 10 - Wednesday

COMP 4290

Last time

- Finished network vulnerabilities
- Network security controls
- Firewalls

Questions?

Project 3

Aidan Kent Presents

Intrusion Detection

Intrusion detection

- Firewalls and authentication mechanisms are supposed to prevent malicious attacks
- Not all attacks can be prevented
  - But it's still useful to know when they are happening
- An **intrusion detection system** (**IDS**) is hardware or software that monitors activity to look for suspicious patterns
- A **network-based** IDS is stand-alone hardware that monitors a whole network
- A **host-based** IDS runs on a host to protect that host

Types of IDSs

- **Signature-based** IDSs do pattern matching, looking for patterns of known malicious behavior
  - Only works for known types of attacks
- **Heuristic** (or **anomaly based**) IDSs build up a model of acceptable behavior
  - If something doesn't fit the model, an alarm is raised
  - An example is a particular user who has a characteristic way of typing that suddenly changes
- **State-based** IDSs try to see when the system is in an unsafe state
- **Model-based** IDSs try to model unacceptable activity and react when activity looks like the model
- **Misuse intrusion detection** is like model-based except that the model is known bad behavior

IDS operation

- Many IDSs are configured in **stealth mode**
  - They cannot send messages on the network they are monitoring
  - Alarms are sent through some alternate means
- Responding to alarms
  - Monitor data
  - Change system settings to protect it
  - Alert a human being
- Because they are often statistical, an IDS can have false positives and false negatives
  - Both are problematic

Database Background

What is a database?

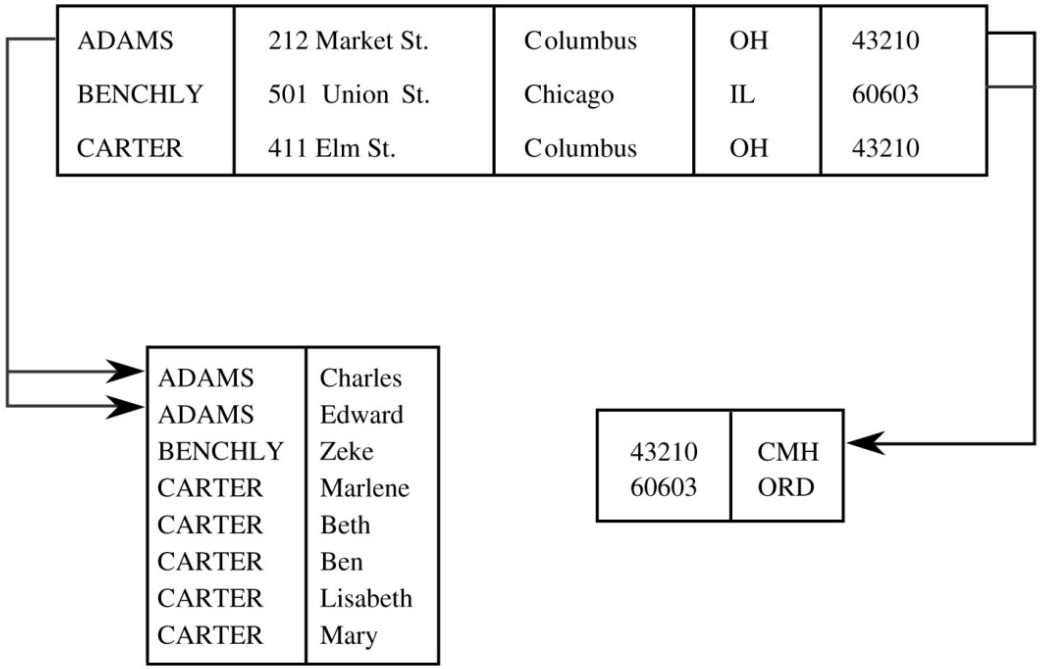
- A **database** is a collection of data and a set of rules to organize the data by relationships
- A **database administrator** makes the rules and controls access
- A **database management system** (**DBMS**) is the program through which the user interacts with the database

