CSCI262 : System Security

Database Security

Schedule

- Databased Management System (DBMS)
- Relational Database Model
- Structured Query Language (SQL)
- SQL Injection Attacks
- Statistical Databases

Information versus data

- What is the difference between information and data?
- Informally not necessarily anything …
- … but we are interested in distinguishing between:
  - The raw data that we might store, such as the ages of 50,000 students in our database.
  - … and the information, such as the average student age that a database allows us to extract.
- Typically we think of data as the collected input, while information is the output resulting from processes applied to data.
  - Information is knowledge.

Database Management System (DBMS)

- A database is a structured collection of data stored for use by one or more applications
  - contains also the relationships between data items and groups of data items
- A database management system (DBMS) allows related data to be centrally stored and controlled.
  - Controlled includes standardising how data is entered, updated, retrieved and deleted.
  - A DBMS integrates the data and the management of the data.
- You might remember the concept of entropy, it can be used to measure the amount of information.
  - (Raw) data is almost certainly an inefficient representation of information.

Databases versus file(/operating) systems

- Database systems store data and/or information, and provide access to information, and sometimes data.
- File systems store and provide access to data.
  - Sure you can access information too, but that is primarily through application interfaces and in a file system the related information may be spread across multiple files or duplicated.
- In a file system, we directly interact with the files, but a DBMS acts as a controlled interface.

- Concurrency issues need independent control in an operating system, in the sense that two copies of data wouldn’t normally be synchronised automatically.
    - A DBMS can deal with concurrency internally, including handling multiple users.
- File formats within file/operating systems are typically tied to applications, whereas the internal structure of a database can be significantly changed without needing to change the applications that use the database.

Types of information

- There are various categories of information that could be extracted from a database.
- **Exact information**:
  - Alice is 25 years old.
- **Bounds**:
  - Alice is less than 70 years old.
- **Negative results**:
  - Alice is not a male.
  - There is nobody more than 95 years old.
- **Existence**:
  - There exists a person with red hair.

- **Associations or relations among pieces of information**:
  - Alice is younger than Bob.
  - Alice has spent more time as a student than Bob.
- We might want to control access to some of those.
- And it isn’t simply a matter of not letting people get access to certain entries in the database!
- Later we will see the idea of inference.
  - The application may insist we cannot get the address of a specific person.
    - But we can find out there is a 25 year old female student living in Crypto Lane.
    - A separate query may reveal that Alice is the only 25 year old female.
    - We can infer that Alice lives in Crypto Lane.
    - Think sets and Venn diagrams, if that helps. ☺

Security properties for databases

- The primary database asset is the stored data, and the information that can be extracted from it.
- **C**onfidentiality…
  - Assets cannot be accessed by persons not authorised to access that asset.
    - "Accessed" could be replaced by a reference to a specific disallowed action.
- **I**ntegrity…
  - Assets cannot be modified by persons not authorised to modify that asset.
- **A**vailability…
  - Assets should be accessible to those allowed access.
- It's still about access control and authentication.

The relational database model

- A relational database is a collection of relational tables, which are referred to as relations or entities.

Relation name

N-tuple (row)

Attribute name

Relational schema (header)

EMPLOYEE

E#	NAME	DOB	SALARY	CITY	STREET	BLDG#
10	John	12-DEC-67	60K	Sydney	Station	12
20	Mary	01-MAR-56	100K	Perth	Victoria	5
30	James	08-SEP-68	90K	W'Gong	Barham	13
40	Kate	17-OCT-80	50K	W'gong	Barham	13
…	…	…	…	…	…	…

Column

Attribute value

Relation (relational table)

- A **relation** (or table or sometimes file) can be written as

$$\mathbf{R \subseteq D_1 \times D_2 \times \ldots \times D_n}$$

where $D_1, D_2, \ldots, D_n$ are the domains of the attributes $A_1, A_2, \ldots, A_n$.

- The attributes are the columns so the relation corresponds to the product space over the spaces of each attribute, and are sometimes called fields.
- The elements of the relation are called n-tuples and are simply rows in the table.
  - They are sometimes called records.
- An n-tuple is a sequence $<v_1, v_2, \ldots, v_n>$ where $v_i \in D_i$ for each $i = 1, 2, \ldots, n$.

Structured Query Language (SQL)

- SQL has a long history.
  - It was developed by IBM in the mid 1970's.
- SQL is a (sort-of) standard language designed to carry out interactions with relational databases.
  - This includes inserts, updates, retrievals and deletions.
  - The SQL commands fall into two categories: Data Manipulation Language (DML) or Data Definition Language (DDL).
- There seem to be several versions of the standard and various implementation differences but at the basic level SQL is quite similar.
- On the next couple of slides there are examples of different SQL statements…

An INSERT statement

INSERT INTO EMPLOYEE VALUES
(70,'James','12-08-78', 90000,'Sydney', Pitt,45);

Two UPDATE statements

UPDATE EMPLOYEE
SET DOB ='13-08-78'
WHERE E# = 70;

UPDATE EMPLOYEE
SET SALARY = SALARY + 0.1*SALARY;

EMPLOYEE

E#	NAME	DOB	SALARY	CITY	STREET	BLDG#
10	John	12-DEC-67	66K	Sydney	Station	12
20	Mary	01-MAR-56	110K	Perth	Victoria	5
30	James	08-SEP-68	99K	W'Gong	Barham	13
40	Kate	17-OCT-80	55K	W'gong	Barham	13
70	James	13-08-78	99K	Sydney	Pitt	45

A DELETE statement

```
DELETE FROM EMPLOYEE
WHERE DOB < '01-JAN-70';
```

Some SELECT statements

```
SELECT *
FROM EMPLOYEE;

SELECT E#, NAME, DOB
FROM EMPLOYEE;

SELECT E#, NAME, SALARY
FROM EMPLOYEE
WHERE (DOB <= '01-JAN-70') AND (CITY = 'Sydney');
```

SELECT retrieves data/information from a relation.

* is a wildcard…

Relational Views

- A relational view is a named derived virtual relation defined in terms of the other named relations and/or other relational views.
- One use of views is for access control.
  - We allow access to a derived view rather a base relation.

Two CREATE VIEW statements

```
CREATE VIEW SYD_EMP AS(
        SELECT *
        FROM EMPLOYEE
        WHERE CITY = 'Sydney');

CREATE VIEW OLD_EMP AS(
        SELECT E#, NAME
        FROM EMPLOYEE
        WHERE DOB < '01-01-50');
```

Changing data through a view...

```
SELECT E#, NAME
FROM SYD_EMP
WHERE STREET = 'Broadway';

DELETE FROM SYD_EMP
WHERE NAME = 'Patrick';

UPDATE SYD_EMP
SET NAME = 'Maggie'
WHERE E# = 77;

INSERT INTO SYD_EMP VALUES(
(13,'Mike','29-FEB-72',60000,'Hobart',Victoria,5);
```

Database keys

- These are not cryptographic keys. ☹
- There are a few terms we need to describe.
- *A* **minimal key for a relation** is an attribute that identifies all tuples of the relation in a unique way, or a set of attributes that does the same but for which no proper subset is a minimal key.

- For EMPLOYEE(E#,NAME,DOB,SALARY,CITY,STREET,BLDG#), (E#) is a minimal key.
- (E#,NAME) is NOT a minimal key because (E#) is a proper subset and a minimal key.

EMPLOYEE

E#	NAME	DOB	SALARY	CITY	STREET	BLDG#
10	John	12-DEC-67	60K	Sydney	Station	12
20	Mary	01-MAR-56	100K	Perth	Victoria	5
30	James	08-SEP-68	90K	W'Gong	Barham	13
40	Kate	17-OCT-80	50K	W'gong	Barham	13
…	…	…	…	…	…	…

More key key terminology ☺

- The **Primary key** is the arbitrarily selected minimal key.
- A **Candidate key** is a minimal key which is not the primary key.
- A **Foreign key** is an attribute, or set of which together, the values of which are the same as the values of a primary or candidate key in another relation.
    - This defines a relationship between the two relations.

An example…

- Did and Eid are, respectively, the primary keys for the Department and Employee relations.
- Did is a foreign key in the Employee relation.

Department Table

Did	Dname	Dacctno
4	human resources	528221
8	education	202035
9	accounts	709257
13	public relations	755827
15	services	223945

$\underbrace{\text{Did}}_{\text{Primary key}}$

Employee Table

Ename	Did	Salarycode	Eid	Ephone
Robin	15	23	2345	6127092485
Neil	13	12	5088	6127092246
Jasmine	4	26	7712	6127099348
Cody	15	22	9664	6127093148
Holly	8	23	3054	6127092729
Robin	8	24	2976	6127091945
Smith	9	21	4490	6127099380

$\underbrace{\text{Did}}_{\text{Foreign key}}$ $\underbrace{\text{Eid}}_{\text{Primary key}}$

(a) Two tables in a relational database

Figure 5.4(a) in [SB18]

Dname	Ename	Eid	Ephone
human resources	Jasmine	7712	6127099348
education	Holly	3054	6127092729
education	Robin	2976	6127091945
accounts	Smith	4490	6127099380
public relations	Neil	5088	6127092246
services	Robin	2345	6127092485
services	Cody	9664	6127093148

(b) A view derived from the database

Figure 5.4(b) provides a view that includes the employee name, ID, and phone number from the Employee table and the corresponding department name from the Department table.

SQL Injection (SQLi)

- Part of #1: Injection on the OWASP Top Ten.
  - Open Web Application Security Project.
- There are different types of SQL injection attacks, and the classification varies a fair bit.
  - The classifications are on different characteristics, how/where …
  - There are distinctions between integer and string based injections, and between where the information goes, and …
- However, the underlying principles are the same:
  - Something is entered somewhere, by the client, in the context of SQL, causing something not nice to happen, such as meaning some confidential information to be able to be extracted.
  - Typically via some application server interactions with a database.

SQL Injection (SQLi)

- First, the attacker provides input into an application that passes the data over to the database.
  - Giving the attacker some form of authentication and granting a level of privilege on the database.
- Then, the application passes the user input and a command associated with that input field to the database, which interprets the user data according to the application command.
- Finally, the database executes the application command with the interpreted data and produces some result such as database corruption, record deletion, or operating system compromise.

Typical Example of SQLi attack

1. Hacker finds a vulnerability injects an SQL
2. command to a database
3. The Web server receives the malicious code and sends it to the Web application server.
4. The Web application server receives the malicious code from the Web server and sends it to the database server.
5. The database server executes the malicious code on the database. The database returns data from credit cards table.
6. The Web application server dynamically generates a page with data including credit card details from the database.
7. The Web server sends the credit card details to the hacker.

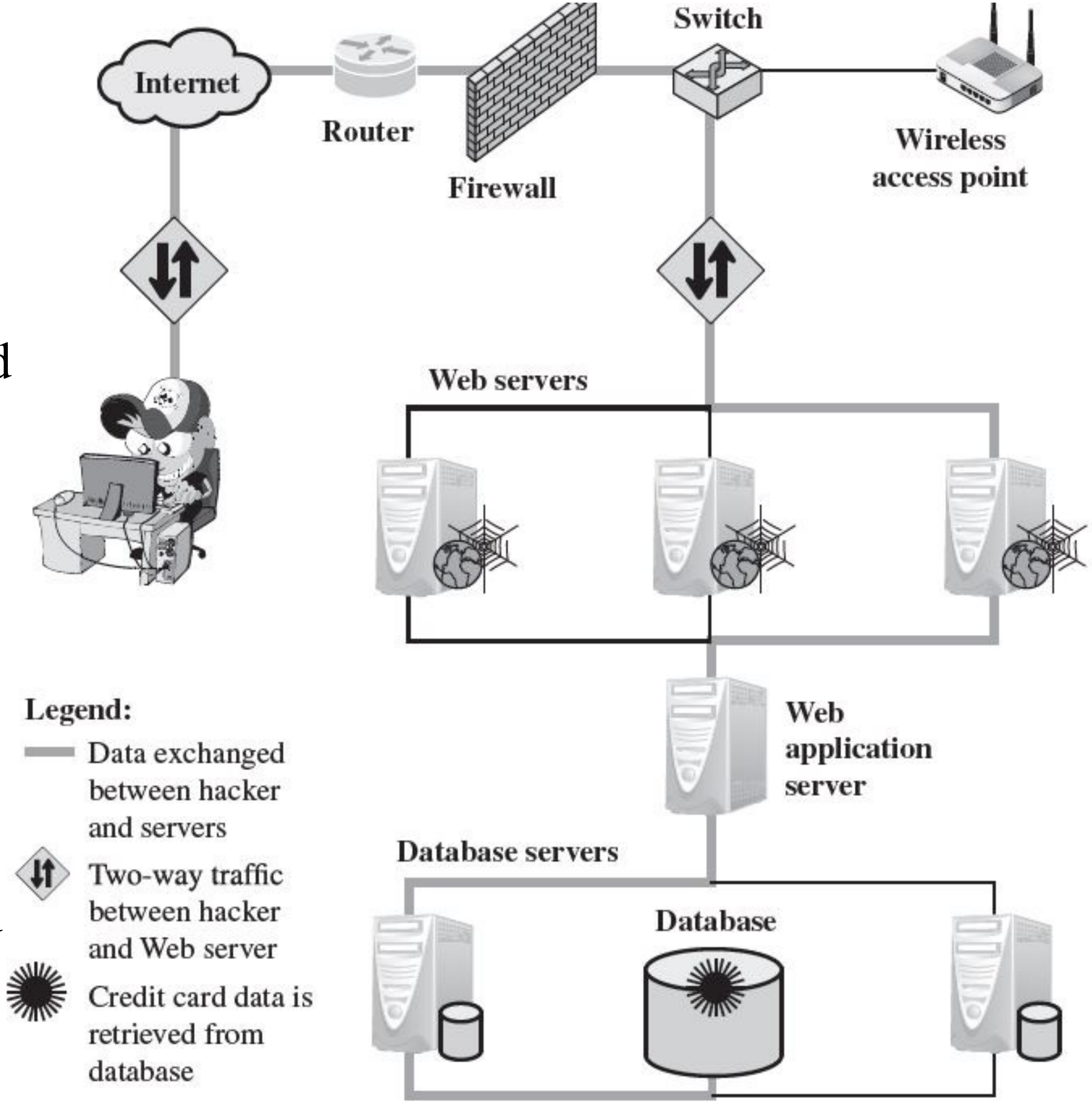


Figure 5.5 Typical SQL Injection Attack

Some simple SQLi attacks

- The most common type of SQL Injection attacks are manipulation attacks where existing SQL statements are modified:
  - Adding elements to the WHERE clause, or
  - Extending the SQL statement with set operators like UNION, INTERSECT, or MINUS.
- Typically they are cutting short or extending statements.