Database components

- Most modern databases use the relational database model
  - The fundamental unit of organization is a table
  - An older format for databases was hierarchical, like a tree
- A table consists of **records**
- A record consists **fields** or **elements**, which are each a specific item of data

Schemas

- The tables in a database are usually related to each other in some way
- The logical structure of a database is called a **schema**
- A user may only see part of it, called a **subschema**
- An **attribute** is the name of a column
- A **relation** is a set of columns



Queries

- A **query** is the name of a command given to a database by a user
- Queries can:
  - Retrieve
  - Modify
  - Add
  - Delete
- Most databases allow commands to be issued through a variant of SQL

Table example

- Table CLIENTS

Name	First	Address	City	State	Zip	Airport
ADAMS	Charles	212 Market St.	Columbus	OH	43210	CMH
ADAMS	Edward	212 Market St.	Columbus	OH	43210	CMH
BENCHLY	Zeke	501 Union St.	Chicago	IL	60603	ORD
CARTER	Marlene	411 Elm St.	Columbus	OH	43210	CMH
CARTER	Beth	411 Elm St.	Columbus	OH	43210	CMH
CARTER	Ben	411 Elm St.	Columbus	OH	43210	CMH
CARTER	Lisabeth	411 Elm St.	Columbus	OH	43210	CMH
CARTER	Mary	411 Elm St.	Columbus	OH	43210	CMH

Query example

- Query:

```
SELECT * FROM CLIENTS WHERE FIRST="BEN" OR CITY="CHICAGO"
```

Name	First	Address	City	State	Zip	Airport
BENCHLY	Zeke	501 Union St.	Chicago	IL	60603	ORD
CARTER	Ben	411 Elm St.	Columbus	OH	43210	CMH

Database advantages

- Databases have many advantages:
  - **Shared access** for many users
  - **Minimal redundancy** so that space is used efficiently
  - **Data integrity** with rules that protect relationships
  - **Controlled access** with authorized users
- Databases have also been heavily optimized for speed
- Users don't need to know anything about the actual physical layout of the database on disk

Database Security Requirements

Database security requirements

- Because they are a central part of modern business, several aspects of database security are crucial:
  - Physical database integrity
  - Logical database integrity
  - Element integrity
  - Access control
  - User authentication
  - Availability

Database integrity

- Physical database integrity
  - What happens in a power failure?
  - Disk drives fail all the time
- Regular backups are necessary
- Google and Amazon offer distributed database services
- Transaction logs should be kept

Element integrity

- The integrity of an individual element is its correctness or accuracy
- **Field checks** make sure that data values fall within appropriate ranges or have the right types
  - Usually these checks are done as data is being entered
- **Access control** is key
  - Partly to protect data from malicious users
  - Partly to avoid situations where two users update information at the same time, leading to inconsistency
- A **change log** keeps track of all changes, allowing for an undo of mistaken updates

Auditability

- Like with OS logs, we want to be able to keep track of who has accessed the database
- Similarly, the log can become very long
- Should we record when a user has indirectly accessed a value through a SELECT statement?
  - This is called the **pass-through problem**

Access control and authentication

- Managing access control for a database is very difficult
  - Many systems allow for very fine-grained control
  - But some human has to assign all the privileges
- Worse, giving a person access to some data but not others might not be enough
- Some queries can leak information about hidden data
  - Getting this data is called **inference**
- Most DBMSs are separate from the OS
  - Since there is no trusted path, the DBMS must do its own authentication

Availability

- Availability is another challenge for a DBMS
- Since these systems make the world work, everyone notices when they go down
- If a field is not available to user A while user B is editing it, user A may have to wait an unacceptable amount of time
- To avoid inference, data that should be publicly known might be unreasonably hidden
- CIA all come together in DBMSs

Database Reliability and Integrity

Reliability and integrity

- **Reliability** is a measure of how long a software system can run without failing
  - Reliability is often quoted in terms of uptime percentage
  - Or mean time between failures
- Database reliability and integrity has three aspects:
  - Database integrity
    - Is the database as a whole protected from disk failure or corruption
  - Element integrity
    - Are only authorized users allowed to change elements
  - Element accuracy
    - Are the values in the elements correct