Username	Bob
Password	***********

At the front

Behind the scenes

```
SELECT * FROM users
WHERE username = '<The entered username>'
AND password = '<The entered password>'
```

Username	Bob' OR 1=1--
Password	

```
SELECT * FROM users
WHERE username = 'Bob' OR 1=1--'
AND password = ' '
```

- The -- means the rest is treated as a comment and ignored.

Main categories of attack

- The textbook [SB18] describes 3 main categories of attack:
- **Inband:** Using the same communication channel for attacking and retrieving data.
- **Out-of-band:** attacks typically involve the data being collected through different channels.
- **Inferential:** reconstruct database structure

Statistical database

- A statistical database is primarily defined on the basis of the data or information that it provides… that is based on the nature of the query results.
- Rather than getting exact entries, or precise lists of entries, we get statistical information, such as an average or sum.
  - Direct access isn’t allowed.
- A single physical database can have a statistical database interface and the typical type of interface we saw earlier with the standard select queries.
  - A pure statistical database stores only statistical data.

Why use one? → What are we protecting?

- Data inherently sensitive:
  - Information about the locations of WMD (Weapons of Mass Destruction).
- Data from a sensitive source:
  - Information that compromises the identity of the informer.
- Data declared as sensitive:
  - Classified military data.
- A sensitive component of a tuple (row):
  - The values of attribute salary.
- Data sensitive in relation to earlier disclosed information:
  - X invested money in company Y.
  - The owner of Y and X belong to the same golf club and frequently play together.

Types of disclosure

- **Exact data:**
  - Analysis of the results of correlated complex queries provides the exact value of data item.
- **Bounds:**
  - Analysis of the results of correlated complex queries provides the lower and upper bounds for a value of item.
- **Negative results:**
  - Analysis of the results of correlated complex queries provides information that v is NOT a value of item x.
- **Existence:**
  - Analysis of the results of correlated complex queries provides information that a data item exists.

Types of information

- There are various categories of information that could be extracted a database.
  - We mentioned this earlier but this is specifically relevant to inference so we will repeat these categories and examples.
- **Exact information**:
  - Alice is 25 years old.
- **Bounds**:
  - Alice is less than 70 years old.
- **Negative results**:
  - Alice is not a male.
  - There is nobody more than 95 years old.
- **Existence**:
  - There exists a person with red hair.

- **Associations or relations among pieces of information**:
  - Alice is younger than Bob.
  - Alice has spent more time as a student than Bob.
- We might want to control access to some of those.
- And it isn’t simply a matter of not letting people get access to certain entries in the database!
- The application may insist we cannot get the address of a specific person.
  - But we can find out there is a 25 year old female student living in Crypto Lane.
  - A separate query may reveal that Alice is the only 25 year old female.
  - We can infer that Alice lives in Crypto Lane.
  - Think sets and Venn diagrams, if that helps. ☺

Attacks…an example…

- Before describing some of the formalisms for statistical databases we will consider an example of a non-statistical inference attack.
- We will try and construct queries to precisely match one data item.

```
SELECT NAME
FROM EMPLOYEE
WHERE SALARY BETWEEN 50000 AND 100000;
```

- If 3 names are found then we could try:

```
SELECT NAME
FROM EMPLOYEE
WHERE SALARY BETWEEN 75000 AND 100000;
```

- If no names are found then we try:

```
SELECT NAME
FROM EMPLOYEE
WHERE SALARY BETWEEN 62500 AND 75000;
```

- And so on …

- The database system might enforce a consistency constraint:
  - *If manager then salary between 100,000 and 200,000.*
- Then a sequence of INSERT statements may disclose the constraint, on the basis of their INSERT's being denied, …
  - And the constraint may itself be confidential.
- A subsequent query …

```
SELECT COUNT(*)
FROM EMPLOYEE
WHERE POSITION = 'MANAGER'
```

- … would provide an indication of how much money the company spends on management salaries.

COUNT?

- COUNT is an example of an aggregate function.
- SUM, AVG, MIN, MAX are the other basic ones.
- These provide aggregate information on a column of a relation.
- Statistical databases provide information by means of statistical (aggregate) queries on an attribute (column) of a relational table.
- There are a lot of other aggregate functions, and it is possible to define new ones for use by the user.

AVG	COLLECT	CORR
CORR_*	COUNT	COVAR_POP
COVAR_SAMP	CUME_DIST	DENSE_RANK
FIRST	GROUP_ID	GROUPING
GROUPING_ID	LAST	MAX
MEDIAN	MIN	PERCENTILE_CONT
PERCENTILE_DISC	PERCENT_RANK	RANK
REGR_ (Linear Regression) Functions		
STATS_BINOMIAL_TEST		
STATS_CROSSTAB		
STATS_F_TEST	STATS_KS_TEST	STATS_MODE
STATS_MW_TEST	STATS_ONE_WAY_ANOVA	
STATS_T_TEST_*		
STATS_WSR_TEST	STDDEV	
STDDEV_POP	STDDEV_SAMP	SUM
SYS_XMLAGG	VAR_POP	VAR_SAMP
VARIANCE	XMLAGG	

Statistical queries …

SELECT *aggregate-function*
FROM *relational-table*
WHERE *query-predicate*
[GROUP BY group-by-*attribute-list*];

- We have already seen the *aggregate functions*.
- The *query-predicates* are predicates that determine the tuples (rows) that are used to compute the *aggregate-function*.
  - These are conditions we need to match.
  - The values in the columns used are not necessarily directly accessible.
- The tuples (rows) matching the query-predicate form the *query-set*.

Some examples…

- Consider that we have a relationship described by the following schema:

```
EMPLOYEE(E#,NAME,GENDER,SUBURB,STREET,SALARY,POSITION,DEPTNAME)
```

- Then one statistical query would be …

```
SELECT AVG(SALARY)
FROM EMPLOYEE
WHERE DEPTNAME = 'SALES' AND GENDER = 'F';
```

- Another would be …

```
SELECT COUNT(*)
FROM EMPLOYEE
WHERE SALARY > 100000
GROUP BY DEPTNAME;
```

Sensitivity levels relationships: Aggregation

- The sensitivity level, or classification, of an aggregate computed over a group of values usually differs from the sensitivity levels of the individual values.
- For example:
  - The sensitivity level of an average salary in a department is lower than sensitivity level of the salaries of individual employees.
- Should the MIN and MAX have lower classifications?
  - After all, they are the salaries of individuals.
  - But they aren’t tied to identities.

… and inference

- Inference means the derivation of sensitive information from non-sensitive (typically aggregated) data.
  - Or strictly speaking the sensitive/non-sensitive are higher/lower.
- For example:
  - An average salary of all employees older than 60 discloses an exact value of salary if exactly one employee older than 60 is employed.
  - Having the information and knowing we have the information are subtly different.
    - How do we know there is only one employee?

Inference attacks

- We can clearly have attacks where aggregates are over small enough samples that information about individual elements of data is leaked.
- Statistical inference tends to involve more complex aggregation.

- Out of channel attacks involves attackers obtaining information from external sources:
    - For example, we might know, independent of the database, who lives in which suburb and who is a member of which department.
- The query …

```
SELECT SUM(SALARY), COUNT(*)
FROM EMPLOYEE
WHERE GROUP BY DEPTNAME, SUBURB;
```

- … may then disclose the salaries of the employees who are the only people employed in a department and living in a particular suburb.

- Another example:
  - We know from external sources that Mary is the only female employee in Security department.
  - So we query:

```
SELECT SUM(SALARY)
FROM EMPLOYEE
WHERE DEPTNAME = 'SECURITY' AND GENDER = 'F'
```

- This discloses Mary's salary, since the summation is over a query set consisting of one row.
- What if the system refused to reveal the results if the summation is performed over a small number of rows, in other words with a small query set?
  - We can still construct an attack, but now from a series of acceptable aggregate queries.

- So, we know from the external sources that Mary is employed in Security department.
- We can perform a series of queries to infer Mary's salary.

```
SELECT COUNT(*)
FROM EMPLOYEE
WHERE DEPTNAME = 'SECURITY';
```

⇒ **5 (total number of employees in Security)**

```
SELECT COUNT(*)
FROM EMPLOYEE
WHERE DEPTNAME = 'SECURITY' AND GENDER = 'M';
```

⇒ **4 (there is only one female employed in Security)**

```
SELECT AVG(SALARY))
FROM EMPLOYEE
WHERE DEPTNAME = 'SECURITY';
```

⇒ **40,000 (total salaries in Security = 200,000)**

```
SELECT AVG(SALARY)
FROM EMPLOYEE
WHERE DEPTNAME = 'SECURITY' AND GENDER = 'M';
```

⇒ **38,000 (total salaries of male employees in Security = 152,000)**

- And now we can now infer her salary…

⇒ **200,000 - 152,000 = 48,000 (a salary of the only female in Security; Mary is a female**

Tracker attacks…

- At some point the sophistication of the combined attacks produces something called a Tracker.
  - This is a sequence of queries that allow us to isolate characteristics of an individual.
  - By analysing the tracker results we can sometimes produce a series of algebraic relationships, containing variables corresponding to particular query results.
    - The characteristics can then be expressed as functions of the queries results.

A tracker example …

- In a yet another example if we know that a triple [`NAME,SUBURB,STREET`] usually uniquely identifies the employees, then we can find a salary of any employee in the following way:

```
SELECT SUM(SALARY)
FROM EMPLOYEE
WHERE (NAME = 'JOHN' AND
     SUBURB = 'LIVERPOOL' AND
     STREET = 'STATION') OR
      POSITION ='MANAGER';
```

$\Rightarrow \Sigma M = 5*10^5$ (assume that John is a manager)

```
SELECT SUM(SALARY)
FROM EMPLOYEE
WHERE (NAME = 'JOHN' AND
     SUBURB = 'LIVERPOOL' AND
     STREET = 'STATION') OR
      NOT (POSITION ='MANAGER');
```

$\Rightarrow \Sigma nonM + John = 7*10^5$

```
SELECT SUM(SALARY)
FROM EMPLOYEE;
```

$\Rightarrow \Sigma M + \Sigma nonM = 11*10^5$

- Continuing, we get a system of 3 linear equations with 3 unknown variables, `ΣM`, `ΣnonM`, and `John`:

```
(1) ΣM = 5*10^5
(2) ΣnonM + John = 7*10^5
(3) ΣM + ΣnonM = 11*10^5
```

- From `(1) + (2) - (3)` we obtain: `John = 1*10^5`

- What if John is not a manager ? Then the system of equations is as follows:

```
(1) ΣM + John = 5*10^5
(2) ΣnonM = 7*10^5
(3) ΣM + ΣnonM = 11*10^5
```

- Why can we not simply say:

```
SELECT SUM(SALARY)
FROM EMPLOYEE
WHERE (NAME = 'JOHN' AND SUBURB = 'LIVERPOOL' AND
    STREET = 'STATION');
```

?

- Because the system refuses to reveal the results when the summation is performed over 1 row !

Tracker attacks

- A query predicate T that allows to track down information about a single tuple (row) is called an individual tracker for that tuple (row).
- For example a triple [NAME,SUBURB,STREET] uniquely identifies John in the previous example.
- A general tracker is a predicate that can be used to find the answer to any inadmissible query.
- For example the predicates:

(NAME = 'JOHN' AND SUBURB = 'LIVERPOOL' AND STREET = 'STATION') OR POSITION ='MANAGER'

(NAME = 'JOHN' AND SUBURB = 'LIVERPOOL' AND STREET = 'STATION') OR NOT (POSITION ='MANAGER')

- … are general trackers.

- Let us lot at a more general tracker mechanism.
- Assume that the predicate ϕ_t is an individual tracker.
- Then the system of general trackers can be constructed in the following way:

$Q_1 = \phi_t$ or ϕ_1 or ϕ_2

$Q_2 = \phi_t$ or ϕ_1

$Q_3 = \phi_t$ or ϕ_3

$Q_4 = \phi_2$ or ϕ_3

- … where the results of queries `q:`ϕ_1`, q':`ϕ_2`, q":`ϕ_3 are mutually disjoint. This means any two pairs have no overlap.
- Using the results of queries v_1,v_2,v_3,v_4, we obtain a system of linear equations:

(1) $f_t + f_1 + f_2 = v_1$

(2) $f_t + f_1 = v_2$

(3) $f_t + f_3 = v_3$

(4) $f_2 + f_3 = v_4$

- The solution is:

(((1)-(2))+(3))-(4): $f_t = (((v_1 - v_2) + v_3) - v_4$

Protecting against inference

- There seem to be three classes of solution.
  - The first is try and design the database in such a way that inference is reduced.
  - The second is to attempt to detect and reject specific queries, or sequences of queries, that would result in an inference channel.
  - The third is to present results in such a way as to make inference difficult, or information theoretically impossible.
- An inference channel is a path by which unauthorised data is obtained, from authorised data.

In database design...

- We can alter the database structure.
  - For example, splitting a table.
- Or we can change the access control structure.
  - Giving a finer tuned model.
- These approaches are likely to reduce availability. ☹

A splitting problem ...

- [SB18] provides an example
- Consider the table Inventory (Figure 5.8)...
- We want to stop Item and Cost being viewed together, so we create some views.

Item	Availability	Cost ($)	Department
Shelf support	in-store/online	7.99	hardware
Lid support	online only	5.49	hardware
Decorative chain	in-store/online	104.99	hardware
Cake pan	online only	12.99	housewares
Shower/tub cleaner	in-store/online	11.99	housewares
Rolling pin	in-store/online	10.99	housewares

(a) Inventory table

```
CREATE VIEW V1 AS(
  SELECT Availability, Cost
  FROM Inventory
  WHERE Department = 'Hardware');
```

```
CREATE VIEW V2 AS(
  SELECT Item, Department
  FROM Inventory
  WHERE Department = 'Hardware');
```

- There is no functional dependence between Item and Cost, so giving both of those views to someone who isn't allowed to see the relationship between Item and Cost, won't have that relationship.
- But, the database itself maintains row order on the views, so someone knowing the structure of Inventory can determine the relation between Item and Cost, and recover the first three rows of the Inventory table.
  - That's inference ☹

```
CREATE view V1 AS
SELECT Availability, Cost
FROM Inventory
WHERE Department = "hardware"
```

```
CREATE view V2 AS
SELECT Item, Department
FROM Inventory
WHERE Department = "hardware"
```

Availability	Cost ($)
in-store/online	7.99
online only	5.49
in-store/online	104.99

Item	Department
Shelf support	hardware
Lid support	hardware
Decorative chain	hardware

(b) Two views

Item	Availability	Cost ($)	Department
Shelf support	in-store/online	7.99	hardware
Lid support	online only	5.49	hardware
Decorative chain	in-store/online	104.99	hardware

(c) Table derived from combining query answers

Protection at the query level...