Two-phase update

- A key problem for database integrity is what happens if the system fails in the middle of an update
  - Then the database is inconsistent
- A two-phase update is a common solution
  - During the **intent** phase, the DBMS computes the results needed for the update, but does not change the database
  - During the **commit** phase, it changes all of the fields to the values computed in the intent phase
  - If the intent phase fails, the DBMS can start over from the beginning
  - If the commit phase fails, the DBMS can try to write all the data from the intent phase again

Two-phase example part 1

- Avon and Stringer use a database to organize their heroin distribution cartel
  - Assume that they have 1483 doses in their warehouse
- If a request for a re-up for 100 WMD's comes from the Pit's crew chief, the following steps happen:

1. They check the warehouse to see if they have enough, otherwise the request is postponed
2. If they have enough, they remove 100 from the warehouse (1483 – 100 = 1383)
3. They add 100 doses to the crew chief's sheet of product
   - If the crew chief is more than 1000 doses behind on payment, he is shot
4. If the warehouse's quantity on hand (1383) is below 500, an order is made to the Greek importers for another heroin shipment
5. The re-up delivery to the Pit is made

Problems

- If the steps are not correctly carried out in order, bad things happen
- Imagine a failure in the process
  - If 100 is removed from the warehouse inventory field and the process fails, the accounting for the warehouse is off
  - If repeated failures cause 100 doses to be added to the crew chief's sheet several times without a delivery, he might get shot
- In the two-phase system, we use **shadow values** to keep track of changes
- When the process has finished, we write the list of shadow values

Two-phase example part 2

- To make the protocol robust to failure, we use the following intent phase:

1. Check the COMMIT-FLAG, if true, return failure or wait until false
2. Check the warehouse to see if they have enough, otherwise the request is postponed
3. Compute TDOSES = ONHAND - REQUESTED
4. Compute TSHEET = SHEET + REQUESTED
5. If TSHEET > 1000, set TKILLCHIEF = true, otherwise set TKILLCHIEF = false
6. If TDOSES < 500, set TREORDER = true, otherwise set TREORDER = false

Two-phase example part 3

- This is the corresponding commit phase:

1. Set the COMMIT-FLAG in the database
2. Set ONHAND = TDOSES
3. Set SHEET = TSHEET
4. Set KILLCHIEF = TKILLCHIEF
5. Set REORDER = TREORDER
6. Unset COMMIT-FLAG

- When finished, make the delivery

Redundancy

- DBMSs often keep information for error correction and detection:
  - Parity bits
  - Hamming codes
  - Cyclic redundancy checks
- Shadow fields (like the ones used in two-phase updates) can be used to replicate individual fields or entire records
- Because events are also logged, it should be possible to reconstruct the database from a backup based on the log data

Concurrency

- Most database systems allow more than one user or process to access it at the same time
- Updates must be controlled to avoid **race conditions**
  - Race conditions are sequences of commands that result in different states depending on timing
  - If there is one ticket left to a Bad Bunny concert, it should be impossible for two people to buy it
- Commands that both query (is there a ticket remaining) and update (buy the ticket) should be executed atomically
- Reading data also needs to be protected
  - If a user is writing data, it should be locked so that it can't be read

Constraints

- A **monitor** is the part of the DBMS responsible for structural integrity
- **Range comparisons** check newly entered numerical data for sanity
- **Filters** or **patterns** can be arbitrarily complex to make sure that a zip code or a VIN is correctly formatted
- The job of a DBA is to set these up, as well as the more complex state and transition constraints

State and transition constraints

- A **state constraint** is a characteristic that should be invariant over the database
  - Only one person is labeled president
  - Only one table has a given name
  - If such a constraint is violated, something has gone wrong in the database
- A **transition constraint** must be met before certain changes can be made to the database
  - A vacant position has to be listed before a new employee can be added
  - A student record must exist before that student's ID can be added to a class

Upcoming

Next time…

- Database disclosure
- Database inference
- Big data and data mining
- Nfaly Toure presents

Reminders

- Read Sections 7.3 through 7.5
- Work on Project 3
  - User names and passwords need to be turned in next Friday in class