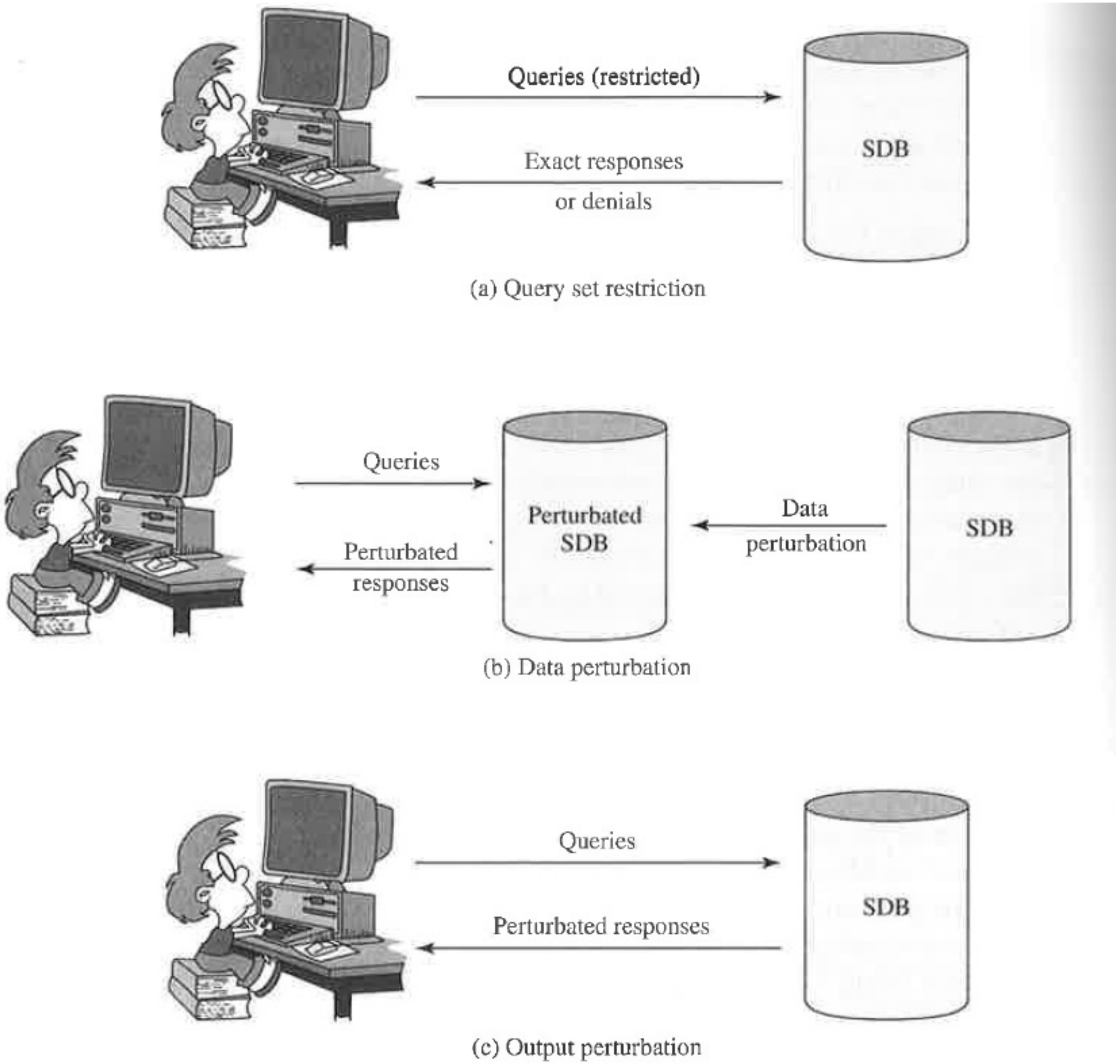


Figure 5.8 Approaches to Statistical Database Security
Source: Based on [ADAM89].

Query set restriction…

- Suppress sensitive information.
  - Do not disclose an answer when a query set is too small, or too large.
- We have seen already that this doesn’t necessarily work.
- We can still sometimes, actually most of the time, construct trackers.

- Consider a database with N rows or records.
- A query q(C) is permitted only if the number of records X(C) matching C satisfies…

- … where k is a fixed integer greater than 1.

$$k \leq |X(C)| \leq N - k$$

Data perturbation

- The data in the database is changed, in such a way that the statistics that are generated are still accurate, but inferential information about characteristics on individual rows is inaccurate.
- How can you calculate the average mark in a class if you don't trust anybody enough to give them your mark?

- How can we perturb data?
- One method is data-swapping.
- Another is to analyse the confidential data and construct a distribution which seems to represent it.
- Then sample from the distribution to construct fake data, for use in a modified database, which is statistically consistent with the original.
  - It will be pretty easy to generate global statistics that are consistent but for subsets of the data we are likely to obtain inaccurate results.

Output perturbation

- This is similar to data perturbation but here we distort the statistical output.
- We want the results to be fairly accurate, but inference to leak little information about the individual data.
- One technique is the Random-sample query method.
  - Here an appropriate subset of the query set that the statistic would be calculated on is determined, and the statistic is calculated on that.
- Alternatively the statistical result on the real query set can itself be changed, likely in a randomized way.

Assessing a protection mechanism

- We need to be concerned about:
  - The possibility of disclosure, partial or complete.
  - The amount of non-confidential information removed, including loss of precision and so on.
  - The costs in setting up the infrastructure, in dealing with queries.
  - Tied to all of these the need for user education regarding the changed level of meaning in the information.

CSCI262 : System Security

Malware (Malicious Software)

Schedule

- What is Malware?
- Classifications
- Propagation Mechanisms
- Payload Actions
- Countermeasures

Malware

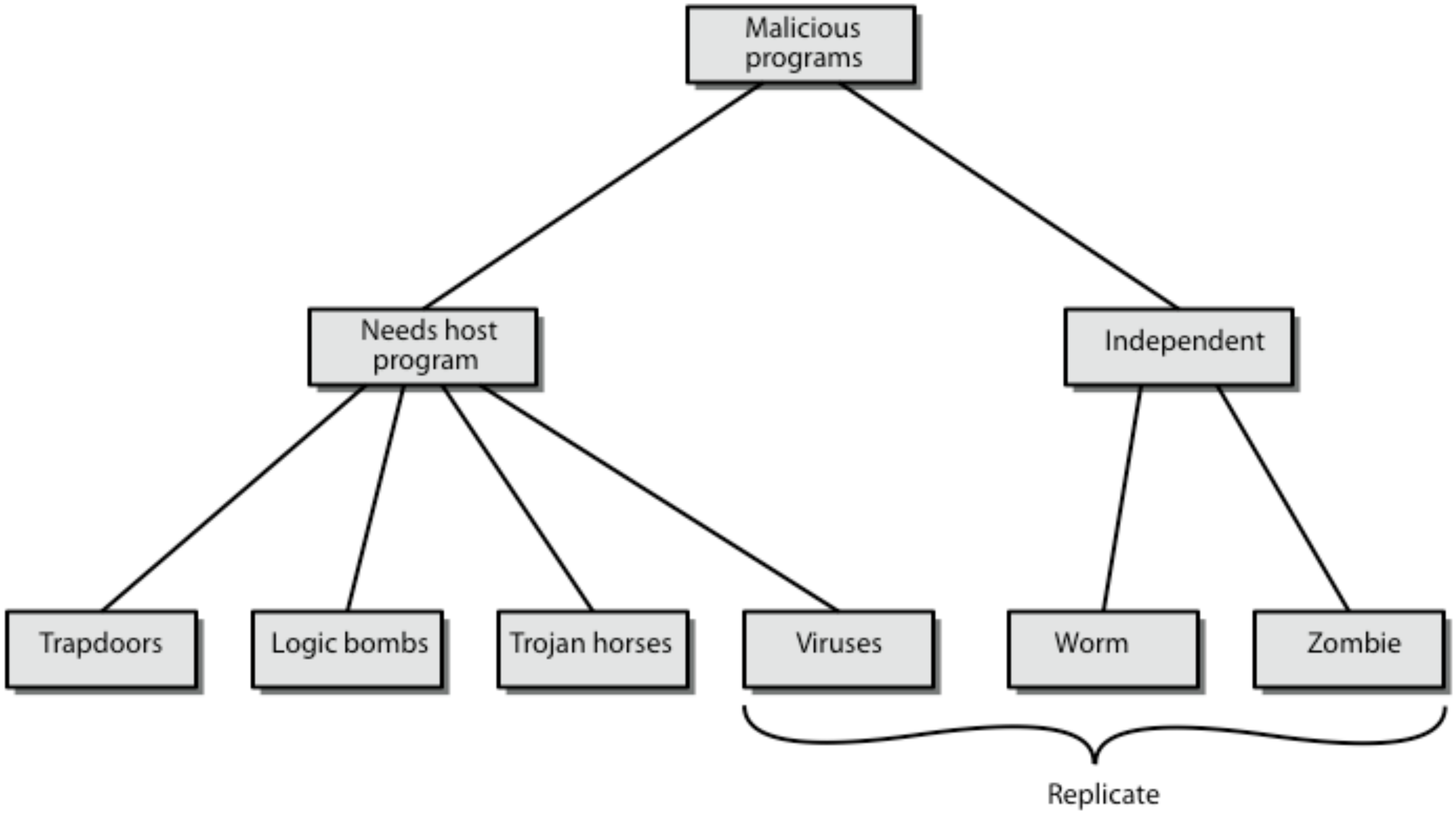
- NIST SP 800-83 (Guide to Malware Incident Prevention and Handling for Desktops and Laptops, July 2013):

“Malware (Malicious software) is a program that is inserted into a system, usually covertly, with the intent of compromising the confidentiality, integrity, or availability of victim’s data, applications or operating systems or otherwise annoying or disrupting the victim.”

- Malicious mobile code is malware that is furthermore designed to move:
  - From computer to computer and network to network

Classification

- Early approach



From Stallings (Cryptography & Network Security)

Viruses

- A computer virus, first appeared in the early 1980s, is a piece of software that can “infect” other programs, or indeed any type of executable content, by modifying them.
  - Injecting the original code with a routine to make copies of the virus code, which can then go on to infect other content
- In early years, viruses dominated the malware scene due to lack of user authentication and access control
- Tighter access controls on modern operating systems resulted in development of macro viruses that exploited the active content supported by some documents types, such as Microsoft Word or Excel files, or Adobe PDF documents

Virus logic

- A virus is often quite a simple program, not just conceptually but also in implementation.
- A direct action virus can be modelled in terms of an algorithm such as the following:

```
begin
  Look for (one or more infectable objects)
    If (none found)
    then
      exit
    else (infect object or objects).
    endif
end
```

- Direct action viruses are only active when an infected object is active.

- A lot of viruses install themselves into the memory of the host computer when the original virus program is executed.
- This means that even after the original virus program is closed, new objects can be infected without having to run anything else.
- These are referred to as memory resident viruses.
- Hybrid viruses are both direct action and memory resident.

Viruses’ components

- **Infection mechanisms**: The means by which a virus spreads or propagates, enabling it to replicate.
  - also referred as the **infection vector**
- **Trigger**: The event or condition that determines when the payload is activated or delivered
  - Sometimes known as **logic bomb**
- **Payload**: What the virus does, besides spreading
  - May involve damage of benign but noticeable activity

Virus lifetime phases

- **Dormant phase**: The virus is idle. It may be waiting for a trigger before propagation begins. Not all viruses have this.
- **Propagation phase**: The virus places a copy of itself into other programs or into certain system areas on the disk.
- **Triggering phase**: The virus is activated to perform the function.
- **Execution phase**: The function is performed.

Classification of Viruses

- … and according to the method of concealment
  - **Stealth virus:** designed to hide itself from detection by anti-virus software. Thus, the entire virus, not just a payload, is hidden.
  - **Encrypted virus**: use encryption to obscure its content
  - **Polymorphic virus**: change form each time they are inserted into another program.
  - **Metamorphic virus**:
    - a higher order of polymorphic viruses.
    - Not only do they change in form between transitions they can be completely re-written

Stealth viruses

- Viruses with this technology explicitly try to hide all of themselves from detection.
- One typical approach is to include compression.
  - Detecting that a file has changed by checking the length won't work anymore.
- The virus can also include placing "intercept logic" to capture dangerous queries (from the viewpoint of the virus) and provide acceptable responses.

```
    program CV :=

{goto main;
    01234567;

    subroutine infect-executable :=
          {loop:
              file := get-random-executable-file;
          if (first-line-of-file = 01234567) then goto loop;
        (1)     compress file;
        (2)     prepend CV to file;
        }

main:   main-program :=
          {if ask-permission then infect-executable;
        (3)     uncompress rest-of-file;
        (4)     run uncompressed file;}
        }
```

Encrypted Viruses

- Viruses which are encrypted with a cipher.
- Why?
    - To avoid detection. The virus code is hidden!
- How?
    - The virus code is encrypted, except for the decryption routine and key.
- Does it work?
    - We will look at detection a little later.

Deciphering routine	Enciphered virus	Key

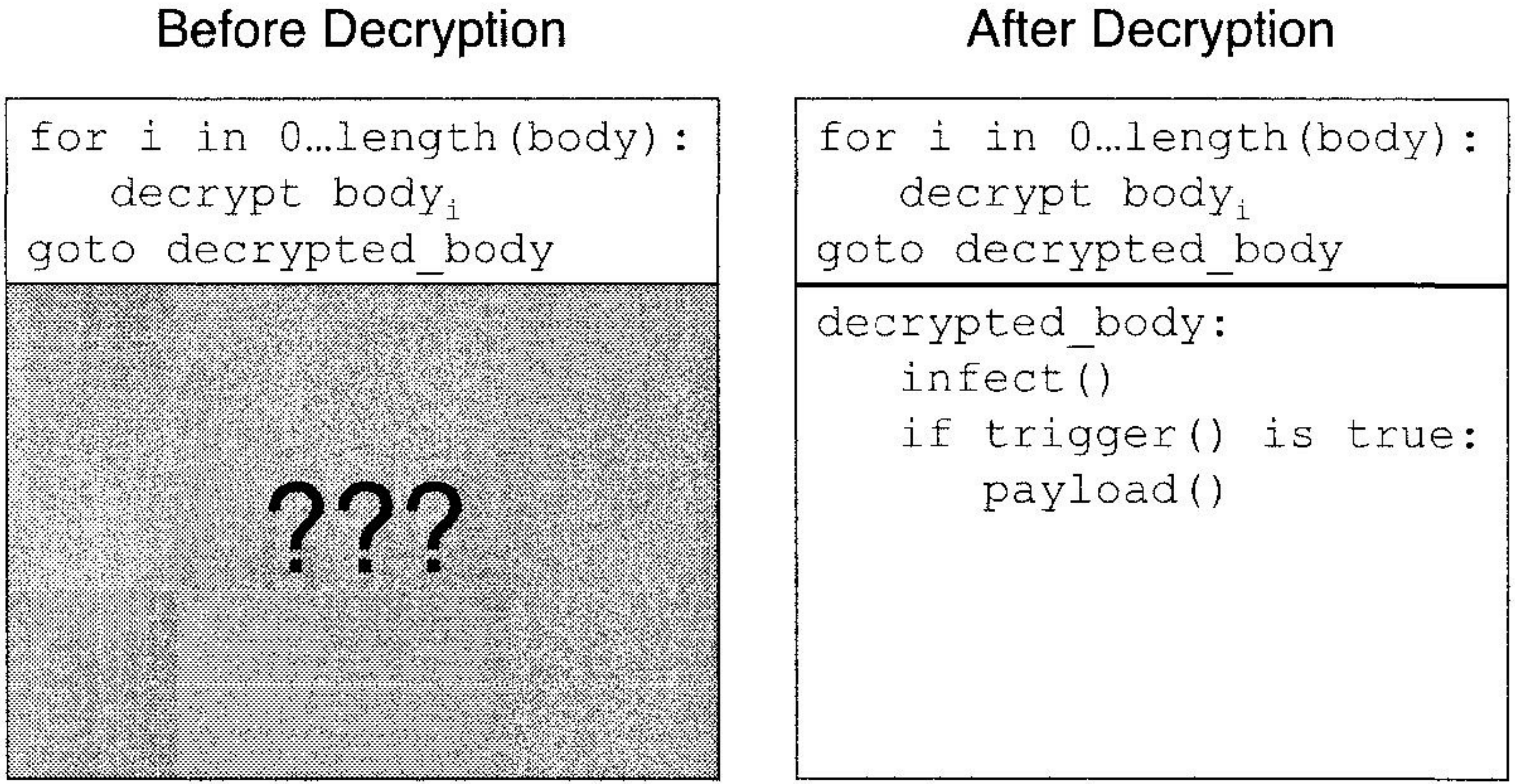


Figure 3.5. Encrypted virus pseudocode

- From the book: “A Pathology of Computer Viruses” by Ferbrache.

An example of encrypted virus

```
(* initialize the registers with the keys *)
rA := k1;
rB := k2;
(* initialize rC with the message *)
rC := sov;
(* the encipherment loop *)
while (rC != eov) do begin
        (* encipher the byte of the message *)
        (* ˆrC means the value at the address stored
           in rC *)
        (ˆrC) := (ˆrC) xor rA xor rB;
        (* advance all the counters *)
        rC := rC + 1;
        rA := rA + 1;
end
```

D. Ferbrache. *A Pathology of Computer Viruses*, Springer 1992

Polymorphic Viruses

- Polymorphic viruses change the form of its decryption routine each time it inserts itself into another program.
- If the viruses is encrypted, as is fairly common, the decryption code is the segment of the virus that is changed.
  - For example, at the instruction level, all the following have exactly the same effect.

```
add 0 to operand
Logical AND 1 with operand
no operation
subtract 0 from operand
```

```
(* initialize the registers with the keys *)
rA := k1;
rA := rA + 0;                     (* random line *)
rB := k2;
rD := k1 + k2;                    (* random line *)
(* initialize rC with the message *)
rC := sov;
rC := rC + 1;                     (* random line *)
(* the encipherment loop *)
while (rC != eov) do begin
        rC := rC - 1;             (* random line *)
        (* encipher the byte of the message *)
        (* ^rC means the value at the address stored in rC *)
        (^rC) := (^rC) xor rA xor rB;
        (* advance all the counters *)
        rC := rC + 2;             (* counter incremented ... *)
        (* to handle random line X *)
        rD := rD - 0;             (* random line *)
        rA := rA + 1;
end
(* the next block does nothing *)
while (rC != sov) do begin
        rD := rD - 1;
        rC = rC - 1;
end
```

Metamorphic viruses

- These are, in some sense, a higher order of polymorphic viruses.
- Not only do they change in form between transitions they can be completely re-written.
  - They can re-write in a version suitable for executables on a different platform too.

Countermeasures

- Malware acts as both data and instructions:
  - A virus inserts code (a set of instructions) into another program.
  - The set of instructions is treated as data.
  - The virus executes itself, the set of instructions is treated as an executable.
  - Protection:
    - Treat all programs as type "data“.
    - Some certifying authority can change the type to executable, after verification takes place.

- Against Malicious code assuming the identity of a user:
  - When a user executes malicious code, that code can access and affect objects within the user's protection domain.
  - Protection: Limiting the objects accessible to a given process run by the user.
    - Information Flow Metrics.
    - Reducing the Rights.
    - Sandboxing.

- **Information flow metrics**:
- Define the flow distance metric $fd(x)$ for some information x as follows:
  - Initially, all information has $fd(x) = 0$.
  - Whenever x is shared, $fd(x)$ increases by 1.
  - Whenever x is used as input to a computation, the flow distance of the output is the maximum of the flow distances of the input.
- Information is accessible only while the distance is **less** than some value V.
- Information vs data?

- Example of applying the flow distance metric.
  - A, B, and C work on the same computer.
  - $V_A = 3$. $V_B = V_C = 2$.
  - A creates a program P containing a virus. ☺
  - B executes P.
    - The contents of P have a flow distance of 0, so when the virus infects B's file Q, the flow distance of the virus is 1, and so B can access it.
      - Hence, the copying succeeds.
  - C executes Q, when the virus tries to spread to her files, its flow distance increases to 2.
    - Hence, the infection is not permitted, because C can only access information with a flow distance 0 or 1.
    - C can however execute P and it will flow. ☹

- **Reducing the Rights**:
  - The user can reduce their associated work domain when running a suspect program.
  - This follows from the principle of least privilege:

A subject should be given only those privileges that it needs in order to complete its task.

More on defence

- **Sandboxing** can be used.
  - Virtual environments.
    - From Symantec report (April 2015).
      - "In 2014, up to 28 percent of all malware was "virtual machine aware.""
- Restrict sharing by controlling the domain boundaries:
  - Restrict users in different protecting domains from sharing programs or data.
  - Programs to be protected should be placed at the lowest level of an implementation of a multilevel security policy.

Detection

- Normal behaviour of a system is usually different from the activity profile of an infected system.
  - Virus monitors monitor known methods of virus activity, such as attempts to write to a boot sector, modify interrupt vectors, write to system files... and detect abnormal behaviour of the system.
- **Advantages:**
  - Works for all viruses.
  - Detection is before (complete) infection.
- **Disadvantages:**
  - To detect a high percentage of viruses, the sensitivity of the monitor must be set high and this may generate many false alarms.

- Theorem (by Cohen):

  It is undecidable whether an arbitrary program contains a computer virus.

There are formal definitions of viruses that allow this type of result to be derived.

Multiple copy testing …

- Run several copies of the “same program or algorithm”.
  - This is not simply running the same program several times.
- Majority rules…
- The check could be based on results, calls made, efficiency …
- Majority still might not be trusted, because they might all be corrupted.
- But different performances can imply action needs to be taken.

Signature scanning

- Signature scanning (signature= search string= scan string) is the simplest and the most common approach to virus detection.
- Signature extraction is a non-trivial process:
    - The infection is disassembled and the key portions are identified.
    - The key portions are combined to form a signature.
    - The signature is checked against a large library of programs to reduce the chance of false positives occurring when signature accidentally matches some library code.

```
seg000:7C40 BE 04 00                  mov     si, 4           ; Try it 4 times
seg000:7C40                                                   ;
seg000:7C43
seg000:7C43             next:                                 ; CODE XREF: sub_7C3A+27↓j
seg000:7C43 B8 01 02                  mov     ax, 201h        ; read one sector
seg000:7C46 0E                        push    cs
seg000:7C47 07                        pop     es
seg000:7C48                           assume es:seg000
seg000:7C48 BB 00 02                  mov     bx, 200h        ; to here
seg000:7C4B 33 C9                     xor     cx, cx
seg000:7C4D 8B D1                     mov     dx, cx
seg000:7C4F 41                        inc     cx
seg000:7C50 9C                        pushf
seg000:7C51 2E FF 1E 09 00            call    dword ptr cs:9  ; int 13
seg000:7C56 73 0E                     jnb     short fine
seg000:7C58 33 C0                     xor     ax, ax
seg000:7C5A 9C                        pushf
seg000:7C5B 2E FF 1E 09 00            call    dword ptr cs:9  ; int 13
seg000:7C60 4E                        dec     si
seg000:7C61 75 E0                     jnz     short next
seg000:7C63 EB 35                     jmp     short giveup
```

Figure 11.2 A code snippet of the Stoned virus loaded to IDA.

- This is from page 428 of The Art of Computer Virus Research and Defense by Peter Szor (2005).

- **Advantages**:
  - Signature scanning can be used against Trojan horses, logic bombs and other malicious software.
- **Disadvantage:**
  - Scanning cannot find new viruses before their patterns are known.
  - It is also ineffective against polymorphic viruses.
- First-generation scanner's use virus signatures only.
- GPU's can be used to speed up signature processing! https://developer.nvidia.com/gpugems/gpugems3/part-v-physics-simulation/chapter-35-fast-virus-signature-matching-gpu

2^{nd}, 3^{rd} and 4^{th} generation scanners

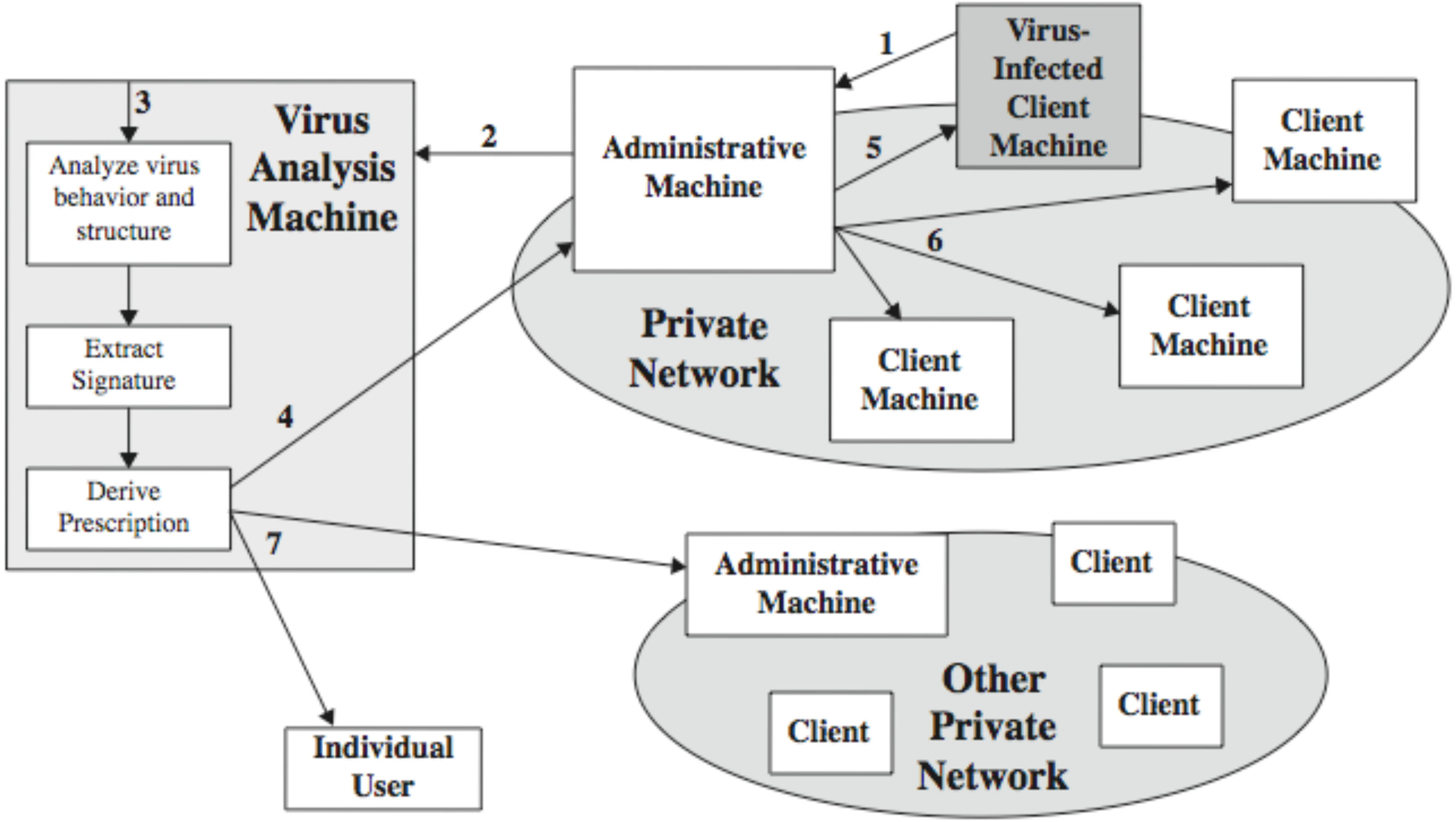
- **Second generation** scanners don’t just use specific signatures, since the signatures of many viruses change (polymorphic).
  - User heuristic rules to search for probable malware instances
  - Integrity checks (e.g. checksum) can also be applied

- Manipulation Detection Codes (or MDC's).
    - Apply some function to a file to obtain a set of bits called the signature block and then protect that block.
    - If the recomputing the signature block, the result differs from the stored signature block, the file has changed, possibly as a result of a malicious code.
    - We can use hash functions here!

- … and timestamps are useful too.
- The time of the last change to a program is kept separate from the environment that the program is stored.
- Timestamps should be frequently checked to ascertain the integrity of the program.

- Third generation scanners detect viruses by behaviour.
    - For example, attempts to interact inappropriately with certain system files could trigger detection.
- Fourth generation basically use a collect of antivirus techniques together.
- One quite important, and fairly new tool, is virtualisation.
    - We can run a virus in an isolated but realistic environment and see what happens.

Digital Immune System



Originally IBM, subsequently Symantec.

Figure 6.6, page 213 of Stallings and Brown. Edition 2

- Objective: To provide a rapid response so viruses can be stamped out soon after being introduced.
    - On detecting a new virus, the immune system captures and analyses it, adds detection and shielding information, removes it, and passes information about that virus to other systems so it can be detected before being allowed to run elsewhere.

1. Monitoring programs on the PC's use heuristics to infer a virus may be present. A copy is passed to an administrative machine.

2. Admin machine encrypts the "virus" and sends it to a central virus analysis machine.

3. The central virus analysis machine provides a safe environment for analysis, and produces a prescription for virus identification and removal.

4. Prescription sent back to the admin machine.

5. Prescription forwarded to the infected client.

6. Prescription forwarded to other clients in the organization.

7. Worldwide subscribers get regular antivirus updates.